LECTURE NOTES IN ELECTRICAL ENGINEERING 9

Rajkumar Buyya
Mukaddim Pathan
Athena Vakali *Editors*

# Content Delivery Networks

Springer

MICROSOFT-1010
Microsoft Corp. v. Sandpiper CDN, LLC
IPR2026-00180

Lecture Notes Electrical Engineering

Volume 9

Rajkumar Buyya · Mukaddim Pathan · Athena Vakali (Eds.)

# Content Delivery Networks

Springer

*Editors*

Rajkumar Buyya
University of Melbourne
Dept. Computer Science&
Software Engineering
111 Barry Street
Carlton VIC 3053
Australia
raj@csse.unimelb.edu.au

Mukaddim Pathan
University of Melbourne
Dept. Computer Science &
Software Engineering
111 Barry Street
Carlton VIC 3053
Australia
apathan@csse.unimelb.edu.au

Athena Vakali
Aristotle University of
Thessaloniki
Dept. Informatics
541 24 Thessaloniki
Greece
avakali@csd.auth.gr

ISBN: 978-3-540-77886-8 e-ISBN: 978-3-540-77887-5

Library of Congress Control Number: 2008930075

© 2008 Springer-Verlag Berlin Heidelberg

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

*Cover design:* eStudio Calamar S.L.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

*To my personal and professional family – Raj*
*To my parents – Mukaddim*
*To my daughters – Athena*

# Preface

The emergence of Web as a ubiquitous media for sharing content and services has led to the rapid growth of the Internet. At the same time, the number of users accessing Web-based content and services are growing exponentially. This has placed a heavy demand on Internet bandwidth and Web systems hosting content and application services. As a result, many Web sites are unable to manage this demand and offer their services in a timely manner.

Content Delivery Networks (CDNs) have emerged to overcome these limitations by offering infrastructure and mechanisms to deliver content and services in a scalable manner, and enhancing users' Web experience. Applications of CDNs can also be found in many communities, such as academic institutions, advertising media and Internet advertisement companies, data centers, Internet Service Providers (ISPs), online music retailers, mobile operators, consumer electronics manufacturers, and other carrier companies. Along with the proliferation, formation, and consolidation of the CDN landscape, new forms of Internet content and services are coming into picture while distribution and management of content is introducing new challenges in this domain. This raises new issues in the architecture, design and implementation of CDNs. The technological trends in this domain need to be explored in order to provide an exclusive research roadmap to the CDN community.

The book, entitled *"Content Delivery Networks"* offers the state-of-the-art CDN concepts, principles, characteristics, applications, platforms, design tips and hints, modeling, simulation, engineering approaches, and recent technological developments. The book builds on academic and industrial research and developments, and case studies that are being carried out at many different institutions around the world. In addition, the book identifies potential research directions and technologies that drive future innovations. We expect the book to serve as a valuable reference for larger audience such as systems architects, practitioners, product developers, researchers, and graduate level students.

**Overview and scope of the book:** This book will enable the readers to understand the basics, to identify the underlying technology, to summarize their knowledge on concepts, ideas, principles and various paradigms which span on broad CDNs areas. Therefore, aspects of CDNs in terms of basics, design process, practice, techniques,

performances, platforms, applications, and experimental results have been presented in a proper order. Fundamental methods, initiatives, significant research results, as well as references for further study have also been provided. Comparison of different design and development approaches are described at the appropriate places so that new researchers as well as advanced practitioners can use the CDNs evaluation as a research roadmap. All the contributions have been reviewed, edited, processed, and placed in the appropriate order to maintain consistency so that any reader irrespective of their level of knowledge and technological skills in CDNs would get the most out of it. The book is organized into three parts, namely, Part I: CDN Fundamentals; Part II: CDN Modeling and Performance; and Part III: Advanced CDN Platforms and Applications. The organization ensures the smooth flow of material as successive chapters build on prior ones. In particular, the topics of the book are the following:

- CDN basics and state of the art
- A taxonomy for categorizing CDN technologies
- Dynamic, scalable and efficient content replication techniques
- Content distribution and management
- Integrated use of replication with caching and its performance
- Request redirection for dynamic content
- Economics-informed modeling of CDNs
- Pricing schemes and CDN business models
- Mathematical modeling for resource allocation and management
- CDN performance
- CDN internetworking scenarios, architecture, and methodology
- Media streaming
- Dynamic CDNs and QoS-based adaptive content delivery
- Mobile dynamic CDNs
- Applications: live and on-demand video services, content delivery for community networks

Part I of the book focuses on the basic ideas, techniques, and current practices in CDNs. In Chap. 1, Pathan et al. provide an introduction to CDNs and their origins, evolution, and the start-of-the-art. This chapter defines CDNs and related terminologies, identifies its uniqueness when compared to related distributed computing paradigms, provides insights for CDNs, and presents authors' visions for future technological evolutions in the CDN domain. As there exist a wide range of studies covering different aspects of CDNs such as content distribution, replication, caching, and Web server placement, in Chap. 2, Pathan and Buyya present a comprehensive taxonomy of CDNs with a broad coverage of applications, features, and implementation techniques. In Chap. 3, Chen highlights on the need of a dynamic, scalable, and efficient replication technique for CDNs. In this context, the author presents algorithms for dynamic and self-organized replica placements respecting client QoS and server capacity constraints. In Chap. 4, Cardellini et al. explore the issues of content delivery through CDNs, with a special focus on the delivery of dynamically generated and personalized content. To analyze and

model simulations of various caching scheme, along with the integrated use of caching and replication, in Chap. 5, Stamos et al. present related design methodology and share implementation experiences, while cover various topics related to Web caching in a CDN simulation framework. In Chap. 6, Ranjan describes request redirection techniques in CDNs and presents a proof-of-the-concept implementation of a technique called WARD to assist the redirection of dynamic content.

Part II of this book focuses on the economic and mathematical modeling of CDNs and their performance. Building on game theory, in Chap. 7, Christin et al. present a cost-based model for agents participating in a CDN overlay and analyze incentives in link establishments in CDNs. In Chap. 8, Hosanagar discusses the economics of content delivery market and provides a model to capture content providers' value from CDN services and uses that to discuss pricing policies. In Chap. 9, Bektaş: and Ouveysi demonstrate how a variety of resource management and allocation problems in CDN domain can be formulated in terms of mathematical models. Sitaraman et al. present global overlay routing supported by Akamai along with its performance and availability benefits in Chap. 10.

Part III, the final part of the book, focuses on advanced CDN platforms and applications with wide appeal. In Chap. 11, Yoshida describes FCAN, a dynamic CDN network to alleviate flash crowds. Fortino et al. presents a CDN-based architecture supporting the collaborative playback service in Chap. 12, by describing the Hierarchical COoperative COntrol Protocol (HCOCOP) which enables the shared media streaming control in collaborative playback sessions. In Chap. 13, Czyrnek et al. address the key aspects of the multimedia CDN design based on the presentation of iTVP, a platform which is built for IP-based delivery of multimedia content on a country-wide scale to a large number of concurrent users. The information dissemination techniques in mobile CDNs, along with related challenges and current status are presented in Chap. 14 by Loulloudes et al. In Chap. 15, Plagemann et al. discuss the infrastructures for community networks based on CDNs. Finally, in the last chapter of the book, Pathan et al. present different models for internetworking between CDNs and identify the challenges in realizing them.

**Acknowledgements:** The book came into light due to the direct and indirect involvement of many researchers, academicians, developers, designers, and industry practitioners. Therefore, we acknowledge and thank the contributing authors, research institutions, and companies whose papers, reports, articles, notes, Web sites, study materials have been referred to in this book. Furthermore, many of the authors have acknowledged their respective funding agencies and co-researchers, who made significant influence in carrying out research. Finally, we offer our special appreciation to Springer and its publishing editor, Christoph Baumann, for helping us to bring this book out in record time.

Prior technical sources are acknowledged citing them at appropriate places in the book. In case of any error we would like to receive feedback so that it could be taken into consideration in the next edition.

We hope that this book will serve as a valuable text for students especially at graduate level and reference for researchers and practitioners working in the CDNs and its emerging consumer applications.

The University of Melbourne, Australia *Rajkumar Buyya*
The University of Melbourne, Australia *Mukaddim Pathan*
Aristotle University of Thessaloniki, Greece *Athena Vakali*

# Contents

**Part I CDN Fundamentals**

**1 Content Delivery Networks: State of the Art, Insights, and Imperatives** . . . . . 3
Mukaddim Pathan, Rajkumar Buyya and Athena Vakali

**2 A Taxonomy of CDNs** . . . . . 33
Mukaddim Pathan and Rajkumar Buyya

**3 Dynamic, Scalable, and Efficient Content Replication Techniques** . . . 79
Yan Chen

**4 Content Delivery and Management** . . . . . 105
Claudia Canali, Valeria Cardellini, Michele Colajanni and Riccardo Lancellotti

**5 Caching Techniques on CDN Simulated Frameworks** . . . . . 127
Konstantinos Stamos, George Pallis and Athena Vakali

**6 Request Redirection for Dynamic Content** . . . . . 155
Supranamaya Ranjan

**Part II CDN Modeling and Performance**

**7 Economics-Informed Design of CDNs** . . . . . 183
Nicolas Christin, John Chuang and Jens Grossklags

**8 CDN Pricing** . . . . . 211
Kartik Hosanagar

**9 Mathematical Models for Resource Management and Allocation in CDNs** . . . . . 225
Tolga Bektaş and Iradj Ouveysi

**10 Performance and Availability Benefits of Global Overlay Routing** . . . 251
Hariharan S. Rahul, Mangesh Kasbekar, Ramesh K. Sitaraman, and Arthur W. Berger

**Part III Advanced CDN Platforms and Applications**

**11 Dynamic CDN Against Flash Crowds** . . . . . 275
Norihiko Yoshida

**12 Collaborative Media Streaming Services Based on CDNs** . . . . . 297
Giancarlo Fortino, Carlo Mastroianni, and Wilma Russo

**13 CDN for Live and On-Demand Video Services over IP** . . . . . 317
Mirosław Czyrnek, Ewa Kuśmierek, Cezary Mazurek, Maciej Stroiński, and Jan Węglarz

**14 Information Dissemination in Mobile CDNs** . . . . . 343
Nicholas Loulloudes, George Pallis, and Marios D. Dikaiakos

**15 Infrastructures for Community Networks** . . . . . 367
Thomas Plagemann, Roberto Canonico, Jordi Domingo-Pascual, Carmen Guerrero, and Andreas Mauthe

**16 Internetworking of CDNs** . . . . . 389
Mukaddim Pathan, Rajkumar Buyya, and James Broberg

**Index** . . . . . 415

# Contributors

Tolga Bektaş
School of Management, University of Southampton, Highfield, Southampton SO17 1BJ, UK, e-mail: T.Bektas@soton.ac.uk

Arthur W. Berger
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA and Akamai Technologies, Cambridge, MA 02142, USA, e-mail: awberger@csail.mit.edu

James Broberg
GRIDS Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, e-mail: brobergj@csse.unimelb.edu.au

Rajkumar Buyya
GRIDS Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, e-mail: raj@csse.unimelb.edu.au

Claudia Canali
University of Modena and Reggio Emilia, 41100 Modena, Italy, e-mail: claudia.canali@unimore.it

Roberto Canonico
Consorzio Interuniversitario Nazionale per l'Informatica CINI, – Laboratorio Nazionale per l'Informatica e la Telematica Multimediali ITEM at University of Napoli, Italy, e-mail: roberto.canonico@unina.it

Valeria Cardellini
Dipartimento di Informatica, Sistemi e Produzione, Università di Roma "Tor Vergata", Via del Politecnico 1, 00133 Roma, Italy, e-mail: cardellini@ing.uniroma2.it

Yan Chen
Department of Electrical Engineering and Computer Science, Northwestern University, 2145 Sheridan Road, Evanston, IL, USA, e-mail: ychen@northwestern.edu

Nicolas Christin
Carnegie Mellon University, INI and CyLab Japan, 1-3-3-17 Higashikawasaki-cho, Chuo-ku, Kobe, 650-0044, Japan, e-mail: nicolasc@cmu.edu

John Chuang
School of Information, The University of California at Berkeley 102 South Hall, Berkeley, CA 94720, USA, e-mail: chuang@ischool.berkeley.edu

Michele Colajanni
University of Modena and Reggio Emilia, 41100 Modena, Italy,
e-mail: michele.colajanni@unimore.it

Mirosław Czyrnek
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61–704 Poznan, Poland, e-mail: majrek@man.poznan.pl

Marios D. Dikaiakos
Department of Computer Science, University of Cyprus, 75 Kallipoleos Str. 1678, Nicosia, Cyprus, e-mail: mdd@cs.ucy.ac.cy

Jordi Domingo-Pascual
Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya, Jordi Girona, 1–3. Campus Nord. Barcelona 08034, Spain,
e-mail: jordi.domingo@ac.upc.es

Giancarlo Fortino
Dipartimento di Informatica, Elettronica e Sistemistica (DEIS), Università della Calabria, Rende (CS), Italy, e-mail: g.fortino@unical.it

Jens Grossklags
School of Information, The University of California at Berkeley, 102 South Hall, Berkeley, CA 94720, USA, e-mail: jensg@ischool.berkeley.edu

Carmen Guerrero
Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, Spain,
e-mail: carmen.guerrero@uc3m.es

Kartik Hosanagar
Operations and Information Management, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104, USA, e-mail: kartikh@wharton.upenn.edu

Mangesh Kasbekar
Akamai Technologies, Staines, TW18 4EP, UK, e-mail: mkasbeka@akamai.com

Ewa Kuśmierek
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61–704 Poznan, Poland, e-mail: kusmiere@man.poznan.pl

Riccardo Lancellotti
University of Modena and Reggio Emilia, 41100 Modena, Italy, e-mail: riccardo.lancellotti@unimore.it

Nicholas Loulloudes
Department of Computer Science, University of Cyprus, 75 Kallipoleos Str. 1678, Nicosia, Cyprus, e-mail: loulloudes.n,@cs.ucy.ac.cy

Carlo Mastroianni
ICAR-CNR (Italian National Research Council), Rende (CS), Italy, e-mail: mastroianni@icar.cnr.it

Andreas Mauthe
InfoLab 21, Computing Department, Lancaster University, Lancaster LA1 4WA, UK, e-mail: andreas@comp.lancs.ac.uk

Cezary Mazurek
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61–704 Poznan, Poland, e-mail: mazurek@man.poznan.pl

Iradj Ouveysi
Honorary research fellow, Electrical and Electronic Engineering Department, The University of Melbourne, Australia, e-mail: iradjouveysi@yahoo.co.uk

George Pallis
Department of Computer Science, University of Cyprus, 75 Kallipoleos Str. 1678, Nicosia, Cyprus, e-mail: gpallis@cs.ucy.ac.cy

Mukaddim Pathan
GRIDS Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, e-mail: apathan@csse.unimelb.edu.au

Thomas Plagemann
Department of Informatics, University of Oslo, Blindern N-0316 Oslo, Norway, e-mail: plageman@ifi.uio.no

Hariharan S. Rahul
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA, e-mail: rahul@csail.mit.edu

Supranamaya Ranjan
Narus Inc., 500 Logue Ave, Mountain View, CA 94043, USA,
e-mail: soups.ranjan@gmail.com

Wilma Russo
Dipartimento di Informatica, Elettronica e Sistemistica (DEIS), Università della Calabria, Rende (CS), Italy, e-mail: russow@si.deis.unical.it

Ramesh K. Sitaraman
Department of Computer Science, The University of Massachusetts, Amherst, MA 01003–4610, USA, e-mail: ramesh@cs.umass.edu

Konstantinos Stamos
Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece, e-mail: kstamos@csd.auth.gr

Maciej Stroiński
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61–704 Poznan, Poland, e-mail: stroins@man.poznan.pl

Athena Vakali
Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece, e-mail: avakali@csd.auth.gr

Jan Węglarz
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61–704 Poznan, Poland, e-mail: weglarz@man.poznan.pl

Norihiko Yoshida
Division of Mathematics, Electronics and Informatics, Saitama University, Saitama 338-8570, Japan, e-mail: yoshida@mail.saitama-u.ac.jp

# Part I
# CDN Fundamentals

# Chapter 1
# Content Delivery Networks: State of the Art, Insights, and Imperatives

Mukaddim Pathan, Rajkumar Buyya and Athena Vakali

## 1.1 Introduction

Over the last decades, users have witnessed the growth and maturity of the Internet which has caused enormous growth in network traffic, driven by the rapid acceptance of broadband access, the increases in systems complexity, and the content richness. The over-evolving nature of the Internet brings new challenges in managing and delivering content to users, since for example, popular Web services often suffer congestion and bottlenecks due to the large demands posed on their services. Such a sudden spike in Web content requests (e.g. the one occurred during the 9/11 incident in USA) is often termed as flash crowds [14] or SlashDot [11] effects. It may cause heavy workload on particular Web server(s), and as a result a "hotspot" [14] can be generated. Coping with such unexpected demand causes significant strain on a Web server and eventually the Web servers are totally overwhelmed with the sudden increase in traffic, and the Web site holding the content becomes temporarily unavailable.

A Content Delivery Network (CDN) [47, 51, 54, 61, 63] is a collaborative collection of network elements spanning the Internet, where content is replicated over several mirrored Web servers in order to perform transparent and effective delivery of content to the end users. Collaboration among distributed CDN components can occur over nodes in both homogeneous and heterogeneous environments. CDNs have evolved to overcome the inherent limitations of the Internet in terms of user perceived Quality of Service (QoS) when accessing Web content. They provide services that improve network performance by maximizing bandwidth, improving

Mukaddim Pathan
GRIDS Lab, Department of CSSE, The University of Melbourne, Australia,
e-mail: apathan@csse.unimelb.edu.au

Rajkumar Buyya
GRIDS Lab, Department of CSSE, The University of Melbourne, Australia,
e-mail: raj@csse.unimelb.edu.au

Athena Vakali
Department of Informatics, Aristotle University of Thessaloniki, Greece,
e-mail: avakali@csd.auth.gr

R. Buyya et al. (eds.), *Content Delivery Networks*,
© Springer-Verlag Berlin Heidelberg 2008

accessibility, and maintaining correctness through content replication. The typical functionalities of a CDN include:

- *Request redirection and content delivery services,* to direct a request to the closest suitable CDN cache server using mechanisms to bypass congestion, thus overcoming flash crowds [14] or SlashDot [11] effects.
- *Content outsourcing and distribution services,* to replicate and/or cache content from the origin server to distributed Web servers.
- *Content negotiation services,* to meet specific needs of each individual user (or group of users).
- *Management services,* to manage the network components, to handle accounting, and to monitor and report on content usage.

The major application domains of CDNs are public content networking services, enterprise content networks, and edge services. As CDNs being a thriving research field, advances, solutions, and new capabilities are being introduced constantly. Therefore, in this chapter, we capture a "snapshot" of the state of the art at the time of writing this book. However, it can be expected that the core information and principles presented in this chapter will remain relevant and useful for the readers.

The remainder of this chapter is structured as follows: we start with providing an overview of CDNs. Next we describe the background highlighting the evolution of CDNs and identify uniqueness of CDNs from other related distributed computing paradigms. In Sect. 1.4 we provide insights for CDNs. The state of the art in CDN landscape is presented in Sect. 1.5. Our visions about future technological evolutions in CDNs domain follows next, along with a research roadmap in Sect. 1.7 by exploring future research directions. Finally, Sect. 1.8 concludes the chapter.

## 1.2 Overview

Figure 1.1 shows the model of a CDN where the replicated Web server clusters spanning the globe are located at the edge of the network to which end users are connected. A CDN distributes content to a set of Web servers, scattered over the globe, for delivering content to end users in a reliable and timely manner. The content is replicated either on-demand when users request for it, or it can be replicated beforehand, by pushing the content to the distributed Web servers. A user is served with the content from the nearby replicated Web server. Thus, the user ends up unknowingly communicating with a replicated CDN server close to it and retrieves files from that server.

### *1.2.1 Terminologies*

In the context of CDNs, *content delivery* describes an action of servicing content based on end user requests. *Content* refers to any digital data resources and

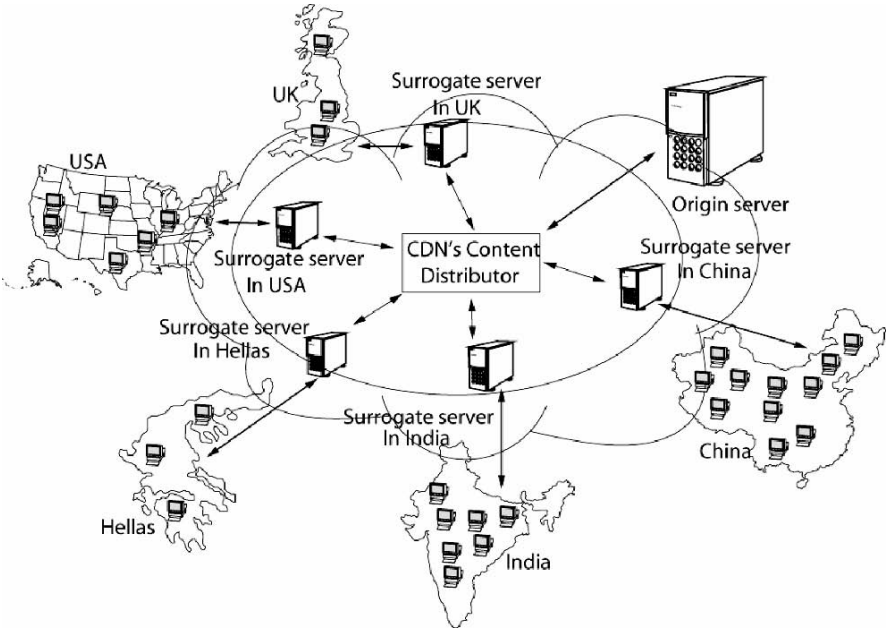



**Fig. 1.1** Model of a CDN

it consists of two main parts: the *encoded media* and *metadata* [53]. The encoded media includes static, dynamic, and continuous media data (e.g. audio, video, documents, images and Web pages). Metadata is the content description that allows identification, discovery, and management of multimedia data, and facilitates its interpretation. Content can be pre-recorded or retrieved from live sources; it can be persistent or transient data within the system [53]. CDNs can be seen as a new virtual overlay to the Open Systems Interconnection (OSI) network reference model [32]. This layer provides overlay network services relying on application layer protocols such as Hyper Text Transfer Protocol (HTTP) or Real Time Streaming Protocol (RTSP) for transport [26].

The three main entities in a CDN system are the following: *content provider*, *CDN provider*, and *end users*. A *content provider* or *customer* is one who delegates the Uniform Resource Locator (URL) name space of the Web objects to be distributed. The *origin server* of the content provider holds those objects. A *CDN provider* is a proprietary organization or company that provides infrastructure facilities to content providers in order to deliver content in a timely and reliable manner. *End users* or *clients* are the entities who access content from the content provider's Web site.

CDN providers use *caching* and/or *replica servers* located in different geographical locations to replicate content. CDN cache servers are also called *edge servers* or *surrogates*. The edge servers of a CDN are called *Web cluster* as a whole. CDNs distribute content to the edge servers in such a way that all of them share the same

content and URL. Client requests are redirected to the nearby optimal edge server and it delivers requested content to the end users. Thus, transparency for users is achieved. Additionally, edge servers send accounting information for the delivered content to the accounting system of the CDN for traffic reporting and billing purposes.

### *1.2.2 CDN Components*

Figure 1.2 shows the general architecture of a CDN system which involves four main components:

- The content-delivery component which consists of the origin server and a set of replica servers that deliver copies of content to the end users;
- The request-routing component which is responsible for directing client requests to appropriate edge servers and for interacting with the distribution component to keep an up-to-date view of the content stored in the CDN caches;
- The distribution component which moves content from the origin server to the CDN edge servers and ensures consistency of content in the caches; and
- The accounting component which maintains logs of client accesses and records the usage of the CDN servers. This information is used for traffic reporting

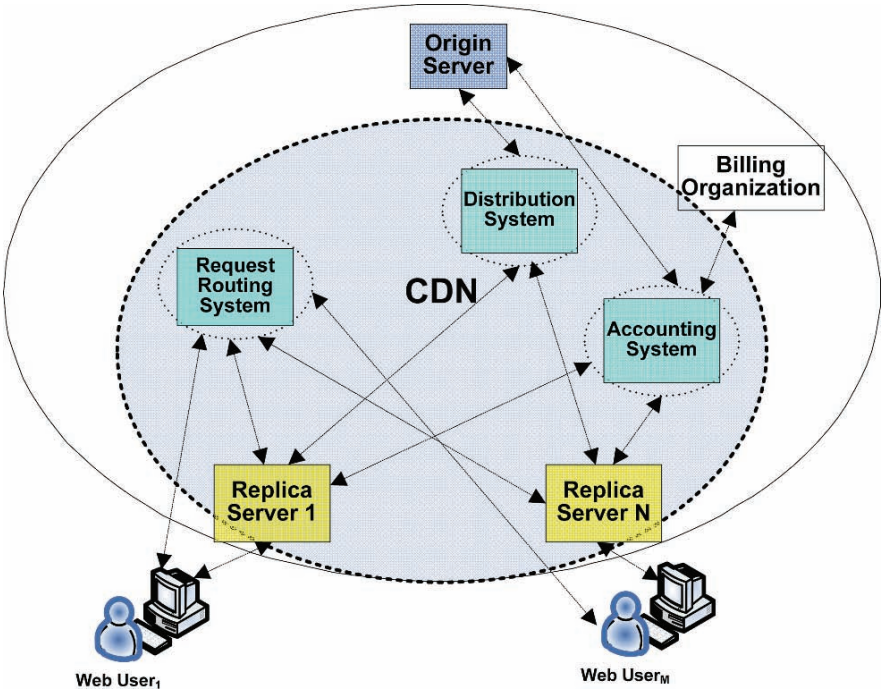



**Fig. 1.2** Architectural components of a CDN

and usage-based billing by the content provider itself or by a third-party billing organization.

A CDN focuses on building its network infrastructure to provide the following services and functionalities: storage and management of content; distribution of content among edge servers; cache management; delivery of static, dynamic, and streaming content; backup and disaster recovery solutions; and monitoring, performance measurement, and reporting.

A content provider (i.e. customer) can sign up with a CDN provider for service and have its content placed on the cache servers. In practice, CDNs typically host third-party content including static content (e.g. static HTML pages, images, documents, software patches), streaming media (e.g. audio, real time video), User Generated Videos (UGV), and varying content services (e.g. directory service, e-commerce service, file transfer service). The sources of content include large enterprises, Web service providers, media companies, and news broadcasters. Typical customers of a CDN are media and Internet advertisement companies, data centers, Internet Service Providers (ISPs), online music retailers, mobile operators, consumer electronics manufacturers, and other carrier companies. Each of these customers wants to publish and deliver their content to the end users on the Internet in a reliable and timely manner. End users can interact with the CDN by specifying the content/service request through cell phone, smart phone/PDA, laptop and desktop. Figure 1.3 depicts the different content/services served by a CDN provider to end users.

CDN providers charge their customers according to the content delivered (i.e. traffic) to the end users by their edge servers. CDNs support an accounting mechanism that collects and tracks client usage information related to request-routing, distribution, and delivery [26]. This mechanism gathers information in real time

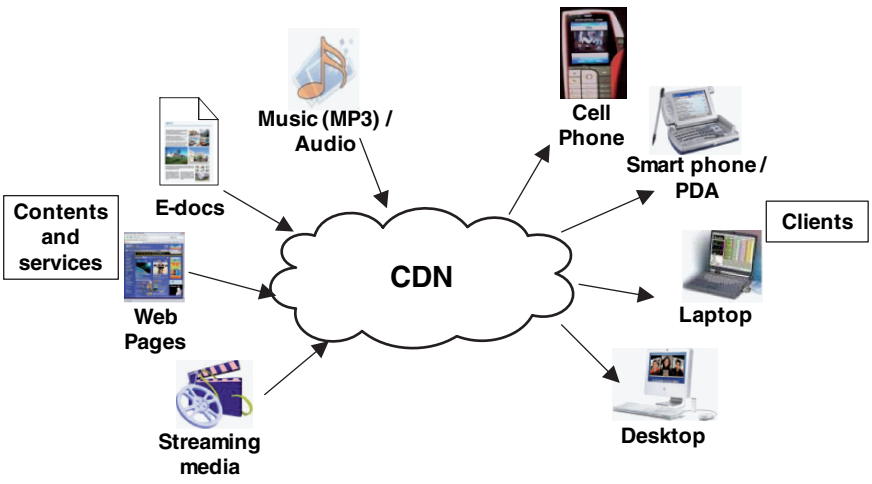



**Fig. 1.3** Content/services provided by a CDN

and collects it for each CDN component. This information can be used in CDNs for accounting, billing, and maintenance purposes. The average cost of charging of CDN services is quite high [35], often out of reach for many small to medium enterprises (SME) or not-for-profit organizations. The most influencing factors [47] affecting the price of CDN services include:

- Bandwidth usage which is measured by the content provider to charge (per Mbps) customers typically on a monthly basis;
- Variation of traffic distribution which characterizes pricing under different situations of congestion and bursty traffic;
- Size of the content replicated over edge servers which is a critical criterion for posing charges (e.g. price per GB) on customer audiences;
- Number of edge servers which capture the ability of a CDN provider to offer content at charges that will not overcome the typical caching scenarios; and
- Reliability and stability of the whole system and security issues of outsourcing content delivery also inhibit a cost of sharing confidential data which varies over different content providers on the basis of the type of the protected content.

## 1.3 Background and Related Systems

Content providers view the Web as a vehicle to bring rich content to their users since decreases in services quality, along with high access delays (mainly caused by long download times) leaves users in frustration. Companies earn significant financial incentives from Web-based e-business and they are concerned to improve the service quality experienced by the users while accessing their Web sites. As such, the past few years have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Web. When used together, the infrastructures supporting these technologies form a new type of network, which is often referred to as "content network" [26].

### *1.3.1 The Evolution of CDNs*

Several content networks attempt to address the performance problem by using different mechanisms to improve QoS:

- An initial approach is to modify the traditional Web architecture by improving the Web server hardware adding a high-speed processor, more memory and disk space, or maybe even a multi-processor system. This approach is not flexible, since small enhancements are not possible and at some point, the complete server system might have to be replaced [31].
- Caching proxy deployment by an ISP can be beneficial for the narrow bandwidth users accessing the Internet, since to improve performance and reduce bandwidth

utilization, caching proxies are deployed close to the users. Caching proxies may also be equipped with technologies to detect a server failure and maximize efficient use of caching proxy resources. Users often configure their browsers to send their Web request through these caches rather than sending directly to origin servers. When this configuration is properly done, the user's entire browsing session goes through a specific caching proxy. Thus, the caches contain most popular content viewed by all the users of the caching proxies.

- A provider may also deploy different levels of local, regional, international caches at geographically distributed locations. Such arrangement is referred to as *hierarchical caching*. This may provide additional performance improvements and bandwidth savings [17]. The establishment of server farms is a more scalable solution which has been in widespread use for several years. A server farm is comprised multiple Web servers, each of them sharing the burden of answering requests for the same Web site [31]. It also makes use of a Layer 4-7 switch (intelligent switching based on information such as URL requested, content type, and username, which can be found in layers 4-7 of the OSI stack of the request packet), Web switch or content switch that examines content requests and dispatches them among the group of servers. A server farm can also be constructed with surrogates instead of a switch [24]. This approach is more flexible and shows better scalability. Moreover, it provides the inherent benefit of fault tolerance. Deployment and growth of server farms progresses with the upgrade of network links that connects the Web sites to the Internet.
- Although server farms and hierarchical caching through caching proxies are useful techniques to address the Web performance problem, they have limitations. In the first case, since servers are deployed near the origin server, they do little to improve the network performance due to network congestion. Caching proxies may be beneficial in this case. But they cache objects based on client demands. This may force the content providers with a popular content source to invest in large server farms, load balancing, and high bandwidth connections to keep up with the demand. To address these limitations, another type of content network has been deployed in late 1990s. This is termed as *Content Distribution Network* or *Content Delivery Network*, which is a system of computers networked together across the Internet to cooperate transparently for delivering content to end users.

With the introduction of CDN, content providers started putting their Web sites on a CDN. Soon they realized its usefulness through receiving increased reliability and scalability without the need to maintain expensive infrastructure. Hence, several initiatives kicked off for developing infrastructure for CDNs. As a consequence, Akamai Technologies [1, 27] evolved out of an MIT research effort aimed at solving the flash crowd problem and scientists developed a set of breakthrough algorithms for intelligently routing and replicating content over a large network of distributed servers spanning the globe. Within a couple of years, several companies became specialists in providing fast and reliable delivery of content, and CDNs became a huge market for generating large revenues. The flash crowd events [14, 34] like the 9/11 incident in USA [10], resulted in serious caching problems for some sites. This influenced the providers to invest more in CDN infrastructure development, since

CDNs provide desired level of protection to Web sites against flash crowds. First generation CDNs mostly focused on static or Dynamic Web documents [36, 61]. On the other hand, for second generation of CDNs the focus has shifted to Video-on-Demand (VoD), news on-demand, audio and video streaming with high user interactivity. The CDNs of this generation may also be dedicated to deliver content to mobile devices. However, most of the research efforts on this type of CDNs are still in research phase and have not yet exclusively reached the market. We anticipate that the third generation CDNs would be community-based CDNs, i.e. it would be mainly driven by the common "people" or the average end users. More information on such community-based CDNs can be found in Chap. 15 of this book. Figure 1.4 shows the evolutions of CDNs over time with a prediction of their evolution in the upcoming years.

With the booming of the CDN business, several standardization activities also emerged since vendors started organizing themselves. The Internet Engineering Task Force (IETF) as an official body has taken several initiatives through releasing Request For Comments (RFCs) [15, 16, 24, 26] in relation to many research initiatives in this domain. Other than IETF, several other organizations such as Broadband Services Forum (BSF) [3], ICAP forum [6], Internet Streaming Media Alliance [7] have taken initiatives to develop standards for delivering broadband content, streaming rich media content – video, audio, and associated data – over the Internet. In the same breath, by 2002, large-scale ISPs started building their own CDN functionality, providing customized services.

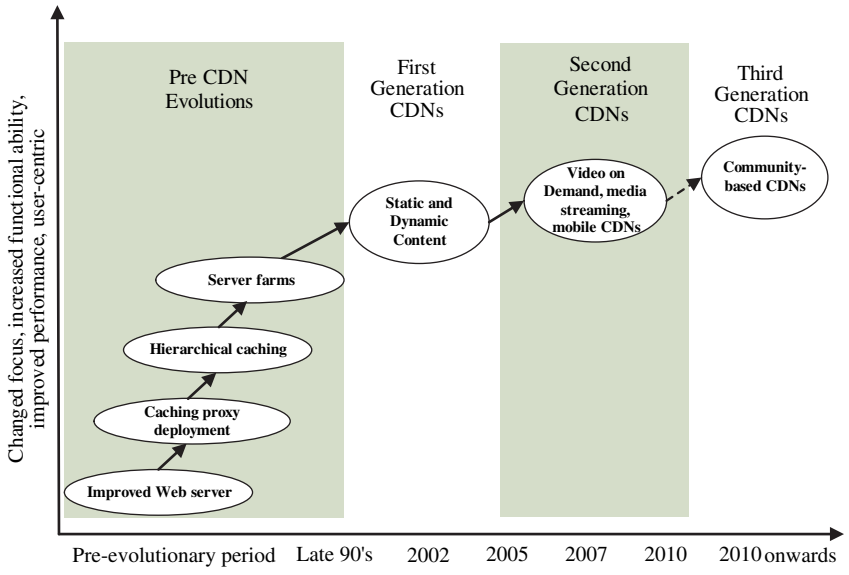



**Fig. 1.4** CDN evolutions

### *1.3.2 Related Systems*

Data grids, distributed databases, and peer-to-peer (P2P) networks are three distributed systems that have some characteristics in common with CDNs. These three systems have been described here in terms of requirements, functionalities, and characteristics. Table 1.1 presents the comparison between CDNs and these three related systems based on their unique characteristics/features.

#### 1.3.2.1 Data Grids

A data grid [43, 62] is a data intensive computing environment that provides services to the users in different locations to discover, transfer, and manipulate large datasets stored in distributed repositories. At the minimum, a data grid provides two basic functionalities: a high-performance, reliable data transfer mechanism, and a scalable replica discovery and management mechanism [22]. A data grid consists of computational and storage resources in different locations connected by high-speed networks. They are especially targeted to large scientific applications such as high energy physics experiments at the Large Hadron Collidor [37], astronomy projects – Virtual Observatories [59], and protein simulation – BioGrid [2] that require analyzing a huge amount of data. The data generated from an instrument, experiment, or a network of sensors is stored at a principle storage site and is transferred to other storage sites around the world on request through the data replication mechanism. Users query the local replica catalog to locate the datasets that they require. With proper rights and permissions, the required dataset is fetched from the local repository if it is present there; otherwise it is fetched from a remote repository. The data may be transmitted to a computational unit for processing. After processing, the results may be sent to a visualization facility, a shared repository, or to individual users' desktops. Data grids promote an environment for the users to analyze data, share the results with the collaborators, and maintain state information about the data seamlessly across organizational and regional boundaries. Resources in a data grid are heterogeneous and are spread over multiple administrative domains. Presence of large datasets, sharing of distributed data collections, having the same logical namespace, and restricted distribution of data can be considered as the unique set of characteristics for data grids. Data grids also contain some application specific characteristics. The overall goal of data grids is to bring together existing distributed resources to obtain performance gain through data distribution. Data grids are created by institutions who come together to share resources on some shared goal(s) by forming a Virtual Organization (VO). On the other hand, the main goal of CDNs is to perform caching of data to enable faster access by end users. Moreover, all the commercial CDNs are proprietary in nature – individual companies own and operate them.

**Table 1.1** Comparison between CDNs and related systems

| Features | CDNs | Data Grids | Distributed Databases | P2P Networks |
|---|---|---|---|---|
| Category | A collection of networked computers spanning the Internet | Data intensive computing environment | Locally organized collection of data distributed across multiple physical locations | Information retrieval network formed by ad-hoc aggregation of resources |
| Constitution | Distribution of cache servers to the edge of the Internet | Formation of a VO of participating institutions. | Federation or splitting of existing database(s) | Collaboration among peers |
| Main goal | Reducing Web latency during content delivery | Performance gain through data distribution by pre-staging, optimal source selection, and high speed data movement | Integration of existing databases and replication of database fragments in a transparent manner | File sharing among peers |
| Integrity | Integrity between caches | Integrity between data grid replicas | Integrity between multiple DBs | N/A |
| Consistency | Strong cache consistency between replicated content | Weak consistency between data grid replicas | Strong database consistency between distributed DBs | Weak consistency between cached content |
| Autonomy | None | Autonomous participants | Autonomous DDB sites | Autonomous peers |
| Operational activities | Content caching | Seamless analysis, collaboration, and maintenance of data across organizational and regional boundaries | Query processing, optimization, and management | Locating or caching content, encrypting, retrieving, decrypting, and verifying content |
| Administration | Individual companies. Proprietary in nature | Institutions who cooperate on some shared goals | Single authoritative entity | Self-interested end users/peers |

#### 1.3.2.2 Distributed Databases

A Distributed Database (DDB) [21, 45] is a logically organized collection of data distributed across multiple physical locations. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers. Each computer in a distributed database system is a node. A node in a distributed database system acts as a client, server, or both depending on the situation. Each site has a degree of autonomy, is capable of executing a local query, and participates in the execution of a global query. A distributed database can be formed by splitting a single database or by federating multiple existing databases. The distribution of such a system is transparent to the users as they interact with the system as a single logical system. The transactions in a distributed database are transparent and each transaction must maintain integrity across multiple databases. Distributed databases have evolved to serve the need of large organizations that need to replace existing centralized database systems, interconnect existing databases, and to add new databases as new organizational units are added. Applications provided by DDB include distributed transaction processing, distributed query optimization, and efficient management of resources. DDBs are dedicated to integrate existing diverse databases to provide a uniform, consisting interface for query processing with increased reliability and throughput. Integration of databases in DDBs is performed by a single organization. Like DDBs, the entire network in CDNs is managed by a single authoritative entity. However, CDNs differ from DDBs in the fact that CDN cache servers do not have the autonomic property as in DDB sites. Moreover, the purpose of CDNs is content caching, while DDBs are used for query processing, optimization, and management.

#### 1.3.2.3 P2P Networks

P2P networks [13, 44] are designed for the direct sharing of computer resources rather than requiring any intermediate and/or central authority. They are characterized as information retrieval networks that are formed by ad-hoc aggregation of resources to form a fully or partially decentralized system. Within a P2P system, each peer is autonomous and relies on other peers for resources, information, and forwarding requests. Ideally there is no central point of control in a P2P network. Therefore, the participating entities collaborate to perform tasks such as searching for other nodes, locating or caching content, routing requests, encrypting, retrieving, decrypting, and verifying content. P2P systems are more fault-tolerant and scalable than the conventional centralized system, as they have no single point of failure. An entity in a P2P network can join or leave anytime. P2P networks are more suited to the individual content providers who are not able to access or afford the common CDN. An example of such system is BitTorrent [33], which is a popular P2P file sharing application. Content and file sharing P2P networks are mainly focused on creating efficient strategies to locate particular files within a group of peers, to provide reliable transfers of such files in case of high volatility, and to

manage heavy traffic (i.e. flash crowds) caused by the demand for highly popular files. This is in contrast to CDNs where the main goal lies in respecting client's performance requirements rather than efficiently sharing file/content among peers. Moreover, CDNs differ from the P2P networks because the number of nodes joining and leaving the network per unit time is negligible in CDNs, whereas the rate is important in P2P networks.

## 1.4 Insights for CDNs

From the above discussion it is clear that a CDN is essentially aimed at content providers or customers who want to ensure QoS to the end users when accessing their Web content. The analysis of present day CDNs reveals that, at the minimum, a CDN focuses on the following business goals: scalability, security, reliability, responsiveness and performance.

### *1.4.1 Scalability*

Scalability refers to the ability of the system to expand in order to handle new and large amounts of data, users, and transactions without any significant decline in performance. To expand to a global scale, CDN providers need to invest time and costs in provisioning additional network connections and infrastructures. It includes provisioning resources dynamically to address flash crowds and varying traffic. A CDN should act as a shock absorber for traffic by automatically providing capacity-on-demand to meet the requirements of flash crowds. This capability allows a CDN to avoid costly over-provisioning of resources and to provide high performance to every user.

Within the structure of present day CDN business model, content providers pay the CDN providers to maximize the impact of their content. However, current trends reveal that the type of applications that will be supported by CDNs in future, will transform the current business model [53]. In future, the content providers as well as the end users will also pay to receive high quality content. In this context, scalability will be an issue to deliver high quality content, maintaining low operational costs.

### *1.4.2 Security*

One of the major concerns of a CDN is to provide potential security solutions for confidential and high-value content [19]. Security is the protection of content against unauthorized access and modification. Without proper security control, a CDN platform is subject to cyber fraud, Distributed Denial-of-Service (DDoS) attacks,

viruses, and other unwanted intrusions that can cripple business. A CDN aims at meeting the stringent requirements of physical, network, software, data, and procedural security. Once the security requirements are met, a CDN can eliminate the need for costly hardware and dedicated component to protect content and transactions. In accordance to the security issues, a CDN provider combat against any other potential risk concerns including DDoS attacks or other malicious activity that may interrupt business.

### *1.4.3 Reliability, Responsiveness, and Performance*

Reliability refers to when a service is available and what are the bounds on service outages that may be expected. A CDN provider can improve client access to specialized content through delivering it from multiple locations. For this a fault-tolerant network with appropriate load balancing mechanisms is to be implemented [42]. Responsiveness implies, while in the face of possible outages, how soon a service would start performing the normal course of operation. Performance of a CDN is typically characterized by the response time (i.e. latency) perceived by end users. Slow response time is the single greatest contributor to customers' abandoning Web sites and processes. The reliability and performance of a CDN is affected by the distributed content location and routing mechanism, as well as by data replication and caching strategies. Hence, a CDN employs caching and streaming to enhance performance especially for delivery of media content [57]. A CDN hosting a Web site also focuses on providing fast and reliable service since it reinforces the message that the company is reliable and customer-focused.

## 1.5 Existing CDNs: State of the Art

In this section, we provide the state of art in current CDN landscape. We also describe the different services and technologies of existing CDNs. First, we provide a brief description on commercial CDNs (e.g. Akamai, EdgeStream, Limelight Networks, and Mirror Image) which exist in the content distribution space. Then we present a snapshot on academic CDNs (e.g. CoDeeN, Coral, and Globule) which gives a picture of what the CDN technologies are at this moment.

### *1.5.1 Commercial CDNs*

Most or all of the operational CDNs are developed by commercial companies which are subject to consolidation over time due to acquisition and/or mergers. Hence, in the section, we focus on studying only those commercial CDNs that have been

**Table 1.2** Summary of the existing commercial CDNs

| CDN Name | Service Type | Coverage | Products/Solutions (If any) |
|---|---|---|---|
| **Akamai** www.akamai.com | Provides CDN service, including streaming | Covers 85% of the market. 25,000 servers in 900 networks in 69 countries. It handles 20% of total Internet traffic today | Edge Platform for handling static as well as dynamic content, Edge Control for managing applications, and Network Operations Control Center (NOCC) |
| **EdgeStream** www.edgestream.com | Provides disrupted video streaming applications over the public Internet | Provides video streaming over consumer cable or ADSL modem connections around the globe, even over paths that have 20 router hops between server and end user | EdgeStream video on-demand and IPTV Streaming software for video streaming |
| **Limelight Networks** www.limelightnetworks.com | Provides distributed on-demand and live delivery of video, music, games and download | Edge servers located in 72 locations around the world | Limelight ContentEdge for distributed content delivery via HTTP, Limelight MediaEdge Streaming for distributed video and music delivery via streaming, and Limelight Custom CDN for custom distributed delivery solutions |
| **Mirror Image** www.mirror-image.com | Provides content delivery, streaming media, Web computing and reporting services | Edge servers located in 22 countries | Global Content Caching, Extensible Rules Engine (XRE), Video On-Demand, and Live Webcasting |

in stable operation for a significant period of time. Table 1.2 shows a list of four commercial CDNs and presents a brief summary of each of them. An updated listing of most of the existing commercial CDNs can be found in the research directories of Davison [25] and Pathan [49].

The proprietary nature of commercial CDNs makes it difficult to reveal detailed information about the technical and business strategies used by them. However, in the presented state-of-the-art survey of commercial CDNs, we provide information to significant details. In this context, it is worth mentioning that many CDN-specific information such as fees charged by CDNs, existing customers of CDNs are ignored since they are highly likely to change quickly over time. Therefore, the information provided in this section is expected to be stable and up-to-date. However, for readers' understanding on how a CDN charges its customers (i.e. CDN pricing strategies); we refer to Chap. 8 of the book, which outlines the pricing policies used for CDN services.

#### 1.5.1.1 Akamai

Akamai technologies [1, 27] was founded in 1998 at Massachusetts, USA. It evolved out of an MIT research effort aimed at solving the flash crowd problem. Akamai is the market leader in providing content delivery services. It owns more than 25,000 servers over 900 networks in 69 countries [1]. Akamai's approach is based on the observation that serving Web content from a single location can present serious problems for site scalability, reliability and performance. Hence, a system is devised to serve requests from a variable number of cache servers at the network edge. Akamai servers deliver static (e.g. HTML pages, embedded images, executables, and PDF documents), dynamic content (e.g. animations, scripts, and DHTML), and streaming audio and video.

Akamai's infrastructure handles flash crowds by allocating more servers to sites experiencing high load, while serving all clients from nearby servers. The system directs client requests to the nearest available server likely to have the requested content. Akamai provides automatic network control through the mapping technique (i.e. the direction of request to content servers), which uses a dynamic, fault-tolerant DNS system. The mapping system resolves a hostname based on the service requested, user location, and network status. It also uses DNS for network load-balancing. Akamai name servers resolve hostnames to IP addresses by mapping requests to a server. Akamai agents communicate with certain border routers as peers; the mapping system uses BGP (Border Gateway Protocol) [56] information to determine network topology. The mapping system in Akamai combines the network topology information with live network statistics – such as traceroute data [39] – to provide a detailed, dynamic view of network structure, and quality measures for different mappings.

Akamai's DNS-based load balancing system continuously monitors the state of services and their servers and networks. To monitor the entire system's health end-to-end, Akamai uses agents that simulate the end user behavior by downloading Web objects and measuring their failure rates and download times. Akamai uses this information to monitor overall system performance and to automatically detect and suspend problematic data centers or servers. Each of the content servers frequently reports its load to a monitoring application, which aggregates and publishes

load reports to the local DNS server. That DNS server then determines which IP addresses (two or more) to return when resolving DNS names. If a certain server's load exceeds a certain threshold, the DNS server simultaneously assigns some of the server's allocated content to additional servers. If the server's load exceeds another threshold, the server's IP address is no longer available to clients. The server can thus shed a fraction of its load when it experiences moderate to high load. The monitoring system in Akamai also transmits data center load to the top-level DNS resolver to direct traffic away from overloaded data centers. In addition to load balancing, Akamai's monitoring system provides centralized reporting on content service for each customer and content server. This information is useful for network operational and diagnostic purposes.

Akamai delivers static and dynamic content over HTTP and HTTPS. Akamai content servers apply lifetime and other features (e.g. ability to serve secure content over HTTPS protocol, support alternate content, transfer encodings, and handle cookies) to the static content based on its type. Based on these attributes the edge server ensures the consistency of the content. On the other hand, Akamai handle dynamic content on the edge servers with the use of Edge Side Includes (ESI) [4] technology. The use of ESI enables the content providers to break their dynamic content into fragments with independent cacheability properties. These fragments can be maintained as separate objects in Akamai's edge servers and are dynamically assembled to a dynamic Web page in response to the end user requests.

Akamai supports Microsoft Windows Media, Real, and Apple's QuickTime format for delivering streaming services (live and on-demand media). A live stream is captured and encoded by the content provider and sent to the entry point server of a set of Akamai edge servers, which in turn serve content to the end users. In order to avoid all single points of failure, backups are maintained for the entry point server. Moreover, the entry point server sends data on multiple redundant paths to the edge servers through using information dispersal techniques.

More information on Akamai and its overlay routing, including its performance, availability benefits, different uses in live streaming, application, and IP acceleration can be found in Chap. 10 of this book.

#### 1.5.1.2 EdgeStream

EdgeStream [23] was founded in 2000 at California, USA. It is a provider of video streaming applications over the public Internet. It provides video on-demand and IPTV streaming software to enable transportation of high bit rate video over Internet. It uses HTTP streaming for content delivery. EdgeStream supports different compression formats for delivering content. It has developed Continuous Route Optimization Software (CROS), Internet Congestion Tunnel Through (ICTT) and Real Time Performance Monitoring Service (RPMS) technologies, which together assist to address the latency, packet loss, and congestion bottlenecks. Embedded applications in Consumer Electronics Devices, wireless handheld devices, IP set top boxes, and advanced digital TV's can use the EdgeStream software for video streaming.

EdgeStream platform is made of client and server software modules. The server software consists of Content Management and Online Reporting (CMOR) Server Software Module, EdgeStream Control Server Software Module, EdgeStream Database System Module, and EdgeStream Streaming Server Module. All server modules may be combined to run on a single server, or run separately. CMOR module manages accounts, content, and all other servers in the system. It also generates Web-based real time reports for viewing statistics and transactions from a SQL database. The control module provides necessary authority to obtain the content location information along with streaming delivery management and logging functions. The database module maintains logs for accounting and billing purpose. It uses the Microsoft SQL 2000 Standard or Enterprise server software. The streaming server module is designed for load balancing and for running on standard low cost server platforms. When running on a dual processor server, streaming capacity can excess 500 Mbps with terabyte storage capacity.

EdgeStream client software provides a plug-in interface to the Windows Media and Real players. It can also be used to measure the Internet connection quality on a second by second basis. The PC client software is available for standard Windows platform and it is a 600 KB download. The Firmware client is a 300 KB (or smaller) download and can be either embedded in Windows XP or used with Windows CE.

#### 1.5.1.3 Limelight Networks

Limelight Networks [30] was founded in 2001 at Tempe, Arizona, USA. Its content delivery services include HTTP/Web distribution of digital media files such as video, music, games, software and social media. It delivers content to media companies, including businesses operating in television, music, radio, newspaper, magazine, movie, video game, and software industries.

Content providers upload content either directly to the Limelight CDN's servers or to their own servers, which are connected directly to Limelight's network. Upon request from an end user, Limelight distributes that content to one or more Web server clusters which feed the specially configured servers at each content delivery location around the world. The content is then delivered directly to the end users either through ISPs or over the public Internet if appropriate. Like other commercial CDNs, it uses DNS redirection to reroute client requests to local clusters of machines, having built detailed maps of the Internet through a combination of BGP feeds and their own measurements, such as traceroutes from numerous vantage points.

Limelight Networks support Adobe Flash, MP3 audio, Microsoft Windows Media, Real, and Apple's QuickTime format for delivering on-demand streaming services. Limelight Networks proprietary software include Limelight ContentEdge for distributed content delivery via HTTP, Limelight MediaEdge Streaming for distributed video and music delivery via streaming, Limelight StorageEdge for storing customer's content library within Limelight's CDN architecture, and Limelight Custom CDN for custom distributed delivery solutions. Content providers using

Limelight's streaming services use Limelight User Exchange (LUX), which is a Web-based management and reporting console for tracking the end users' activity with real time reporting. All these software together assist in managing the content delivery system.

#### 1.5.1.4 Mirror Image

Mirror Image [40] was founded in 1999 at Massachusetts, USA. It is a provider of online content, application, streaming media, Web computing, reporting, and transaction delivery services to the end users. It follows a Concentrated "Superstore" architecture, where content is placed in large Web server clusters in central locations close to densely populated user regions. Mirror Image exploits a global Content Access Point (CAP) infrastructure on top of the Internet to provide content providers, service providers, and enterprises with a platform for content delivery.

When a user request for content comes from a Mirror Image provisioned Web site, it is automatically routed to a global load balancer on the CAP network. The load balance uses DNS routing to determine the CAP location with fastest response time. Upon reception of the request at the selected CAP location, the caches and then the core databases are checked for the requested content. If the content is found, it is delivered to the user. On cache miss, the CAP network automatically returns a redirection status code "302" to the origin server's URL. Then the requested content is delivered to the user from the origin server and the CAP network retrieves (or pull) the content from the origin server and stores it for future subsequent requests.

Mirror Image provides content delivery, streaming media, and Web computing solutions, including Global Content Caching solution to offload traffic spikes while serving static content; Digital Asset Download solution to manage the storage and download of digital content; Video On-Demand solution for streaming delivery of digital content; Extensible Rules Engine (XRE) to give customers control over the delivery process; and Webcasting solution to allow users to send "one-to-many" messages for training, marketing, and distance learning outlets.

### *1.5.2 Academic CDNs*

Unlike commercial CDNs, the use of P2P technologies is mostly common in academic CDNs. Thus, the content delivery follows a decentralized approach and request load is spread across all the participating hosts, and thus the system can handle node failures and sudden load surges. Academic CDNs built using P2P techniques are effective for static content only and therefore, are unable to handle dynamically generated content due to the uncachable nature of dynamic content. In this section, we present three representative academic CDNs, namely CoDeeN, Coral, and Globule. Table 1.3 provides a brief summary of these academic CDNs. Two other academic CDNs – FCAN (adaptive CDN for alleviating flash crowds) and COMODIN

**Table 1.3** Summary of the existing academic CDNs

| CDN Name | Description | Service Type | Implementation and Testing | Availability |
|---|---|---|---|---|
| **CoDeeN** www.codeen.cs. princeton. edu | CoDeeN is an academic testbed CDN built on top of PlanetLab | Provides caching of content and redirection of HTTP requests | Implemented in C/C++ and tested on Linux(2.4/2.6) and MacOS(10.2/10.3) | N/A |
| **Coral** www.coralcdn.org | Coral is a free P2P CDN. It is hosted on PlanetLab | Provides content replication in proportion to the content's popularity | Implemented in C++ and tested on Linux, OpenBSD, FreeBSD, and Mac OS X | No official release yet. Coral is a Free software, licensed under GPLv2 (GNU General Public License) |
| **Globule** www.globule.org | Globule is an open source collaborative CDN | Provides replication of content, monitoring of servers and redirecting client requests to available replicas | Implemented using PHP scripting, C/C++ and tested on Unix/Linux and Windows | Globule is open source, subject to a BSD-style license and the Apache software license for the packaged Apache HTTP server |

(streaming CDN for collaborative media streaming services), are presented respectively in Chap. 11 and Chap. 12 of this book.

#### 1.5.2.1 CoDeeN

CoDeeN [46, 64] is a P2P-based proxy server system developed at Princeton University, USA. It is an HTTP-based CDN, which gives *participating* users better performance to *most* Web sites. CoDeeN provides caching of Web content and redirection of HTTP requests. It is built on top of PlanetLab [9], consisting of a network of high performance proxy servers. CoDeeN nodes are deployed as "open" proxies in order to allow access from outside the hosting organization. Each CoDeeN node is capable of acting as a forward proxy, a reverse proxy, and a redirector. CoDeeN operates in the following way: (1) users set their internet caches to a nearby high bandwidth proxy that participates in the CoDeeN system; (2) the CoDeeN node acts as a forward proxy and tries to satisfy the request locally. On cache miss, the redirector logic built in the CoDeeN node determines where the request should be sent. For most requests the redirector take into account request locality, system load, reliability, and proximity information to forward the requests to other CoDeeN nodes, which act as a reverse proxy for the forwarded requests. Requests which are still not satisfied at this stage are sent to the origin server.

CoDeeN has the local monitoring ability that examines the service's primary resources, such as free file descriptors/sockets, CPU cycles, and DNS resolver service. It gathers information about the CoDeeN instance's state and its host environment. This information assists in assessing resource connection as well as external service availability. To monitor the health and status of the peers, each CoDeeN node employs two mechanisms – a lightweight UDP-based heartbeat and a "heavier" HTTP/TCP-level "fetch" helper [64]. In the first case, each proxy sends a heartbeat message once per second to one of its peers, which then responds (heartbeat acknowledgement or ACK) with piggybacked load information including peer's average load, system time, file descriptor availability, proxy and node uptimes, average hourly traffic, and DNS timing/failure statistics. By coupling the history of ACKs with their piggybacked local status information, each CoDeeN instance independently assesses the health of other nodes. In the later case, each CoDeeN node is employed with a toll to specify what fails when it can not retrieve a page within the allotted time. A history of failed fetches for each peer is maintained, which in combination with UDP-level heartbeats assists in determining if a node is viable for request redirection.

A number of projects are related to CoDeeN – CoBlitz (a scalable Web-based distribution system for large files), CoDeploy (an efficient synchronization tool for PlanetLab slices), CoDNS (a fast and reliable name lookup service), CoTop (a command line activity monitoring tool for PlanetLab), CoMon (a Web-based slice monitor that monitors most PlanetLab nodes), and CoTest (a login debugging tool).

A significant application service running on top of CoDeeN is CoBlitz [48]. It is a file transfer service which distributes large files without requiring any modifications

to standard Web servers and clients, since all the necessary support is located on CoDeeN itself. One of the motivations for building CoBlitz on top of CoDeeN is to ensure long duration caching so that client requested content can be served quickly even after demand for it drops. CoBlitz is publicly accessible which allows the clients to prepend the original URL with "`http://coblitz.codeen.org:3125`" and fetch it like any other URL. A customized DNS server maps the name coblitz.codeen.org to a nearby PlanetLab node. To deploy CoBlitz, the HTTP CDN, CoDeeN is made amenable to handling large files. This approach includes modifying large file handling to efficiently support them on CoDeeN and modifying its request-routing to enable more swarm-like behavior under heavy load. In CoBlitz, a large file is considered as a set of small files (chunks) that can be spread across the CDN. CoBlitz works if the chunks are fully cached, partially cached, or not at all cached, fetching any missing chunks from the origin as needed. Thus, while transferring large files over CoBlitz, no assumptions are made about the existence of the file on the peers.

#### 1.5.2.2 Coral

Coral [28] is a free P2P content distribution network. It was developed by the New York University's Secure Computer Systems group during their visit to Stanford University, USA. It is designed to mirror Web content and its goal is to give most *users* better performance to *participating* Web sites. It uses the bandwidth of volunteers to avoid flash crowd and to reduce the load on Web sites and other Web content providers in general. CoralCDN is deployed on PlanetLab, instead of third party volunteer systems. To use CoralCDN, a content publisher, end-host client, or someone posting a link to a high-traffic portal has to append "`.nyud.net:8090`" to the hostname in a URL. Clients are redirected to the nearby Coral Web caches transparently through DNS redirection. Coral Web caches cooperate to transfer data from nearby peers whenever possible, minimizing both the load on the origin Web server and the latency perceived by the user. CoralCDN is built on top of the Coral key-value indexing layer. It allows nodes to access nearby cached objects without redundantly querying more distant nodes. It also prevents the creation of hotspots in the indexing infrastructure, even under degenerate loads.

CoralCDN is comprised of three main parts: a network of cooperative HTTP proxies for handling client requests; a network of DNS nameservers for ".nyud.net" that map clients to nearby CoralCDN HTTP proxy; and an underlying indexing infrastructure and clustering machinery on which the first two applications rely.

Coral uses an indexing abstraction called Distributed Sloppy Hash Table (DSHT), which is a variant of Distributed Hash Tables (DHTs) for building key value indexes. DSHTs are designed for applications storing soft-state key-value pairs, where multiple values may be stored under the same key. A DSHT caches key-value pairs at nodes whose identifiers are increasingly close to the key being referenced, as an inverse function of load. It has a "sloppy" storage technique that leverages cross-layer interaction between the routing and storage layers.

The CoralHTTP proxy satisfies HTTP requests for Coralized URLs. To minimize the load on the origin servers, a CoralHTTP proxy fetch Web pages from other proxies whenever possible. Each proxy keeps a local cache to fulfill requests immediately. If a CoralHTTP proxy discovers the requested content in one or more other proxies, it establishes parallel TCP connections to them (multiple other proxies) and issues an HTTP request to the first proxy to which it successfully connects. Once the neighboring proxy begins to send valid content, all other established TCP connections are closed. When a client requests content from a non-resident URL, CoralHTTP proxy first attempts to locate a cached copy. If the Coral indexing layer does not provide any referral or any of its referrals return the requested content, CoralHTTP proxy fetches the content directly from the origin server. In the face of a flash crowd, all CoralHTTP proxies naturally form a kind of "multicast tree" for retrieving the Web page, instead of making simultaneous requests to the origin server and data flows from the proxy that initially fetch the content from the origin server to those arriving later. Such behavior in CoralCDN is provided by combining optimistic references and cut-through routing.

The CoralDNS server maps the IP addresses to the hostnames of Coralized URLs and returns it to CoralHTTP proxies. Coral's architecture is based on clusters of well-connected machines. Clusters are exposed in the interface to higher-level software, and in fact form a crucial part of the DNS redirection mechanism. In order to improve locality, when a DNS resolver contacts a nearby CoralDNS server instance, the CoralDNS server returns the proxies within an appropriate cluster and ensures that future DNS requests form this client does not leave the cluster. A CoralDNS server only returns the CoralHTTP proxy addresses which is has recently verified first hand in order to check a proxy's liveness status synchronously prior to replying to a DNS query.

#### 1.5.2.3 Globule

Globule [52] is an open-source collaborative CDN developed at the Vrije Universiteit in Amsterdam, the Netherlands. It aims to allow Web content providers to organize together and operate their own world-wide hosting platform. In particular, it is an overlay network composed of end user nodes that operate in a P2P fashion across a wide-area network, where participating members offer resources such as storage capacity, bandwidth, and processing power. It provides replication of content, monitoring of servers and redirection of client requests to available replicas.

In Globule, a site is defined as a collection of documents that belong to one specific user (the site's owner) and a server is a process running on a machine connected to a network, which executes an instance of the Globule software. Each site is composed of the origin, backup, replica, and redirector servers. The origin server(s) has the authority to contain all documents of the site and has the responsibility to distribute content among other involved servers. The backup servers maintain a full up-to-date copy of the hosted site. Other than backup servers, a number of replica servers can be used to host a site. While backup servers just maintain a copy, replica

servers aim to maximize performance based on the request load and QoS requirements. A replica server for a site is typically operated by a different user than the origin and a replica server typically contain a partial copy of the hosted site. One can view the replica server as a caching proxy which fetches the content from the origin server on a local cache miss. A redirector server is responsible for redirecting client requests to the optimal replica server for serving a given request. Redirectors in Globule can use either HTTP or DNS-based redirection. A redirector also implements a policy for client redirection. The default policy redirects clients to the closest replica in terms of estimated latency. Redirectors also monitor the availability of other servers to ensure effective redirection of requests. Depending on the role a server can perform as origin, replica, backup and/or redirector servers.

Globule takes inter-node latency as the proximity measure. This metric is used to optimally place replicas to the clients, and to redirect the clients to an appropriate replica server. Globule is implemented as a third-party module for the Apache HTTP Server that allows any given server to replicate its documents to other Globule servers. To replicate content, content providers only need to compile an extra module into their Apache server and edit a simple configuration file.

## 1.6 Visionary Thoughts for Practitioners

We envision the following technological evolutions to be realized in the coming years in CDN industry related research.

### *1.6.1 A Unified Content Network*

To make content transformations and processing and infrastructure service accessible by the user, vendors have implemented Content Service Networks (CSN) [38], which act as another network infrastructure layer built upon CDNs and provide next generation of CDN services. CSN appears to be a variation of the conventional CDN. Network resources provided by a CSN is used as a "service" distribution channel for value added service providers in order to make their applications as an infrastructure service. This logical separation between content and services under the "Content Delivery/Distribution" and "Content Services" domain, is undesirable considering the on-going trend in content networking. Hence, a unified content network, which supports the coordinated composition and delivery of content and services, is highly desirable.

### *1.6.2 Dynamic Content*

Dynamic content refers to the content which is generated on-demand using Web applications based on user requests. Such content generation is customized depending

on a given user profile and characteristics. A large amount of Web content is generated dynamically. Dynamic content includes scripts, animations, DHTML or XML pages that are generated on the fly based on user specification. The dynamic generation of Web pages can be performed with the use of scalable Web application hosting techniques such as edge computing [55], context-aware data caching [20, 58], data replication [58], and content blind data caching [58]. Instead of replicating the dynamic pages generated by a Web server, these techniques aim to replicate the means of generating pages over multiple edge servers [58]. Commercial CDN providers have their own proprietary solutions and application server platform to handle dynamic content. EdgeSuite content distribution from Akamai and IBM WebSphere edge services [5] are examples of systems to provide usage-based application and (dynamic) content delivery. In order to manage dynamic content, a CDN provider may use such scalable techniques to accelerate the dynamic generation of Web pages. The choice of the appropriate strategy may vary depending on the characteristics of Web applications.

### *1.6.3 Web Services*

Nowadays, a few commercial CDNs host Web services. For instance, Akamai has deployed .NET services on its network. Mirror Image has also developed an Application Delivery Network (ADN) that hosts both .NET and J2EE applications at its edge servers. Several studies [29, 60] have shown that the performance of Web services is relatively poor because of the requirements for processing and special hosting capability. Some solutions can be found in literature, which can be used to address the problem of effective replication of Web services to the CDN edge servers. Geng et al. [29] propose a sharable and tradable cache infrastructure between several ISPs and networks. This solution is characterized by a capacity provisioning network (CPN) for trading cache capacities. A CPN is operated by a trading hub rather than being operated by a particular CDN. Such a solution can be followed by a CDN to acquire (through trading) necessary capacity to meet the demand for Web service caching. Takase et al. [60] present caching using XML messages, improvements by caching event sequences of the XML parser. They also propose caching of application objects using Java serialization, reflection copy, and clone copy.

### *1.6.4 Service-Oriented Architecture*

Future trends in content networking domain are expected to allow services to be composed of other services by building on standard protocols and invocation mechanisms. Thus, content networks should be capable of exploiting an SOA. High-level transparency within SOA is required, which could have impact on all the constituent technologies. Content management in such an SOA-based CDN is expected to be

highly motivated by user preferences. Hence, a comprehensive model for managing the distributed contents and services in future CDN would be crucial to avail end user's preferences. To address this issue, contents can be personalized to meet specific user's (or a group of users) preferences. Like Web personalization [41], user preferences can be automatically learned from content request and usage data by using data mining techniques. Data mining over content network can exploit significant performance improvement through dealing with proper management of traffic, pricing and accounting/billing in SOA-based CDNs.

## 1.7 Future Research Directions

In this section, we provide a roadmap to the academic CDN researchers by exploring possibilities and research challenges that are expected to drive innovations within this domain.

### *1.7.1 Load Balancing and Content replication in Cooperative Domain*

The issue of effective replication and caching of content is critical to the success of traditional as well as cooperative arrangement of CDNs. The concept of caching "hot" content is not new, but in the context of a cooperative content delivery, there will be significant competing considerations. Future research should lead to the outcome of dynamic, scalable, and efficient replication mechanisms that cache content on demand with respect to the locality of requests, focusing on regions where specific content is needed most. Moreover, innovative solutions integrating replication and caching are expected in the management of dynamic and personalized content in the cooperative domain. Chapter 3 provides more information on such innovative content replication techniques. Detailed information on the integrated use of caching and replication as well as cache consistency mechanisms can be found in Chap. 4 and Chap. 5 of this book.

### *1.7.2 Deployment of Market Mechanisms*

An economic model can exploit the dynamism of the CDN market and makes the system more manageable through analyzing the emergent marketplace behavior. This also provides benefits to the CDNs to offer their resources and to open up many interesting new services. Deployment of the market mechanisms can be done based on an SOA. In addition, replication, resource sharing, and load balancing polices need to be guided by profit-driven utility functions that satisfy QoS requirements of end users. More information on economics-informed design of CDNs and pricing of CDNs can be found in Chap. 7 and Chap. 8 respectively.

### 1.7.3 An Adaptive CDN for Media Streaming

Hosting of on-demand media streaming service is challenging because of the enormous network and bandwidth required to simultaneously deliver large amount of content to end users. To avoid network congestion and to improve performance, P2P techniques can be used to build an adaptive CDN. In such a system, content storage and workload from streaming server, network, and storage resources are offloaded to the end users' workstations. The fundamental idea is to allow multiple subscriber peers to serve streams of the same video content simultaneously to a consuming peer rather than the traditional single-server-to-client streaming model, while allowing each peer to store only a small portion of the content. Such a solution for cost-effective media streaming using a P2P approach has been reported in the design of the Decentralized Media Streaming Infrastructure (DeMSI) [65]. Another work on open and adaptive streaming CDN through collaborative control on media streaming services is described in Chap. 12 of this book.

### 1.7.4 A Mobile Dynamic CDN

Mobile networks are becoming increasing popular for distributing information to a large number of highly dynamic users. In comparison to wired networks, mobile networks are distinguished by potentially much higher variability in demand due to user mobility. Content delivery techniques for mobile networks must take into account potentially very high spatial and temporal demand variations to dynamically reconfigure the system, and to minimize the total traffic over the network backbone. A model for mobile dynamic CDN should be designed to allow the access of accurate and up-to-date information and enterprise applications. Such a mobile dynamic CDN model for enterprise networks and related content management policies are presented by Aioffi et al. [12]. Example of a commercial mobile CDN provider is Ortiva Wireless [8], which delivers audio, video, and multimedia content to mobile users. More information on information dissemination in mobile CDNs can be found in Chap. 14 of this book.

### 1.7.5 Content Distribution Through Internetworking/Peering/Brokering

Present trends in content networks and content networking capabilities give rise to the interest in interconnecting content networks. High quality service could be achieved by permitting CDNs to cooperate and thereby providing a means for CDNs to redistribute content delivery between themselves. Such cooperation could reach to a large client population that one CDN cannot achieve otherwise. Therefore, future research will heavily focus on the innovation of technologies for

internetworking, brokering or peering arrangements between CDNs [18, 26, 50]. More information on CDN internetworking can be found in Chap. 16 of this book.

## 1.8 Conclusion

In this chapter, we present a state-of-the-art survey of the existing CDNs and give an insight into the underlying technologies that are currently in use in the content-distribution space. After analyzing the ongoing content networking trend, we can anticipate the integrated uses of existing emerging as well as stable technologies (e.g. agent, P2P, grid, data mining) to augment the effectiveness and boost the efficiency of future CDN infrastructures. We also perceive that there is a possible shift change in the CDN industry as CDN internetworking, adaptive CDNs, mobile CDNs, and to the full, community-based CDNs are evolving. Therefore, this chapter can be used as a basis to provide an in-depth analysis and complete understanding of the current and future trends in the content distribution landscape.

## References

1. Akamai Technologies, 2007. www.akamai.com
2. BioGrid Project, Japan, 2005. http://www.biogrid.jp
3. Broadband Service Forum, 2007. http://broadbandservicesforum.org
4. ESI Developer Resources, 2007. http://www.akamai.com/html/support/esi.html
5. IBM WebSphere Application Server, 2007. http://www-306.ibm.com/software/webservers/appserv/was/
6. ICAP Forum, 2007. http://www.i-cap.org/
7. Internet Streaming Media Alliance, 2007. http://www.isma.tv/
8. Ortiva Wireless, 2007. http://www.ortivawireless.com/
9. PlanetLab Consortium, 2007. http://www.planet-lab.org/
10. Wikipedia. September 11, 2001 attacks. http://en.wikipedia.org/wiki/September_11,_2001_attack
11. Adler, S. The slashdot effect: an analysis of three Internet publications. *Linux Gazette Issue*, 38, 1999.
12. Aioffi, W. M., Mateus, G. R., de Almeida, J. M., and Loureiro, A. A. F. Dynamic content distribution for mobile enterprise networks. *IEEE Journal on Selected Areas in Communications*, 23(10), pp. 2022–2031, 2005.
13. Androutsellis-Theotokis, S. and Spinellis, D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4), ACM Press, NY, USA, pp. 335–371, 2004.
14. Arlitt, M. and Jin, T. A workload characterization study of 1998 world cup Web site. *IEEE Network*, pp. 30–37, 2000.
15. Barbir, A., Batuner, O., Beck, A., Chan, T., and Orman, H. Policy, authorization, and enforcement requirements of the open pluggable edge services (OPES). Internet Engineering Task Force RFC 3838, 2004. www.ietf.org/rfc/rfc3838.txt
16. Barbir, A., Penno, R., Chen, R., Hofmann, H., and Orman, H. An architecture for open pluggable edge services (OPES). Internet Engineering Task Force RFC 3835, 2004. www.ietf.org/rfc/rfc3835.txt

17. Bartolini, N., Casalicchio, E., and Tucci, S. A walk through content delivery networks. In *Proc. of the 11th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, LNCS Vol. 2965/2004, pp. 1–25, April 2004.
18. Biliris, A., Cranor, C., Douglis, F., Rabinovich, M., Sibal, S., Spatscheck, O., and Sturm, W. CDN brokering. *Computer Communications*, 25(4), pp. 393–402, 2002.
19. Brussee, R., Eertink, H., Huijsen, W., Hulsebosch, B., Rougoor, M., Teeuw, W., Wibbels, M., and Zandbelt, H. Content distribution network state of the art," Telematica Instituut, 2001.
20. Buchholz, T., Hochstatter, I., and Linnhoff-Popien, C. A profit maximizing distribution strategy for context-aware services. In *Proc. of 2nd International Workshop on Mobile Commerce and Services (WMCS'05)*, pp. 144–153, 2005.
21. Ceri, S. and Pelagatti, G. *Distributed Databases: Principles and Systems*, McGraw-Hill, NY, 1984.
22. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23, pp. 187–200, 2001.
23. Chung, R. Network latency and its effect on video streaming. EdgeStream, 2004. www.edgestream.com
24. Cooper, I., Melve, I., and Tomlinson, G. Internet Web replication and caching taxonomy. Internet Engineering Task Force RFC 3040, 2001.
25. Davison, B. D. Web caching and content delivery resources. http://www.web-caching.com, 2007.
26. Day, M., Cain, B., Tomlinson, G., and Rzewski, P. A model for content internetworking (CDI). Internet Engineering Task Force RFC 3466, 2003.
27. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing*, pp. 50–58, 2002.
28. Freedman, M. J., Freudenthal, E., and Mazières, D. Democratizing content publication with Coral. In *Proc. of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, San Francisco, CA, USA, 2004.
29. Geng, X., Gopal, R. D., Ramesh, R., and Whinston, A. B. Scaling web services with capacity provision networks. *IEEE Computer*, 36(11), pp. 64–72, 2003.
30. Gordon, M. The Internet streaming media boom: a powerful trend that represents fundamental change. Limelight Networks, 2007. www.limelightnetworks.com
31. Hofmann, M. and Beaumont, L. R. *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129–134, 2005.
32. International Standards Organization (ISO), Open systems interconnection–basic reference model. ISO 7498, 1989.
33. Izal, M., Urvoy-Keller, G., Biersack, E. W., Felber, P., Hamra, A. A., and Garces-Erice, L. Dissecting bittorrent: five months in a torrent's lifetime. In *Proc. of 5th Annual Passive and Active Measurement Workshop (PAM'2004)*, Antibes Juan-Les-Pins, France, 2004.
34. Jung, J., Krishnamurthy, B., and Rabinovich, M. Flash crowds and denial of service attacks: characterization and implications for CDNs and Web sites. In *Proc. of the International World Wide Web Conference*, pp. 252–262, 2002.
35. Kangasharju, J., Roberts, J., and Ross, K. W. Object replication strategies in content distribution networks. *Computer Communications*, 25(4), pp. 367–383, 2002.
36. Lazar, I. and Terrill, W. Exploring content delivery networking. *IT Professional*, 3(4), pp. 47–49, 2001.
37. Lebrun, P. The large hadron collider, a megascience project. In *Proc. of the 38th INFN Eloisatron Project Workshop on Superconducting Materials for High Energy Colliders*, Erice, Italy, 1999.
38. Ma, W. Y., Shen, B., and Brassil, J. T. Content services network: architecture and protocols. In *Proc. of 6th International Workshop on Web Caching and Content Distribution (IWCW6)*, 2001.
39. Malkin, G. Traceroute using an IP option. Internet Engineering Task Force RFC 1393, 1993.

40. Mirror Image Internet. Content Delivery and the Mirror Image Adaptive CAP Network, 2007. www.mirror-image.com
41. Mobasher, B., Cooley, R., and Srivastava, J. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8), pp. 142–151, 2000.
42. Molina, B., Palau, C. E., and Esteve, M. Modeling content delivery networks and their performance. *Computer Communications*, 27(15), pp. 1401–1411, 2004.
43. Moore, R., Prince, T. A., and Ellisman, M. Data intensive computing and digital libraries. *Communications of the ACM*, 41(11), ACM Press, NY, USA, pp. 56–62, 1998.
44. Oram, A. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, 2001.
45. Ozsu, M. T. and Valduriez, P. *Principles of Distributed Database Systems*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1999.
46. Pai, V. S., Wang, L., Park, K. S., Pang, R., and Peterson, L. The dark side of the web: an open proxy's view. In *Proc. of the Second Workshop on Hot Topics in Networking (HotNets-II)*, Cambridge, MA, USA, 2003.
47. Pallis, G. and Vakali, A. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1), ACM Press, NY, USA, pp. 101–106, 2006.
48. Park, K. S. and Pai, V. S. Scale and performance in the CoBlitz large-file distribution service. In *Proc. of the 3rd Symposium on Networked Systems Design and Implementation (NSDI 2006)*, San Jose, CA, USA, 2006.
49. Pathan, M. Content delivery networks (CDNs) research directory, 2007. http://www.gridbus.org/cdn/CDNs.html
50. Pathan, M., Broberg, J., Bubendorfer, K., Kim, K. H., and Buyya, R. An Architecture for Virtual Organization (VO)-Based Effective Peering of Content Delivery Networks, UPGRADE-CN'07. In *Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, CA, USA, 2007.
51. Peng, G. CDN: Content distribution network. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003. http://citeseer.ist.psu.edu/peng03cdn.html
52. Pierre, G. and van Steen, M. Globule: a collaborative content delivery network. *IEEE Communications*, 44(8), 2006.
53. Plagemann, T., Goebel, V., Mauthe, A., Mathy, L., Turletti, T., and Urvoy-Keller, G. From content distribution to content networks – issues and challenges. *Computer Communications*, 29(5), pp. 551–562, 2006.
54. Rabinovich, M. and Spatscheck, O. *Web Caching and Replication*, Addison Wesley, USA, 2002.
55. Rabinovich, M., Xiao, Z., Douglis, F., and Kalmanek, C. Moving edge side includes to the real edge – the clients. In *Proc. of USENIX Symposium on Internet Technologies and Systems*, Sealttle, Washington, USA, 2003.
56. Rekhter, Y. and Li, T. A border gateway protocol 4. Internet Engineering Task Force RFC 1771, 1995.
57. Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. An analysis of Internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36, pp. 315–328, 2002.
58. Sivasubramanian, S., Pierre, G., Van Steen, M., and Alonso, G. Analysis of caching and replication strategies for Web applications. *IEEE Internet Computing*, 11(1), pp. 60–66, 2007.
59. Szalay, A. and Gray, J. The world-wide telescope. *Science* 293(5537), pp. 2037–2040, 2001.
60. Takase, T. and Tatsubori, M. Efficient Web services response caching by selecting optimal data representation. In *Proc. of 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, pp. 188–197, 2004.
61. Vakali, A. and Pallis, G. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6), IEEE Computer Society, pp. 68–74, 2003.
62. Venugopal, S., Buyya, R., and Ramamohanarao, K. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, 38(1), ACM Press, NY, USA, 2006.

63. Verma, D. C. *Content Distribution Networks: An Engineering Approach*, John Wiley & Sons, Inc., New York, USA, 2002.
64. Wang, L., Park, K. S., Pang, R., Pai, V. S., and Peterson, L. Reliability and security in CoDeeN content distribution network. In *Proc. of the USENIX 2004 Annual Technical Conference*, Boston, MA, USA, 2004.
65. Yim, A. and Buyya, R. Decentralized media streaming infrastructure (DeMSI): an adaptive and high-performance peer-to-peer content delivery network. *Journal of Systems Architecture*, 52(12), Elsevier, The Netherlands, pp. 737–772, 2006.

# Chapter 2
# A Taxonomy of CDNs

Mukaddim Pathan and Rajkumar Buyya

## 2.1 Introduction

Content Delivery Networks (CDNs) [79, 97] have received considerable research attention in the recent past. A few studies have investigated CDNs to categorize and analyze them, and to explore the uniqueness, weaknesses, opportunities, and future directions in this field. Peng presents an overview of CDNs [75]. His work describes the critical issues involved in designing and implementing an effective CDN, and surveys the approaches proposed in literature to address these problems. Vakali et al. [95] present a survey of CDN architecture and popular CDN service providers. The survey is focused on understanding the CDN framework and its usefulness. They identify the characteristics and current practices in the content networking domain, and present an evolutionary pathway for CDNs, in order to exploit the current content networking trends. Dilley et al. [29] provide an insight into the overall system architecture of the leading CDN, Akamai [1]. They provide an overview of the existing content delivery approaches and describe Akamai's network infrastructure and its operations in detail. They also point out the technical challenges that are to be faced while constructing a global CDN like Akamai. Saroiu et al. [84] examine content delivery from the point of view of four content delivery systems: Hypertext Transfer Protocol (HTTP) Web traffic, the Akamai CDN, Gnutella [8, 25], and KaZaa [62, 66] peer-to-peer file sharing systems. They also present significant implications for large organizations, service providers, network infrastructure providers, and general content delivery providers. Kung et al. [60] describe a taxonomy for content networks and introduce a new class of content networks that perform "semantic aggregation and content-sensitive placement" of content. They classify content networks based on their attributes in two dimensions: content aggregation and content placement. Sivasubramanian et al. [89] identify the issues

Mukaddim Pathan
GRIDS Lab, Department of CSSE, The University of Melbourne, Australia,
e-mail: apathan@csse.unimelb.edu.au

Rajkumar Buyya
GRIDS Lab, Department of CSSE, The University of Melbourne, Australia,
e-mail: raj@csse.unimelb.edu.au

R. Buyya et al. (eds.), *Content Delivery Networks*,
© Springer-Verlag Berlin Heidelberg 2008

for building a Web replica hosting system. Since caching infrastructure is a major building block of a CDN (e.g. Akamai) and content delivery is initiated from the origin server, they consider CDNs as replica hosting systems. In this context, they propose an architectural framework, review related research work, and categorize them. A survey of peer-to-peer (P2P) content distribution technologies [11] studies current P2P systems and categorize them by identifying their non-functional properties such as security, anonymity, fairness, increased scalability, and performance, as well as resource management, and organization capabilities. Through this study the authors make useful insights for the influence of the system design on these properties. Cardellini et al. [20] study the state of the art of Web system architectures that consists of multiple server nodes distributed on a local area. They provide a taxonomy of these architectures, and analyze routing mechanisms and dispatching algorithms for them. They also present future research directions in this context.

### *2.1.1 Motivations and Scope*

As mentioned above, there exist a wide range of work covering different aspects of CDNs such as content distribution, replication, caching, and Web server placement. However, none of them attempts to perform a complete categorization of CDNs by considering the functional and non-functional aspects. The first aspects include technology usage and operations of a CDN, whereas the latter focus on CDN characteristics such as organization, management, and performance issues. Our approach of considering both functional and non-functional aspects of CDNs assists in examining the way in which the characteristics of a CDN are reflected in and affected by the architectural design decision followed by the given CDN. Therefore, our aim is to develop a comprehensive taxonomy of CDNs that identifies and categorizes numerous solutions and techniques related to various design dynamics.

The taxonomy presented in this chapter is built around the core issues for building a CDN system. In particular, we identify the following key issues/aspects that pose challenges in the development of a CDN:

- *What is required for a harmonious CDN composition?* It includes decisions based on different CDN organization, node interactions, relationships, and content/service types.
- *How to perform effective content distribution and management?* It includes the right choice of content selection, surrogate placement, content outsourcing, and cache organization methodologies.
- *How to route client requests to appropriate CDN node?* It refers to the usage of dynamic, scalable, and efficient routing techniques.
- *How to measure a CDN's performance?* It refers to the ability to predict, monitor, and ensure the end-to-end performance of a CDN.

A full-fledged CDN system design seeks to address additional issues to make the system robust, fault tolerant (with the ability to detect wide-area failures), secure,

and capable of wide-area application hosting. In this context, the issues to be addressed are:

- *How to handle wide-area failures in a CDN?* It involves the use of proper tools and systems for failure detection.
- *How to ensure security in a wide-area CDN system?* It refers to the solutions to counter distributed security threats.
- *How to achieve wide-area application hosting?* It seeks to develop proper techniques to enable CDNs to perform application hosting.

Each of the above issues aspects is an independent research area itself and many solutions and techniques can be found in literature and in practice. While realizing proper solution for the additional issues is obvious for a CDN development, the taxonomy presented in this chapter concentrates only on the first four core issues. However, we present the ideas in the context of the additional issues and also provide pointers to some recent related research work. Thus, the readers can get sufficient materials to comprehend respective issues to dive directly into their interested topic.

### *2.1.2 Contributions and Organization*

The key contributions of this chapter are twofold:

- A taxonomy of CDNs with a complete coverage of this field to provide a comprehensive account of applications, features, and implementation techniques. The main aim of the taxonomy, therefore, is to explore the functional and non-functional features of CDNs and to provide a basis for categorizing the related solutions and techniques in this area.
- Map the taxonomy to a few representative CDNs to demonstrate its applicability to categorize and analyze the present-day CDNs. Such a mapping helps to perform "gap" analysis in this domain. It also assists to interpret the related essential concepts of this area and validates the accuracy of the taxonomy.

The remainder of this chapter is structured as follows: we start by presenting the taxonomy of CDNs in Sect. 2.2. In the next section, we map the taxonomy to the representative CDN systems, along with the insights perceived from this mapping. Thus, we prove the validity and applicability of the taxonomy. We discuss the additional issues in CDN development in Sect. 2.4 by highlighting research work in failure handling, security, and application hosting. Finally, we summarize and conclude the chapter in Sect. 2.5.

## 2.2 Taxonomy

In this section, we present a taxonomy of CDNs based on four key issues as shown in Fig. 2.1. They are – *CDN composition*, *content distribution and management*, *request-routing*, and *performance measurement*.

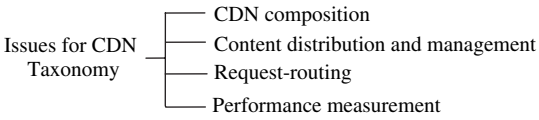



**Fig. 2.1** Issues for CDN taxonomy

The first issue covers several aspects of CDNs related to organization and formation. This classifies the CDNs with respect to their structural attributes. The next issue pertains to the content distribution mechanisms in the CDNs. It describes the content distribution and management approaches of CDNs in terms of surrogate placement, content selection and delivery, content outsourcing, and organization of caches/replicas. Request-routing techniques in the existing CDNs are described as the next issue. Finally, the last issue deals with the performance measurement methodologies of CDNs.

### *2.2.1 CDN Composition*

The analysis of the structural attributes of a CDN reveals that CDN infrastructural components are closely related to each other. Moreover, the structure of a CDN varies depending on the content/services it provides to its users. Within the structure of a CDN, a set of surrogates is used to build the content-delivery component, some combinations of relationships and mechanisms are used for redirecting client requests to a surrogate and interaction protocols are used for communications between CDN elements.

Figure 2.2 shows a taxonomy based on the various structural characteristics of CDNs. These characteristics are central to the composition of a CDN and they address the organization, types of servers used, relationships, and interactions among CDN components, as well as the different content and services provided.

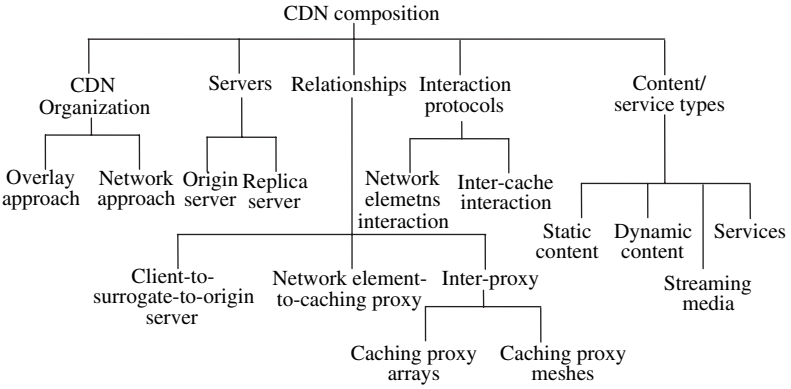



**Fig. 2.2** CDN composition taxonomy

#### 2.2.1.1 CDN Organization

There are two general approaches for building CDNs: *overlay* and *network* approach [61]. In the overlay approach, application-specific servers and caches at several places in the network handle the distribution of specific content types (e.g. Web content, streaming media, and real time video). Other than providing the basic network connectivity and guaranteed QoS for specific request/traffic, the core network components such as routers and switches play no active role in content delivery. Most of the commercial CDN providers such as Akamai and Limelight Networks follow the overlay approach for CDN organization. These CDN providers replicate content to cache servers worldwide. When content requests are received from the end users, they are redirected to the nearest CDN server, thus improving Web site response time. As the CDN providers need not to control the underlying network elements, the management is simplified in an overlay approach and it opens opportunities for new services.

In the network approach, the network components including routers and switches are equipped with code for identifying specific application types and for forwarding the requests based on predefined policies. Examples of this approach include devices that redirect content requests to local caches or switch traffic (coming to data centers) to specific servers, optimized to serve specific content types. Some CDNs (e.g. Akamai, Mirror Image) use both network and overlay approaches for CDN organization. In such case, a network element (e.g. switch) can act at the front end of a server farm and redirects the content request to a nearby application-specific surrogate server.

#### 2.2.1.2 Servers

The servers used by a CDN are of two types – *origin* and *replica* servers. The server where the definitive version of the content resides is called origin server. It is updated by the content provider. On the other hand, a replica server stores a copy of the content but may act as an authoritative reference for client responses. The origin server communicates with the distributed replica servers to update the content stored in it. A replica server in a CDN may serve as a media server, Web server or as a cache server. A media server serves any digital and encoded content. It consists of media server software. Based on client requests, a media server responds to the query with the specific video or audio clip. A Web server contains the links to the streaming media as well as other Web-based content that a CDN wants to handle. A cache server makes copies (i.e. caches) of content at the edge of the network in order to bypass the need of accessing origin server to satisfy every content request.

#### 2.2.1.3 Relationships

The complex distributed architecture of a CDN exhibits different relationships between its constituent components. The graphical representations of these

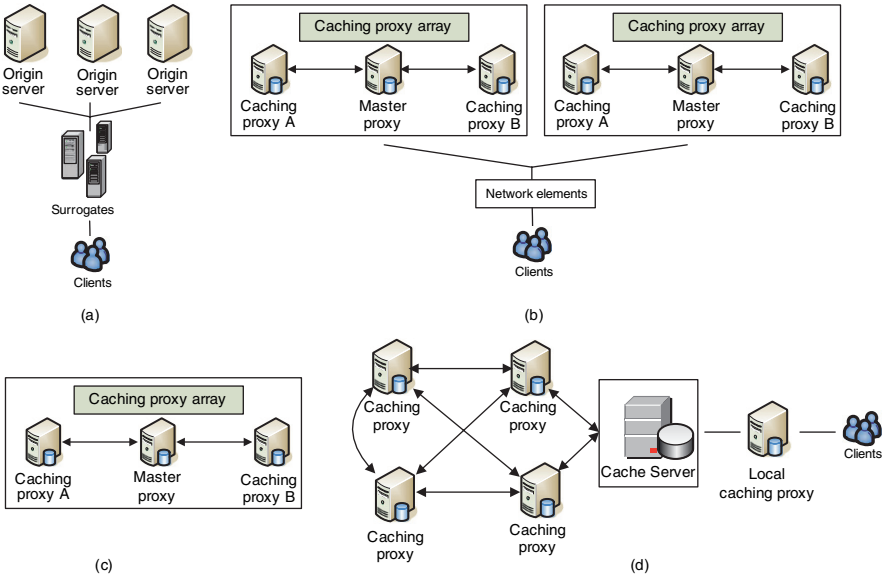



**Fig. 2.3** Relationships: (a) Client-to-surrogate-to-origin server; (b) Network element-to-caching proxy; (c) Caching proxy arrays; (d) Caching proxy meshes

relationships are shown in Fig. 2.3. These relationships involve components such as clients, surrogates, origin server, proxy caches, and other network elements. These components communicate to replicate and cache content within a CDN. Replication involves creating and maintaining duplicate copy of a given content on different computer systems. It typically involves "pushing" content from the origin server to the replica servers [17]. On the other hand, caching involves storing cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests [26, 27, 99].

In a CDN environment, the basic relationship for content delivery is among the client, surrogates and origin servers. A client may communicate with surrogate server(s) for requests intended for one or more origin servers. Where a surrogate is not used, the client communicates directly with the origin server. The communication between a user and surrogate takes place in a transparent manner, as if the communication is with the intended origin server. The surrogate serves client requests from its local cache or acts as a gateway to the origin server. The relationship among client, surrogates, and the origin server is shown in Fig. 2.3(a).

As discussed earlier, CDNs can be formed using a network approach, where logic is deployed in the network elements (e.g. router, switch) to forward traffic to caching servers/proxies that are capable of serving client requests. The relationship in this case is among the client, network element, and caching servers/proxies (or proxy arrays), which is shown in Fig. 2.3(b). Other than these relationships, caching proxies within a CDN may communicate with each other. A proxy cache is an application-layer network service for caching Web objects. Proxy caches can be

simultaneously accessed and shared by many users. A key distinction between the CDN proxy caches and ISP-operated caches is that the former serve content only for certain content provider, namely CDN customers, while the latter cache content from all Web sites [41].

Based on inter-proxy communication [26], caching proxies can be arranged in such a way that proxy arrays (Fig. 2.3(c)) and proxy meshes (Fig. 2.3(d)) are formed. A caching proxy array is a tightly-coupled arrangement of caching proxies. In a caching proxy array, an authoritative proxy acts as a master to communicate with other caching proxies. A user agent can have relationship with such an array of proxies. A caching proxy mesh is a loosely-coupled arrangement of caching proxies. Unlike the caching proxy arrays, proxy meshes are created when the caching proxies have one-to-one relationship with other proxies. Within a caching proxy mesh, communication can happen at the same level between peers, and with one or more parents [26]. A cache server acts as a gateway to such a proxy mesh and forwards client requests coming from client's local proxy.

#### 2.2.1.4 Interaction Protocols

Based on the communication relationships described earlier, we can identify the interaction protocols that are used for interaction among CDN components. Such interactions can be broadly classified into two types: *interaction between network elements* and *interaction between caches*. Figure 2.4 shows different protocols that are used in a CDN for interaction among CDN elements. Examples of protocols for network element interaction are *Network Element Control Protocol (NECP)* and *Web Cache Control Protocol*. On the other hand, *Cache Array Routing Protocol (CARP)*, *Internet Cache Protocol (ICP)*, *Hypertext Caching protocol (HTCP)*, and *Cache Digest* are the examples of inter-cache interaction protocols.

The Network Element Control Protocol (NECP) [24] is a lightweight protocol for signaling between servers and the network elements that forward traffic to them. The network elements consist of a range of devices, including content-aware switches and load-balancing routers. NECP allows network elements to perform load balancing across a farm of servers and redirection to interception proxies. However, it does not dictate any specific load balancing policy. Rather, this protocol provides methods for network elements to learn about server capabilities, availability and hints as

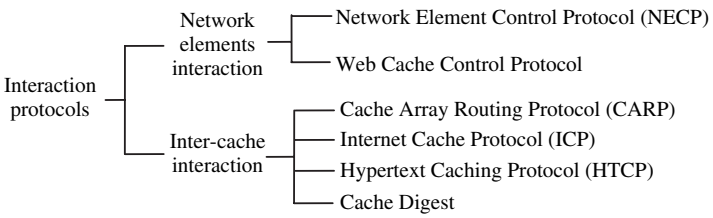



**Fig. 2.4** Interaction protocols

to which flows can and cannot be served. Hence, network elements gather necessary information to make load balancing decisions. Thus, it avoids the use of proprietary and mutually incompatible protocols for this purpose. NECP is intended for use in a wide variety of server applications, including for origin servers, proxies, and interception proxies. It uses the Transport Control Protocol (TCP). When a server is initialized, it establishes a TCP connection to the network elements using a well-known port number. Messages can then be sent bi-directionally between the server and network elements. Most messages consist of a request followed by a reply or acknowledgement. Receiving a positive acknowledgement implies the recording of some state in a peer. This state can be assumed to remain in that peer until it expires or the peer crashes. In other words, this protocol uses a "hard state" model. Application level KEEPALIVE messages are used to detect a crashed peer in such communications. When a node detects that its peer has been crashed, it assumes that all the states in that peer need to be reinstalled after the peer is revived.

The Web Cache Control Protocol (WCCP) [24] specifies interaction between one or more routers and one or more Web-caches. It runs between a router functioning as a redirecting network element and interception proxies. The purpose of such interaction is to establish and maintain the transparent redirection of selected types of traffic flow through a group of routers. The selected traffic is redirected to a group of Web-caches in order to increase resource utilization and to minimize response time. WCCP allows one or more proxies to register with a single router to receive redirected traffic. This traffic includes user requests to view pages and graphics on World Wide Web (WWW) servers, whether internal or external to the network, and the replies to those requests. This protocol allows one of the proxies, the designated proxy, to dictate to the router how redirected traffic is distributed across the caching proxy array. WCCP provides the means to negotiate the specific method used to distribute load among Web caches. It also provides methods to transport traffic between router and cache.

The Cache Array Routing Protocol (CARP) [96] is a distributed caching protocol based on a known list of loosely coupled proxy servers and a hash function for dividing URL space among those proxies. An HTTP client implementing CARP can route requests to any member of the Proxy Array. The proxy array membership table is defined as a plain ASCII text file retrieved from an Array Configuration URL. The hash function and the routing algorithm of CARP take a member proxy defined in the proxy array membership table, and make an on-the-fly determination about the proxy array member which should be the proper container for a cached version of a resource pointed to by a URL. Since requests are sorted through the proxies, duplication of cache content is eliminated and global cache hit rates are improved. Downstream agents can then access a cached resource by forwarding the proxied HTTP request for the resource to the appropriate proxy array member.

The Internet Cache Protocol (ICP) [101] is a lightweight message format used for inter-cache communication. Caches exchange ICP queries and replies to gather information to use in selecting the most appropriate location in order to retrieve an object. Other than functioning as an object location protocol, ICP messages can also be used for cache selection. ICP is a widely deployed protocol. Although, Web

caches use HTTP for the transfer of object data, most of the caching proxy implementations support it in some form. It is used in a caching proxy mesh to locate specific Web objects in neighboring caches. One cache sends an ICP query to its neighbors and the neighbors respond with an ICP reply indicating a "HIT" or a "MISS". Failure to receive a reply from the neighbors within a short period of time implies that the network path is either congested or broken. Usually, ICP is implemented on top of User Datagram Protocol (UDP) in order to provide important features to Web caching applications. Since UDP is an unreliable and connectionless network transport protocol, an estimate of network congestion and availability may be calculated by ICP loss. This sort of loss measurement together with the round-trip-time provides a way to load balancing among caches.

The Hyper Text Caching Protocol (HTCP) [98] is a protocol for discovering HTTP caches, cached data, managing sets of HTTP caches and monitoring cache activity. HTCP is compatible with HTTP 1.0. This is in contrast with ICP, which was designed for HTTP 0.9. HTCP also expands the domain of cache management to include monitoring a remote cache's additions and deletions, requesting immediate deletions, and sending hints about Web objects such as the third party locations of cacheable objects or the measured uncacheability or unavailability of Web objects. HTCP messages may be sent over UDP or TCP. HTCP agents must not be isolated from network failure and delays. An HTCP agent should be prepared to act in useful ways in the absence of response or in case of lost or damaged responses.

Cache Digest [42] is an exchange protocol and data format. It provides a solution to the problems of response time and congestion associated with other inter-cache communication protocols such as ICP and HTCP. They support peering between cache servers without a request-response exchange taking place. Instead, other servers who peer with it fetch a summary of the content of the server (i.e. the Digest). When using Cache Digest it is possible to accurately determine whether a particular server caches a given URL. It is currently performed via HTTP. A peer answering a request for its digest will specify an expiry time for that digest by using the HTTP Expires header. The requesting cache thus knows when it should request a fresh copy of that peer's digest. In addition to HTTP, Cache Digest could be exchanged via FTP. Although the main use of Cache Digest is to share summaries of which URLs are cached by a given server, it can be extended to cover other data sources. Cache Digest can be a very powerful mechanism to eliminate redundancy and making better use of Internet server and bandwidth resources.

#### 2.2.1.5 Content/Service Types

CDN providers host third-party content for fast delivery of any digital content, including – *static content*, *dynamic content*, *streaming media* (e.g. audio, real time video), and different *content services* (e.g. directory service, e-commerce service, and file transfer service). The sources of content are large enterprises, Web

service providers, media companies, and news broadcasters. Variation in content and services delivered requires a CDN to adopt application-specific characteristics, architectures, and technologies. Due to this reason, some of the CDNs are dedicated for delivering particular content and/or services. Here, we analyze the characteristics of the content/service types to reveal their heterogeneous nature.

Static content refers to content for which the frequency of change is low. It does not change depending on user requests. It includes static HTML pages, embedded images, executables, PDF documents, software patches, audio and/or video files. All CDN providers support this type of content delivery. This type of content can be cached easily and their freshness can be maintained using traditional caching technologies.

Dynamic content refers to the content that is personalized for the user or created on-demand by the execution of some application process. It changes frequently depending on user requests. It includes animations, scripts, and DHTML. Due to the frequently changing nature of the dynamic content, usually it is considered as uncachable.

Streaming media can be live or on-demand. Live media delivery is used for live events such as sports, concerts, channel, and/or news broadcast. In this case, content is delivered "instantly" from the encoder to the media server, and then onto the media client. In case of on-demand delivery, the content is encoded and then is stored as streaming media files in the media servers. The content is available upon requests from the media clients. On-demand media content can include audio and/or video on-demand, movie files and music clips. Streaming servers are adopted with specialized protocols for delivery of content across the IP network.

A CDN can offer its network resources to be used as a service distribution channel and thus allows the value-added services providers to make their application as an Internet infrastructure service. When the edge servers host the software of value-added services for content delivery, they may behave like transcoding proxy servers, remote callout servers, or surrogate servers [64]. These servers also demonstrate capability for processing and special hosting of the value-added Internet infrastructure services. Services provided by CDNs can be directory, Web storage, file transfer, and e-commerce services. Directory services are provided by the CDN for accessing the database servers. Users query for certain data is directed to the database servers and the results of frequent queries are cached at the edge servers of the CDN. Web storage service provided by the CDN is meant for storing content at the edge servers and is essentially based on the same techniques used for static content delivery. File transfer services facilitate the worldwide distribution of software, virus definitions, movies on-demand, and highly detailed medical images. All these contents are static by nature. Web services technologies are adopted by a CDN for their maintenance and delivery. E-commerce is highly popular for business transactions through the Web. Shopping carts for e-commerce services can be stored and maintained at the edge servers of the CDN and online transactions (e.g. third-party verification, credit card transactions) can be performed at the edge of CDNs. To facilitate this service, CDN edge servers should be enabled with dynamic content caching for e-commerce sites.

### 2.2.2 Content Distribution and Management

Content distribution and management is strategically vital in a CDN for efficient content delivery and for overall performance. Content distribution includes – *content selection and delivery* based on the type and frequency of specific user requests; *placement of surrogates* to some strategic positions so that the edge servers are close to the clients; and *content outsourcing* to decide which outsourcing methodology to follow. Content management is largely dependent on the techniques for *cache organization* (i.e. caching techniques, cache maintenance, and cache update). The content distribution and management taxonomy is shown in Fig. 2.5.

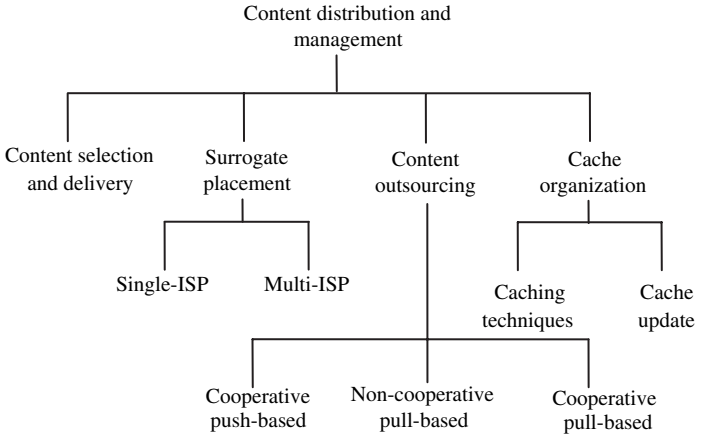



**Fig. 2.5** Content distribution and management taxonomy

#### 2.2.2.1 Content Selection and Delivery

The efficiency of content delivery lies in the right selection of content to be delivered to the end users. An appropriate content selection approach can assist in the reduction of client download time and server load. Figure 2.6 shows the taxonomy of content selection and delivery techniques. Content can be delivered to the customers in *full* or *partial*.

Full-site content selection and delivery is a simplistic approach where the surrogate servers perform "entire replication" in order to deliver the total content site to the end users. With this approach, a content provider configures its DNS in such a way that all client requests for its Web site are resolved by a CDN server, which then delivers all of the content. The main advantage of this approach is its simplicity. However, such a solution is not feasible considering the on-going increase in the size of Web objects. Although the price of storage hardware is decreasing, sufficient storage space on the edge servers is never guaranteed to store all the content

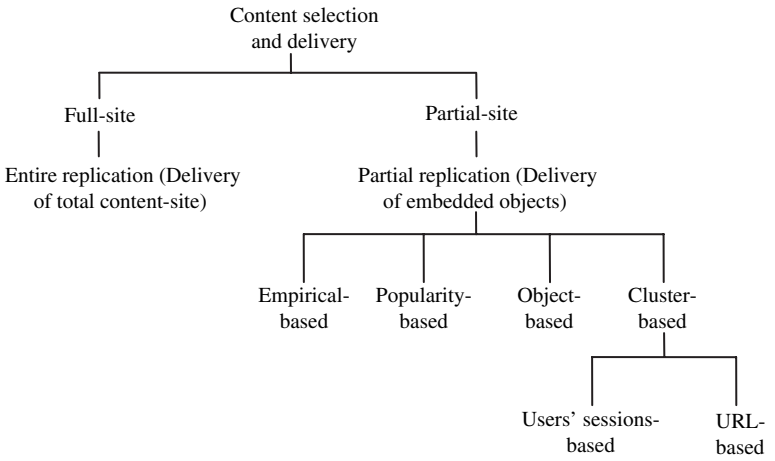



**Fig. 2.6** Taxonomy of content selection and delivery

from content providers. Moreover, since the Web content is not static, the problem of updating such a huge collection of Web objects is unmanageable.

On the other hand, in partial-site content selection and delivery, surrogate servers perform "partial replication" to deliver only embedded objects – such as Web page images – from the corresponding CDN. With partial-site content delivery, a content provider modifies its content so that links to specific objects have host names in a domain for which the CDN provider is authoritative. Thus, the base HTML page is retrieved from the origin server, while embedded objects are retrieved from CDN cache servers. A partial-site approach is better than the full-site approach in the sense that the former reduces load on the origin server and on the site's content generation infrastructure. Moreover, due to the infrequent change of embedded content, a partial-site approach exhibits better performance.

Content selection is dependent on the suitable management strategy used for replicating Web content. Based on the approach to select embedded objects to perform replication, partial-site approach can be further divided into – *empirical*, *popularity*, *object*, and *cluster*-based replication. In a empirical-based [23] approach, the Web site administrator empirically selects the content to be replicated to the edge servers. Heuristics are used in making such an empirical decision. The main drawback of this approach lies in the uncertainty in choosing the right heuristics. In a popularity-based approach, the most popular objects are replicated to the surrogates. This approach is time consuming and reliable objects request statistics is not guaranteed due to the popularity of each object varies considerably. Moreover, such statistics are often not available for newly introduced content. In an object-based approach, content is replicated to the surrogate servers in units of objects. This approach is greedy because each object is replicated to the surrogate server (under storage constraints) that gives the maximum performance gain [23, 102]. Although such a greedy approach achieve the best performance, it suffers from high

complexity to implement on real applications. In a cluster-based approach, Web content is grouped based on either correlation or access frequency and is replicated in units of content clusters. The clustering procedure is performed either by fixing the number of clusters or by fixing the maximum cluster diameter, since neither the number nor the diameter of the clusters can ever be known. The content clustering can be either users' sessions-based or URL-based. In a user's session-based [36] approach, Web log files are used to cluster a set of users' navigation sessions, which show similar characteristics. This approach is beneficial because it helps to determine both the groups of users with similar browsing patterns and the groups of pages having related content. In a URL-based approach, clustering of Web content is done based on Web site topology [23, 36]. The most popular objects are identified from a Web site and are replicated in units of clusters where the correlation distance between every pair of URLs is based on a certain correlation metric. Experimental results show that content replication based on such clustering approaches reduce client download time and the load on servers. However, these schemes suffer from the complexity involved to deploy them.

#### 2.2.2.2 Surrogate Placement

Since location of surrogate servers is closely related to the content delivery process, extra emphasis is put on the issue of choosing the best location for each surrogate. The goal of optimal surrogate placement is to reduce user perceived latency for accessing content and to minimize the overall network bandwidth consumption for transferring replicated content from servers to clients. The optimization of both of these metrics results in reduced infrastructure and communication cost for the CDN provider. Therefore, optimal placement of surrogate servers enables a CDN to provide high quality services and low CDN prices [88].

Figure 2.7 shows different surrogate server placement strategies. Theoretical approaches such as *minimum k-center problem* and *k-Hierarchically well-Separated Trees (k-HST)* model the server placement problem as the *center placement problem* which is defined as follows: for the placement of a given number of centers, minimize the maximum distance between a node and the nearest center. The k-HST [16, 47] algorithm solves the server placement problem according to graph theory. In this approach, the network is represented as a graph $G(V,E)$, where $V$ is the set of nodes and $E \subseteq V \times V$ is the set of links. The algorithm consists of two

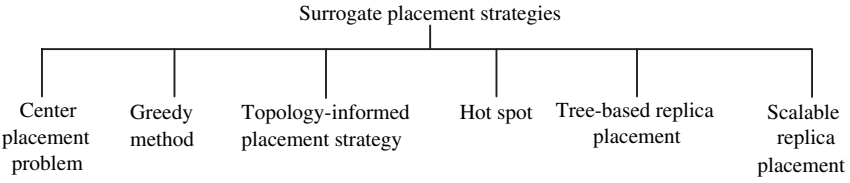



**Fig. 2.7** Surrogate placement strategies

phases. In the first phase, a node is arbitrarily selected from the complete graph (parent partition) and all the nodes which are within a random radius from this node form a new partition (child partition). The radius of the child partition is a factor of $k$ smaller than the diameter of the parent partition. This process continues until each of the nodes is in a partition of its own. Thus the graph is recursively partitioned and a tree of partitions is obtained with the root node being the entire network and the leaf nodes being individual nodes in the network. In the second phase, a virtual node is assigned to each of the partitions at each level. Each virtual node in a parent partition becomes the parent of the virtual nodes in the child partitions and together the virtual nodes form a tree. Afterwards, a greedy strategy is applied to find the number of centers needed for the resulted $k$-HST tree when the maximum center-node distance is bounded by $D$. The minimum $k$-center problem [47] can be described as follows: (1) Given a graph $G(V,E)$ with all its edges arranged in non-decreasing order of edge cost $c : c(e_1) \leq c(e_2) \leq \ldots. \leq c(e_m)$, construct a set of square graphs $G^2{}_1, G^2{}_2, \ldots., G^2{}_m$. Each square graph of $G$, denoted by $G^2$ is the graph containing nodes $V$ and edges $(u,v)$ wherever there is a path between $u$ and $v$ in $G$. (2) Compute the maximal independent set $M_i$ for each $G^2{}_i$. An independent set of $G^2$ is a set of nodes in G that are at least three hops apart in $G$ and a maximal independent set $M$ is defined as an independent set $V'$ such that all nodes in $V - V'$ are at most one hop away from nodes in $V'$. (3) Find smallest $i$ such that $M_i \leq K$, which is defined as $j$. (4) Finally, $M_j$ is the set of $K$ center.

Due to the computational complexity of these algorithms, some heuristics such as *Greedy replica placement* and *Topology-informed placement strategy* have been developed. These suboptimal algorithms take into account the existing information from CDN, such as workload patterns and the network topology. They provide sufficient solutions with lower computation cost. The greedy algorithm [59] chooses $M$ servers among $N$ potential sites. In the first iteration, the cost associated with each site is computed. It is assumed that access from all clients converges to the site under consideration. Hence, the lowest-cost site is chosen. In the second iteration, the greedy algorithm searches for a second site (yielding the next lowest cost) in conjunction with the site already chosen. The iteration continues until $M$ servers have been chosen. The greedy algorithm works well even with imperfect input data. But it requires the knowledge of the clients locations in the network and all pair wise inter-node distances. In topology-informed placement strategy [48], servers are placed on candidate hosts in descending order of outdegrees (i.e. the number of other nodes connected to a node). Here the assumption is that nodes with more outdegrees can reach more nodes with smaller latency. This approach uses Autonomous Systems (AS) topologies where each node represents a single AS and node link corresponds to Border Gateway Protocol (BGP) peering. In an improved topology-informed placement strategy [81], router-level Internet topology is used instead of AS-level topology. In this approach, each LAN associated with a router is a potential site to place a server, rather than each AS being a site.

Other server placement algorithms like *Hot Spot* [78] and *Tree-based* [63] replica placement are also used in this context. The *hotspot* algorithm places replicas near the clients generating greatest load. It sorts the $N$ potential sites according to the

amount of traffic generated surrounding them and places replicas at the top $M$ sites that generate maximum traffic. The *tree-based* replica placement algorithm is based on the assumption that the underlying topologies are trees. This algorithm models the replica placement problem as a dynamic programming problem. In this approach, a tree $T$ is divided into several small trees $T_i$ and placement of $t$ proxies is achieved by placing $t_i'$ proxies in the best way in each small tree $T_i$, where $t = \sum_i t_i'$. Another example is *Scan* [21], which is a scalable replica management framework that generates replicas on demand and organizes them into an application-level multicast tree. This approach minimizes the number of replicas while meeting clients' latency constraints and servers' capacity constraints. More information on Scan can be found in Chap. 3 of this book.

For surrogate server placement, the CDN administrators also determine the optimal number of surrogate servers using *single-ISP* and *multi-ISP* approach [95]. In the Single-ISP approach, a CDN provider typically deploys at least 40 surrogate servers around the network edge to support content delivery [30]. The policy in a single-ISP approach is to put one or two surrogates in each major city within the ISP coverage. The ISP equips the surrogates with large caches. An ISP with global network can thus have extensive geographical coverage without relying on other ISPs. The drawback of this approach is that the surrogates may be placed at a distant place from the clients of the CDN provider. In Multi-ISP approach, the CDN provider places numerous surrogate servers at as many global ISP Points of Presence (POPs) as possible. It overcomes the problems with single-ISP approach and surrogates are placed close to the users and thus content is delivered reliably and timely from the requesting client's ISP. Large CDN providers such as Akamai have more than 25000 servers [1, 29]. Other than the cost and complexity of setup, the main disadvantage of the multi-ISP approach is that each surrogate server receives fewer (or no) content requests which may result in idle resources and poor CDN performance [71]. Estimation of performance of these two approaches shows that single-ISP approach works better for sites with low-to-medium traffic volumes, while the multi-ISP approach is better for high-traffic sites [30].

#### 2.2.2.3 Content Outsourcing

Given a set of properly placed surrogate servers in a CDN infrastructure and a chosen content for delivery, choosing an efficient content outsourcing practice is crucial. Content outsourcing is performed using *cooperative push-based*, *non-cooperative pull-based*, or *cooperative pull-based* approaches.

Cooperative push-based approach depends on the pre-fetching of content to the surrogates. Content is pushed to the surrogate servers from the origin, and surrogate servers cooperate to reduce replication and update cost. In this scheme, the CDN maintains a mapping between content and surrogate servers, and each request is directed to the closest surrogate server or otherwise the request is directed to the origin server. Under this approach, greedy-global heuristic algorithm is suitable for making replication decision among cooperating surrogate servers [54]. Still it is

considered as a theoretical approach since it has not been used by any commercial CDN provider [23, 36].

In non-cooperative pull-based approach, client requests are directed to their closest surrogate servers. If there is a cache miss, surrogate servers pull content from the origin server. Most popular CDN providers (e.g. Akamai, Mirror Image) use this approach. The drawback of this approach is that an optimal server is not always chosen to serve content request [49]. Many CDNs use this approach since the cooperative push-based approach is still at the experimental stage [71].

The cooperative pull-based approach differs from the non-cooperative approach in the sense that surrogate servers cooperate with each other to get the requested content in case of a cache miss. In the cooperative pull-based approach client requests are directed to the closest surrogate through DNS redirection. Using a distributed index, the surrogate servers find nearby copies of requested content and store it in the cache. The cooperative pull-based approach is reactive wherein a data object is cached only when the client requests it. An academic CDN Coral [34], using a distributed index, follows the cooperative pull-based approach where the proxies cooperate each other in case of case miss.

In the context of content outsourcing, it is crucial to determine in which surrogate servers the outsourced content should be replicated. Several works can be found in literature demonstrating the effectiveness of different replication strategies for outsourced content. Kangasharju et al. [54] have used four heuristics, namely *random*, *popularity*, *greedy-single*, and *greedy-global*, for replication of outsourced content. Tse [94] has presented a set of greedy approaches where the placement is occurred by balancing the loads and sizes of the surrogate servers. Pallis et al. [72] have presented a self-tuning, parameterless algorithm called *lat-cdn* for optimally placing outsourced content in CDN's surrogate servers. This algorithm uses object's latency to make replication decision. An object's latency is defined as the delay between a request for a Web object and receiving the object in its entirety. An improvement of the lat-cdn algorithm is *il2p* [70], which places the outsourced objects to surrogate servers with respect to the latency and load of the objects.

#### 2.2.2.4 Cache Organization and Management

Content management is essential for CDN performance, which is mainly dependent on the cache organization approach followed by the CDN. Cache organization is in turn composed of the caching techniques used and the frequency of cache update to ensure the freshness, availability, and reliability of content. Other than these two, the cache organization may also involve the integrated use of caching and replication on a CDN's infrastructure. Such integration may be useful for a CDN for effective content management. Potential performance improvement is also possible in terms of perceived latency, hit ratio, and byte hit ratio if replication and caching are used together in a CDN [91]. Moreover, the combination of caching with replication assists to fortify against flash crowd events. In this context, Stamos et al. [90] have presented a generic non-parametric heuristic method that integrates Web caching with

content replication. They have developed a placement similarity approach, called SRC, for evaluating the level of integration. Another integrated approach called Hybrid, which combines static replication and Web caching using an analytic model of LRU is presented by Bakiras et al. [13]. Hybrid gradually fills the surrogate servers caches with static content in each iteration, as long as it contributes to the optimization of response times. More information on the integrated use of caching and replication can be found in Chap. 4 and Chap. 5 of this book.

Content caching in CDNs can be *intra-cluster* or *inter-cluster* basis. A taxonomy of caching techniques is shown in Fig. 2.8. *Query-based*, *digest-based*, *directory-based*, or *hashing-based* scheme can be used for intra-cluster caching of content. In a query-based [101] scheme, on a cache miss a CDN server broadcasts a query to other cooperating CDN servers. The problems with this scheme are the significant query traffic and the delay because a CDN server has to wait for the last "miss" reply from all the cooperating surrogates before concluding that none of its peers has the requested content. Because of these drawbacks, the query-based scheme suffers from implementation overhead. The digest-based [83] approach overcomes the problem of flooding queries in query-based scheme. In the digest-based scheme, each of the CDN servers maintains a digest of content held by the other cooperating surrogates. The cooperating surrogates are informed about any sort of update of the content by the updating CDN server. On checking the content digest, a CDN server can take the decision to route a content request to a particular surrogate. The main drawback is that it suffers from update traffic overhead, because of the frequent exchange of the update traffic to make sure that the cooperating surrogates have correct information about each other. The directory-based [38] scheme is a centralized version of the digest-based scheme. In directory-based scheme, a centralized server keeps content information of all the cooperating surrogates inside a cluster. Each CDN server only notifies the directory server when local updates occur and queries the directory server whenever there is a local cache miss. This scheme experiences potential bottleneck and single point of failure since the directory server receives update and query traffic from all cooperating surrogates. In a hashing-based [55, 96] scheme, the cooperating CDN servers maintain the same hashing function. A designated CDN server holds a content based on content's URL, IP addresses of the CDN servers, and the hashing function. All requests for that particular content are directed to that designated server. Hashing-based scheme is more

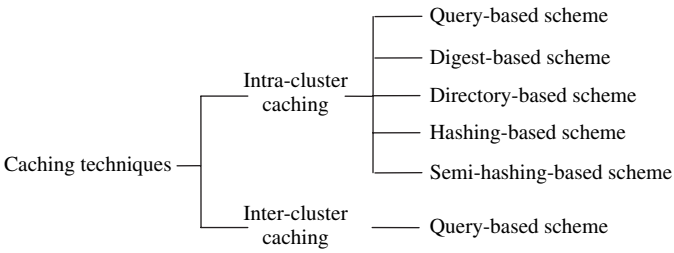



**Fig. 2.8** Caching techniques taxonomy

efficient than other schemes since it has the smallest implementation overhead and highest content sharing efficiency. However, it does not scale well with local requests and multimedia content delivery since the local client requests are directed to and served by other designated CDN servers. To overcome this problem, a semi-hashing-based scheme [24, 67] can be followed. Under the semi-hashing-based scheme, a local CDN server allocates a certain portion of its disk space to cache the most popular content for its local users and the remaining portion to cooperate with other CDN servers via a hashing function. Like pure hashing, semi-hashing has small implementation overhead and high content sharing efficiency. In addition, it has been found to significantly increase the local hit rate of the CDN.

A hashing-based scheme is not appropriate for inter-cluster cooperative caching, because representative CDN servers of different clusters are normally distributed geographically. The digest-based or directory-based scheme is also not suitable for inter-cluster caching since the representative CDN servers have to maintain a huge content digest and/or directory including the content information of CDN servers in other clusters. Hence, a query-based scheme can be used for inter-cluster caching [68]. In this approach, when a cluster fails to serve a content request, it queries other neighboring cluster(s). If the content can be obtained from this neighbor, it replies with a "hit" message or if not, it forwards the request to other neighboring clusters. All the CDN servers inside a cluster use hashing based scheme for serving content request and the representative CDN server of a cluster only queries the designated server of that cluster to serve a content request. Hence, this scheme uses the hashing-based scheme for intra-cluster content routing and the query-based scheme for inter-cluster content routing. This approach improves performance since it limits flooding of query traffic and overcomes the problem of delays when retrieving content from remote servers through the use of a Timeout and Time-to-Live (TTL) value with each query message.

Cached objects in the surrogate servers of a CDN have associated expiration times after which they are considered stale. Ensuring the freshness of content is necessary to serve the clients with up to date information. If there are delays involved in propagating the content, a CDN provider should be aware that the content may be inconsistent and/or expired. To manage the consistency and freshness of content at replicas, CDNs deploy different cache update techniques. The taxonomy of cache update mechanisms is shown in Fig. 2.9.

The most common cache update method is the *periodic update*. To ensure content consistency and freshness, the content provider configures its origin Web servers to provide instructions to caches about what content is cacheable, how long different content is to be considered fresh, when to check back with the origin server

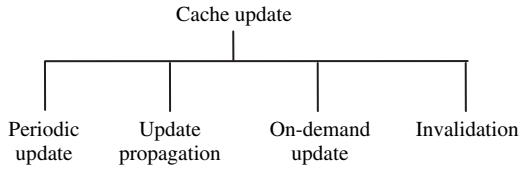



**Fig. 2.9** Cache update taxonomy

for updated content, and so forth [41]. With this approach, caches are updated in a regular fashion. But this approach suffers from significant levels of unnecessary traffic generated from update traffic at each interval. The *update propagation* is triggered with a change in content. It performs active content pushing to the CDN cache servers. In this mechanism, an updated version of a document is delivered to all caches whenever a change is made to the document at the origin server. For frequently changing content, this approach generates excess update traffic. *On-demand update* is a cache update mechanism where the latest copy of a document is propagated to the surrogate cache server based on prior request for that content. This approach follows an assume nothing structure and content is not updated unless it is requested. The disadvantage of this approach is the back-and-forth traffic between the cache and origin server in order to ensure that the delivered content is the latest. Another cache update approach is *invalidation*, in which an invalidation message is sent to all surrogate caches when a document is changed at the origin server. The surrogate caches are blocked from accessing the documents when it is being changed. Each cache needs to fetch an updated version of the document individually later. The drawback of this approach is that it does not make full use of the distribution network for content delivery and belated fetching of content by the caches may lead to inefficiency of managing consistency among cached contents.

Generally, CDNs give the content provider control over freshness of content and ensure that all CDN sites are consistent. However, content providers themselves can build their own policies or use some heuristics to deploy organization specific caching policies. In the first case, content providers specify their caching policies in a format unique to the CDN provider, which propagates the rule sets to its caches. These rules specify instructions to the caches on how to maintain the freshness of content through ensuring consistency. In the latter case, a content provider can apply some heuristics rather than developing complex caching policies. With this approach, some of the caching servers adaptively learn over time about the frequency of change of content at the origin server and tune their behavior accordingly.

### *2.2.3 Request-Routing*

A *request-routing system* is responsible for routing client requests to an appropriate surrogate server for the delivery of content. It consists of a collection of network elements to support request-routing for a single CDN. It directs client requests to the replica server "closest" to the client. However, the closest server may not be the best surrogate server for servicing the client request [22]. Hence, a request-routing system uses a set of metrics such as network proximity, client perceived latency, distance, and replica server load in an attempt to direct users to the closest surrogate that can best serve the request. The content selection and delivery techniques (i.e. full-site and partial-site) used by a CDN have a direct impact on the design of its request-routing system. If the full-site approach is used by a CDN, the request-routing system assists to direct the client requests to the surrogate servers as they hold all the outsourced content. On the other hand, if the partial-site approach is

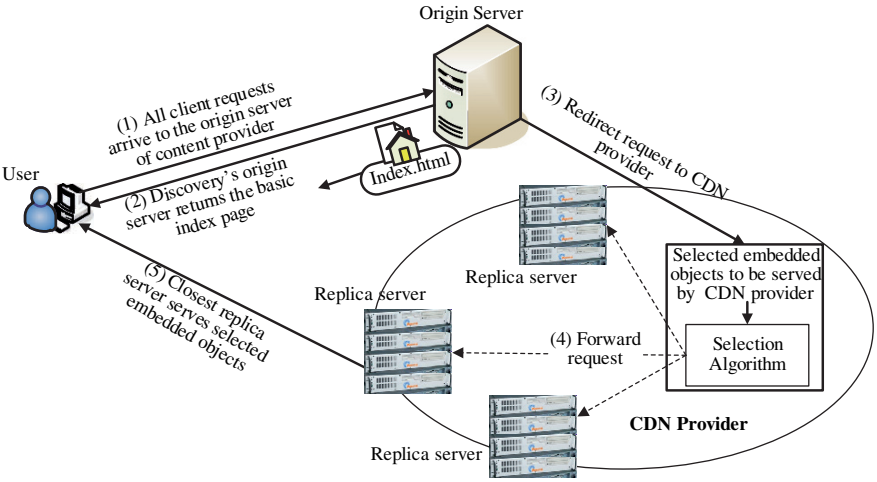



**Fig. 2.10** Request-routing in a CDN environment

used, the request-routing system is designed in such a way that on receiving the client request, the origin server delivers the basic content while surrogate servers deliver the embedded objects. The request-routing system in a CDN has two parts: deployment of a request-routing algorithm and use of a request-routing mechanism [89]. A request-routing algorithm is invoked on receiving a client request. It specifies how to select an edge server in response to the given client request. On the other hand, a request-routing mechanism is a way to inform the client about the selection. Such a mechanism at first invokes a request-routing algorithm and then informs the client about the selection result it obtains.

Figure 2.10 provides a high-level view of the request-routing in a typical CDN environment. The interaction flows are: (1) the client requests content from the content provider by specifying its URL in the Web browser. Client's request is directed to its origin server; (2) when origin server receives a request, it makes a decision to provide only the basic content (e.g. index page of the Web site) that can be served from its origin server; (3) to serve the high bandwidth demanding and frequently asked content (e.g. embedded objects – fresh content, navigation bar, and banner advertisements), content provider's origin server redirects client's request to the CDN provider; (4) using the proprietary selection algorithm, the CDN provider selects the replica server which is "closest" to the client, in order to serve the requested embedded objects; (5) selected replica server gets the embedded objects from the origin server, serves the client requests and caches it for subsequent request servicing.

#### 2.2.3.1 Request-Routing Algorithms

The algorithms invoked by the request-routing mechanisms can be *adaptive* or *non-adaptive* (Fig. 2.11). Adaptive algorithms consider the current system condition to

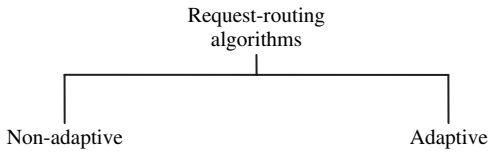



**Fig. 2.11** Taxonomy of request-routing algorithms

select a cache server for content delivery. The current condition of the system is obtained by estimating some metrics like load on the replica servers or the congestion of selected network links. Non-adaptive request-routing algorithms use some heuristics for selecting a cache server rather than considering the current system condition. A non-adaptive algorithm is easy to implement, while the former is more complex. Complexity of adaptive algorithms arises from their ability to change behavior to cope with an enduring situation. A non-adaptive algorithm works efficiently when the assumptions made by the heuristics are met. On the other hand, an adaptive algorithm demonstrates high system robustness [100] in the face of events like flash crowds.

An example of the most common and simple non-adaptive request-routing algorithm is round-robin, which distributes all requests to the CDN cache servers and attempts to balance load among them [93]. It is assumed that all the cache servers have similar processing capability and that any of them can serve any client request. Such simple algorithms are efficient for clusters, where all the replica servers are located at the same place [69]. But the round-robin request-routing algorithm does not perform well for wide area distributed systems where the cache servers are located at distant places. In this case it does not consider the distance of the replica servers. Hence, client requests may be directed to more distant replica servers, which cause poor performance perceived by the users. Moreover, the aim of load balancing is not fully achieved since processing different requests can involve significantly different computational costs.

In another non-adaptive request-routing algorithm, all replica servers are ranked according to the predicted load on them. Such prediction is done based on the number of requests each of the servers has served so far. This algorithm takes client-server distance into account and client requests are directed to the replica servers in such a way that load is balanced among them. The assumption here is that the replica server load and the client-server distance are the most influencing factors for the efficiency of request processing [89]. Though it has been observed by Aggarwal et al. [9] that deploying this algorithm can perform well for request-routing, the client perceived performance may still be poor.

Several other interesting non-adaptive request-routing algorithms are implemented in the Cisco DistributedDirector [28]. One of these algorithms considers the percentage of client requests that each replica server receives. A server receiving more requests is assumed to be more powerful. Hence, client requests are directed to the more powerful servers to achieve better resource utilization. Another algorithm defines preference of one server over another in order to delegate the former to serve client requests. The DistributedDirector also supports random request distribution to replica servers. Furthermore, some other non-adaptive algorithms can be found which considers the client's geographic location to redirect requests to the nearby replica.

However, this algorithm suffers from the fact that client requests may be assigned to overloaded replica servers, which may degrade client perceived performance.

Karger et al. [55] have proposed a request-routing algorithm to adapt to hotspots. It calculates a hashing function $h$ from a large space of identifiers, based on the URL of the content. This hashing function is used to route client requests efficiently to a logical ring consisting of cache servers with IDs from the same space. It is assumed that the cache server having the smallest ID larger than $h$ is responsible for holding the referenced data. Hence, client requests are directed to it. Variations of this algorithm have been used in the context of intra-cluster caching [67, 68] and P2P file sharing systems [14].

Globule [76] uses an adaptive request-routing algorithm that selects the replica server closest to the clients in terms of network proximity [93]. The metric estimation in Globule is based on path length which is updated periodically. The metric estimation service used in globule is passive, which does not introduce any additional traffic to the network. However, Huffaker et al. [45] show that the distance metric estimation procedure is not very accurate.

Andrews et al. [10] and Ardiaz et al. [12] have proposed adaptive request-routing algorithms based on client-server latency. In this approach, either client access logs or passive server-side latency measurements are taken into account, and the algorithms decide to which replica server the client requests are to be sent. Hence, they redirect a client request to a replica which has recently reported the minimal latency to the client. These algorithms are efficient since they consider latency measurements. However, they require the maintenance of central database of measurements, which limits the scalability of systems on which these algorithms are deployed [89].

Cisco DistributedDirector [28] has implemented an adaptive request-routing algorithm. The request-routing algorithm deployed in this system takes into account a weighted combination of three metrics, namely – inter-AS distance, intra-AS distance, and end-to-end latency. Although this algorithm is flexible since it makes use of three metrics, the deployment of an agent in each replica server for metric measurement makes it complex and costly. Moreover, the active latency measurement techniques used by this algorithm introduce additional traffic to the Internet. Furthermore, the isolation of DistributedDirector component from the replica server makes it unable to probe the servers to obtain their load information.

Akamai [1, 29] uses a complex request-routing algorithm which is adaptive to flash crowds. It takes into consideration a number of metrics such as replica server load, the reliability of loads between the client and each of the replica servers, and the bandwidth that is currently available to a replica server. This algorithm is proprietary to Akamai and the technology details have not been revealed.

#### 2.2.3.2 Request-Routing Mechanisms

Request-routing mechanisms inform the client about the selection of replica server generated by the request-routing algorithms. Request-routing mechanisms can be classified according to several criteria. In this section we classify them according

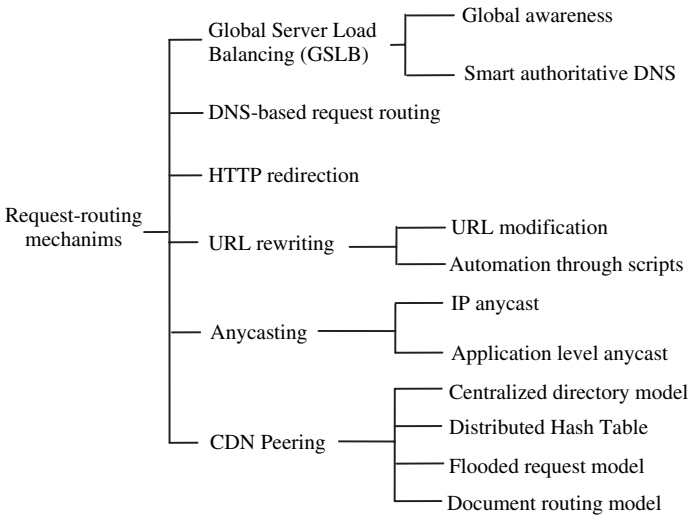



**Fig. 2.12** Taxonomy of request-routing mechanisms

to request processing. As shown in Fig. 2.12, they can be classified as: *Global Server Load Balancing (GSLB)*, *DNS-based request-routing*, *HTTP redirection*, *URL rewriting*, *anycasting*, and *CDN peering*.

In GSLB [44] approach, service nodes, which serve content to the end users, consisting of a GSLB-enabled Web switch and a number of real Web servers are distributed in several locations around the world. Two new capabilities of the service nodes allow them to support global server load balancing. The first is *global awareness* and the second is *smart authoritative DNS* [44]. In local server load balancing, each service node is aware of the health and performance information of the Web servers directly attached to it. In GSLB, one service node is aware of the information in other service nodes and includes their virtual IP address in its list of servers. Hence, the Web switches making up each service node are globally aware and each knows the addresses of all the other service nodes. They also exchange performance information among the Web switches in GSLB configuration. To make use of such global awareness, the GSLB switches act as a smart authoritative DNS for certain domains. The advantage of GSLB is that since the service nodes are aware of each other, each GSLB switch can select the best surrogate server for any request. Thus, this approach facilitates choosing servers not only from the pool of locally connected real servers, but also the remote service nodes. Another significant advantage of GSLB is that the network administrator can add GSLB capability to the network without adding any additional networking devices. A disadvantage of GSLB is the manual configuration of the service nodes to enable them with GSLB capability.

In DNS-based request-routing approach, the content distribution services rely on the modified DNS servers to perform the mapping between a surrogate server's symbolic name and its numerical IP address. It is used for full-site content selection and

delivery. In DNS-based request-routing, a domain name has multiple IP addresses associated to it. When an end user's content request comes, the DNS server of the service provider returns the IP addresses of servers holding the replica of the requested object. The client's DNS resolver chooses a server among these. To decide, the resolver may issue probes to the servers and choose based on response times to these probes. It may also collect historical information from the clients based on previous access to these servers. Both full and partial-site CDN providers use DNS redirection. The performance and effectiveness of DNS-based request-routing has been examined in a number of recent studies [15, 41, 65, 86]. The advantage of this approach is the transparency as the services are referred to by means of their DNS names, and not their IP addresses. DNS-based approach is extremely popular because of its simplicity and independence from any actual replicated service. Since it is incorporated to the name resolution service it can be used by any Internet application [89]. In addition, the ubiquity of DNS as a directory service provides advantages during request-routing. The disadvantage of DNS-based request-routing is that, it increases network latency because of the increase in DNS lookup times. CDN administrators typically resolve this problem by splitting CDN DNS into two levels (low-level DNS and high-level DNS) for load distribution [58]. Another limitation is that DNS provides the IP address of the client's Local DNS (LDNS), rather than the client's IP address. Clients are assumed to be near to the LDNS. When DNS-based server selection is used to choose a nearby server, the decision is based on the name server's identity, not the client's. Thus, when clients and name servers are not proximal, the DNS-based approach may lead to poor decisions. Most significantly, DNS cannot be relied upon to control all incoming requests due to caching of DNS data at both the ISP and client level. Indeed, it can have control over as little as 5% of requests in many instances [20]. Furthermore, since clients do not access the actual domain names that serve their requests, it leads to the absence of any alternate server to fulfill client requests in case of failure of the target surrogate server. Thus, in order to remain responsive to changing network or server conditions, DNS-based schemes must avoid client-side caching or decisions.

HTTP redirection propagates information about replica server sets in HTTP headers. HTTP protocols allow a Web server to respond to a client request with a special message that tells the client to re-submit its request to another server. HTTP redirection can be used for both full-site and partial-site content selection and delivery. This mechanism can be used to build a special Web server, which accepts client requests, chooses replica servers for them and redirects clients to those servers. It requires changes to both Web servers and clients to process extra headers. The main advantage of this approach is flexibility and simplicity. Another advantage is that replication can be managed at fine granularity, since individual Web pages are considered as a granule [75]. The most significant disadvantage of HTTP redirection is the lack of transparency. Moreover, the overhead perceived through this approach is significant since it introduces extra message round-trip into request processing as well as over HTTP.

Though most CDN systems use a DNS based routing scheme, some systems use the URL rewriting or Navigation hyperlink. It is mainly used for partial-site

content selection and delivery where embedded objects are sent as a response to client requests. In this approach, the origin server redirects the clients to different surrogate servers by rewriting the dynamically generated pages' URL links. For example, with a Web page containing an HTML file and some embedded objects, the Web server would modify references to embedded objects so that the client could fetch them from the best surrogate server. To automate this process, CDNs provide special scripts that transparently parse Web page content and replace embedded URLs [58]. URL rewriting can be pro-active or reactive. In the pro-active URL rewriting, the URLs for embedded objects of the main HTML page are formulated before the content is loaded in the origin server. The reactive approach involves rewriting the embedded URLs of an HTML page when the client request reaches the origin server. The main advantage of URL rewriting is that the clients are not bound to a single surrogate server, because the rewritten URLs contain DNS names that point to a group of surrogate servers. Moreover, finer level of granularity can be achieved through this approach since embedded objects can be considered as granule. The disadvantages through this approach are the delay for URL-parsing and the possible bottleneck introduced by an in-path element. Another disadvantage is that content with modified reference to the nearby surrogate server rather than to the origin server is non-cacheable.

The anycasting approach can be divided into *IP anycasting* and *Application-level anycasting*. IP anycasting, proposed by Partridge et al. [73], assumes that the same IP address is assigned to a set of hosts and each IP router holds a path in its routing table to the host that is closest to this router. Thus, different IP routers have paths to different hosts with the same IP address. IP anycasting can be suitable for request-routing and service location. It targets network-wide replication of the servers over potentially heterogeneous platforms. A disadvantage of IP anycasting is that some parts of the IP address space is allocated for anycast address. Fei et al. [32] proposed an application level anycasting mechanism where the service consists of a set of *anycast resolvers*, which perform the *anycast domain names* to IP address mapping. Clients interact with the anycast resolvers by generating an anycast query. The resolver processes the query and replies with an anycast response. A metric database, associated with each anycast resolver contains performance data about replica servers. The performance is estimated based on the load and the request processing capability of the servers. The overhead of the performance measurement is kept at a manageable level. The performance data can be used in the selection of a server from a group, based on user-specified performance criteria. An advantage of application level anycasting is that better flexibility can be achieved through this approach. One disadvantage of this approach is that deploying the anycasting mechanism for request-routing requires changes to the servers as well as to the clients. Hence, it may lead to increased cost considering possibly large number of servers and clients.

Peer-to-peer content networks are formed by symmetrical connections between host computers. Peered CDNs deliver content on each other's behalf. Thus, a CDN could expand its reach to a larger client population by using partnered CDN servers and their nearby forward proxies. A content provider usually has contracts with

only one CDN and each CDN contacts other peer CDNs on the content provider's behalf [74]. Peering CDNs are more fault-tolerant as the necessary information retrieval network can be developed on the peering members themselves instead of relying on a dedicated infrastructure like traditional CDNs. To locate content in CDN peering, a *centralized directory model*, *Distributed Hash Table (DHT)*, *flooded request model*, or *document routing model* can be used [44, 66].

In a centralized directory model, peers contact a centralized directory where all the peers publish content that they want to share with others. When the directory receives a request it responses with the information of the peer that holds the requested content. When more than one peer matches the request, the best peer is selected based on metrics such as network proximity, highest bandwidth, least congestion and highest capacity. On receiving the response from the directory, the requesting peer contacts the peer that it has been referred to for content retrieval. The drawback of this approach is that, the centralized directory is subject to a single point of failure. Moreover, the scalability of a system based on a centralized directory is limited to the capacity of the directory. Archi [31], WAIS [52] are the examples of centralized directory systems for retrieving FTP files located on various systems. In systems using DHTs, peers are indexed through hashing keys within a distributed system. Then a peer holding the desired content can be found through applying queries and lookup functions [43]. Example of a protocol using DHT is Chord [92]. The advantage of this approach is the ability to perform load balancing by offloading excess loads to the less-loaded peers [18]. In the flooded request model, a request from a peer is broadcast to the peers directly connected to it. These peers in turn forward the messages to other peers directly connected to them. This process continues until the request is answered or some broadcast limit is reached. The drawback of this approach is that it generates unnecessary network traffic and hence, it requires enormous bandwidth. Thus, it suffers from scalability problem and it limits the size of the network [44]. Gnutella [8, 25] is the example of a system using the flooded request model. In document routing model an authoritative peer is asked for referral to get the requested content. Each peer in the model is helpful, though they partially complete the referral information [44]. In this approach, each peer is responsible for a range of file IDs. When a peer wants to get some file, it sends a request a request containing the file ID. The request is forwarded to the peer whose ID is most similar to the file ID. Once the file is located, it is transferred to the requesting peer. The main advantage of this approach is that it can complete a comprehensive search within a bounded *O(log n)* number of steps. Moreover, it shows good performance and is scalable enough to grow significantly large.

### *2.2.4 Performance Measurement*

Performance measurement of a CDN is done to measure its ability to serve the customers with the desired content and/or service. Typically five key metrics are used by the content providers to evaluate the performance of a CDN [30, 37, 58]. Those are:

- *Cache hit ratio:* It is defined as the ratio of the number of cached documents versus total documents requested. A high hit rate reflects that a CDN is using an effective cache technique to manage its caches.
- *Reserved bandwidth:* It is the measure of the bandwidth used by the origin server. It is measured in bytes and is retrieved from the origin server.
- *Latency:* It refers to the user perceived response time. Reduced latency indicates that less bandwidth is reserved by the origin server.
- *Surrogate server utilization:* It refers to the fraction of time during which the surrogate servers remain busy. This metric is used by the administrators to calculate CPU load, number of requests served and storage I/O usage.
- *Reliability:* Packet-loss measurements are used to determine the reliability of a CDN. High reliability indicates that a CDN incurs less packet loss and is always available to the clients.

Performance measurement can be accomplished based on internal performance measures as well as from the customer perspective. A CDN provider's own performance testing can be misleading, since it may perform well for a particular Web site and/or content, but poorly for others. To ensure reliable performance measurement, a CDN's performance can be measured by independent third-party such as Keynote Systems [3] or Giga Information Group [6]. The performance measurement taxonomy is shown in Fig. 2.13.

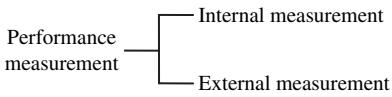



**Fig. 2.13** Performance measurement taxonomy

#### 2.2.4.1 Internal Measurement

CDN servers could be equipped with the ability to collect statistics in order to get an end-to-end measurement of its performance. In addition, deployment of probes (hardware and/or software) throughout the network and correlation of the information collected by probes with the cache and server logs can be used to measure the end-to-end performance.

#### 2.2.4.2 External Measurement

In addition to internal performance measurement, external measurement of performance by an independent third-party informs the CDN customers about the verified and guaranteed performance. This process is efficient since the independent performance-measuring companies support benchmarking networks of strategically located measurement computers connected through major Internet backbones in several cities. These computers measure how a particular Web site performs from the end user's perspective, considering service performance metrics in critical areas [95].

#### 2.2.4.3 Network Statistics Acquisition for Performance Measurement

For internal and external performance measurement, different network statistics acquisition techniques are deployed based on several parameters. Such techniques may involve network probing, traffic monitoring, and feedback from surrogates. Typical parameters in the network statistics acquisition process include geographical proximity, network proximity, latency, server load, and server performance as a whole. Figure 2.14 presents the mechanisms used by the CDNs to perform network statistics acquisition.

*Network probing* is a measurement technique where the possible requesting entities are probed in order to determine one or more metrics from each surrogate or a set of surrogates. Network probing can be used for P2P-based cooperative CDNs where the surrogate servers are not controlled by a single CDN provider. Example of such probing technique is an ICMP ECHO message that is sent periodically from a surrogate or a set of surrogates to a potential requesting entity. Active probing techniques are sometimes not suitable and limited for some reasons. It introduces additional network latency which may be significant for small Web requests. Moreover, performing several probes to an entity often triggers intrusion-detection alerts, resulting in abuse complaints [35]. Probing sometimes may lead to an inaccurate metric as ICMP traffic can be ignored or reprioritized due to concerns of Distributed Denial of Service (DDoS) attacks. A distributed anycasting system by Freedman et al. [35] has shown that ICMP probes and TCP probes to high random ports are often dropped by firewalls and flagged as unwanted port scans.

*Traffic monitoring* is a measurement technique where the traffic between the client and the surrogate is monitored to know the actual performance metrics. Once the client connects, the actual performance of the transfer is measured. This data is then fed back into the request-routing system. An example of such traffic monitoring is to watch the packet loss from a client to a surrogate or the user perceived response time (latency) by observing the TCP behavior. Latency is the simplest and mostly used distance metric, which can be estimated by monitoring the number of packets (i.e. traffic) traveled along the route between client and the surrogate. A metric estimation system such as IDMaps [35] measures and disseminates distance information on the global Internet in terms of latency and bandwidth. This system considers two types of distance information based on timeliness – load sensitive and "raw" (where distance information is obtained considering no load on the network). The estimation of these information is performed through traffic monitoring with an update frequency on the order of days, or if necessary, hours.

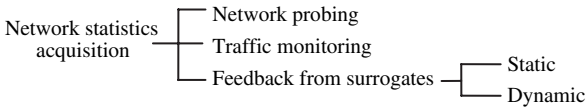



**Fig. 2.14** Network statistics acquisition techniques

*Feedback from surrogates* can be obtained by periodically probing a surrogate by issuing application specific requests (e.g. HTTP) and taking related measures. Feedback information can also be obtained from agents that are deployed in the surrogates. These agents can communicate a variety of metrics about their nodes. Methods for obtaining feedback information can be static or dynamic. Static methods select a route to minimize the number of hops or to optimize other static parameters. Dynamic probing allows computing round-trip time or QoS parameters in "real time" [33].

Figure 2.15 shows the different metrics used by CDNs to measure the network and system performance. *Geographical proximity* is a measure of identifying a user's location within a certain region. It is often used to redirect all users within a certain region to the same Point of Presence (POP). The measurement of such network proximity is typically derived through probing of BGP routing tables. The end user perceived *latency* is a useful metric to select the suitable surrogate for that user. *Packet loss* information through a network path is a measurement metric that is used to select the path with lowest error rate. *Average bandwidth*, *startup time* and *frame rate* are the metrics used to select the best path for streaming media delivery. Server load state can be computed based on metrics such as CPU load, network interface load, active connection, and storage I/O load. This metric is used to select the server with the aggregated least load.

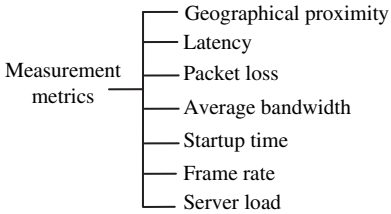



**Fig. 2.15** Metrics used for measuring network and system performance

#### 2.2.4.4 Performance Measurement through Simulation

Other than using internal and external performance measurement, researchers use simulation tools to measure a CDN's performance. Some researchers also experiment their CDN policies on real platforms such as PlanetLab [5]. The CDN simulators implemented in software are valuable tools for researchers to develop, test and diagnose a CDN's performance, since accessing real CDN traces and logs is not easy due to the proprietary nature of commercial CDNs. Such a simulation process is economical because of no involvement of dedicated hardware to carry out the experiments. Moreover, it is flexible because it is possible to simulate a link with any bandwidth and propagation delay and a router with any queue size and queue management technique. A simulated network environment is free of any uncontrollable factors such as unwanted external traffic, which the researchers may experience while running experiments in a real network. Hence, simulation results

are reproducible and easy to analyze. A wide range of network simulators [4, 7] are available which can be used to simulate a CDN to measure its performance. Moreover, there are also some specific CDN simulation systems [2, 7, 23, 54, 100] that allow a (closely) realistic approach for the research community and CDN developers to measure performance and experiment their policies. However, the results obtained from a simulation may be misleading if a CDN simulation system does not take into account several critical factors such as the bottlenecks that are likely to occur in a network, the number of traversed nodes etc., considering the TCP/IP network infrastructure.

## 2.3 Mapping of the Taxonomy to Representative CDNs

In this section, we provide the categorization and mapping of our taxonomy to a few representative CDNs that have been surveyed in Chap. 1 of this book. We also present the perceived insights and a critical evaluation of the existing systems while classifying them. Our analysis of the CDNs based on the taxonomy also examines the validity and applicability of the taxonomy.

### *2.3.1 CDN Composition Taxonomy Mapping*

Table 2.1 shows the annotation of the representative CDNs based on the CDN composition taxonomy. As shown in the table, the majority of the existing CDNs use overlay approach for CDN organization, while some use network approach or both. The use of both overlay and network approaches is common among commercial CDNs such as Akamai and Mirror Image. When a CDN provider uses a combination of these two approaches for CDN formation, a network element can be used to redirect HTTP requests to a nearby application-specific surrogate server.

Academic CDNs are built using P2P techniques, following an overlay approach. However, each of them differs in the way the overlay is built and deployed. For example, CoDeeN overlay consists of deployed "open" proxies, whereas Coral overlay (consisting of cooperative HTTP proxies and a network of DNS servers) is built relying on an underlying indexing infrastructure, and Globule overlay is composed of the end user nodes.

In an overlay approach, the following relationships are common – client-to-surrogate-to-origin server and network element-to-caching proxy. Inter-proxy relationship is also common among the CDNs, which supports inter-cache interaction. When using network approach, CDNs rely on the interaction of network elements for providing services through deploying request-routing logic to the network elements based on predefined policies. The overlay approach is preferred over the network approach because of the scope for new services integration and simplified

**Table 2.1** CDN composition taxonomy mapping

| CDN Name and Type | | CDN Organization | Servers | Relationships | Interaction Protocols | Content/Service Types |
|---|---|---|---|---|---|---|
| **Commercial CDNs** | Akamai | Network and overlay approach | Origin and replica servers | Client-to-surrogate-to-origin server, Network element-to-caching proxy, Inter-proxy | Network elements interaction, inter-cache interaction | Static content, dynamic content, streaming media, and services (network monitoring, geographic targeting) |
| | Edge Stream | Network approach | N/A | N/A | Network elements interaction | Video streaming, video hosting services |
| | Limelight Networks | Overlay approach | Origin and replica servers | Client-to-surrogate-to-origin server, Network element-to-caching proxy | Network elements interaction | Static content, streaming media |
| | Mirror Image | Network and Overlay approach | Origin and replica servers | Client-to-surrogate-to-origin server, Network element-to-caching proxy | Network elements interaction | Static content, streaming media, Web computing and reporting services |
| **Academic CDNs** | CoDeeN | Overlay approach with "open" proxies | Origin and replica/proxy (forward, reverse, redirector) servers | Client-to-surrogate-to-origin server, Network element-to-caching proxy, inter-proxy | Network elements interaction, inter-cache interaction | *Participating* users receive better performance to *most* sites; only provides static content |

**Table 2.1** (continued)

| CDN Name and Type | CDN Organization | Servers | Relationships | Interaction Protocols | Content/Service Types |
|---|---|---|---|---|---|
| Coral | Overlay approach with an underlying indexing infrastructure | Origin and replica/ (cooperative) proxy cache servers | Client-to-surrogate-to-origin server, Network element-to-caching proxy, inter-proxy | Network elements interaction, inter-cache interaction | Most *users* receive better performance to *participating* sites; only provides static content |
| Globule | Overlay approach with end user nodes | Origin, replica, backup and/or redirector servers | Client-to-surrogate-to-origin server, Network element-to-caching proxy, inter-node | Network elements interaction, inter-cache interaction | A Web site's performance and availability is improved; provides static content and monitoring services |

management of underlying network infrastructure. Offering a new service in overlay approach is as simple as distributing new code to CDN servers [61].

CDNs use origin and replica servers to perform content delivery. Most of the replica servers are used as Web servers for serving Web content. Some CDNs such as Akamai, EdgeStream, Limelight Networks, and Mirror Image use their replica servers as media servers for delivering streaming media and video hosting services. Replica servers can also be used for providing services like caching, large file transfer, reporting, and DNS services. In the academic CDN domain, proxy/replica servers can be configured for different purposes. For example, each CoDeeN node is capable of acting as a forward, a reverse, and a redirection proxy; Coral proxies are cooperative; and Globule node can play the role of an origin, replica, backup, and/or replica server.

From Table 2.1, it can also be seen that most of the CDNs are dedicated to provide particular content, since variation of services and content requires the CDNs to adopt application-specific characteristics, architectures and technologies. Most of them provide static content, while only some of them provide streaming media, broadcasting, and other services. While the main business goal of commercial CDNs is to gain profit through content and/or service delivery, the goal of academic CDNs differs from each other. As for instance, CoDeeN provides static content with the goal of providing *participating* users better performance to *most* Web sites; Coral aims to provide most *users* better performance to *participating* Web sites; and Globule targets to improve a Web site's performance, availability and resistance (to a certain extent) to flash crowds and the Slashdot effects.

### *2.3.2 Content Distribution and Management Taxonomy Mapping*

The mapping of the content distribution and management taxonomy to the representative CDNs is shown in Table 2.2.

Most of the CDNs support partial-site content delivery, while both full and partial-site content delivery is also possible. CDN providers prefer to support partial-site content delivery because it reduces load on the origin server and on the site's content generation infrastructure. Moreover, due to the infrequent change of embedded content, partial-site approach performs better than the full-site content delivery approach. Only few CDNs – Akamai, Mirror Image and Coral to be specific, are found to support clustering of contents. The content distribution infrastructure of other CDNs does not reveal any information whether other CDNs use any scheme for content clustering. Akamai and Coral cluster content based on users' sessions. This approach is beneficial because it helps to determine both the groups of users with similar browsing patterns and the groups of pages having related content. The only CDN to use the URL-based content clustering is Mirror Image. But URL-based approach is not popular because it suffers from the complexity involved to deploy them.

From the table it is clear that most of the representative CDNs with extensive geographical coverage follow the multi-ISP approach to place numerous number of surrogate servers at many global ISP POPs. Commercial CDNs such as Akamai, Limelight Networks, Mirror Image, and academic CDNs such as Coral [34] and CoDeeN use multi-ISP approach. The single-ISP approach suffers from the distant placement of the surrogates with respect to the locality of the end users. However, the setup cost, administrative overhead, and complexity associated with deploying and managing of the system in multi-ISP approach is higher. An exception to this can be found for sites with high traffic volumes. Multi-ISP approach performs better in this context since single-ISP approach is suitable only for sites with low-to-medium traffic volumes [95].

Content outsourcing of the commercial CDNs mostly use non-cooperative pull-based approach because of its simplicity enabled by the use of DNS redirection or URL rewriting. Cooperative push-based approach is still theoretical and none of the existing CDNs supports it. Cooperative pull-based approach involves complex technologies (e.g. DHT) as compared to the non-cooperative approach and it is used by the academic CDNs following P2P architecture [71]. Moreover, it imposes a large communication overhead (in terms of number of messages exchanged) when the number of clients is large. It also does not offer high fidelity when the content changes rapidly or when the coherency requirements are stringent.

From Table 2.2 it is also evident that representative commercial and academic CDNs with large geographic coverage, use inter-cluster (and a combination of inter and intra-cluster) caching. CDNs mainly use on-demand update as their cache update policy. Only Coral uses invalidation for updating caches since it delivers static content which changes very infrequently. Globule follows an adaptive cache update policy to dynamically choose between different cache consistency enforcement techniques. Of all the cache update policies, periodic update has the greatest reach since the caches are updated in a regular fashion. Thus, it has the potential to be most effective in ensuring cache content consistency. Update propagation and invalidation are not generally applicable as steady-state control mechanisms, and they can cause control traffic to consume bandwidth and processor resources that could otherwise be used for serving content [41]. Content providers themselves may administer to deploy specific caching mechanisms or heuristics for cache update. Distributing particular caching mechanism is simpler to administer but it has limited effects. On the other hand, cache heuristics are a good CDN feature for content providers who do not want to develop own caching mechanisms. However, heuristics will not deliver the same results as well-planned policy controls [41].

### *2.3.3 Request-Routing Taxonomy Mapping*

Table 2.3 maps the request-routing taxonomy to the representative CDNs. It can be observed from the table that DNS-based mechanisms are very popular for request-routing. The main reason of this popularity is its simplicity and the ubiquity of

**Table 2.2** Content distribution and management taxonomy mapping

| CDN Name | Content Selection and Delivery | Surrogate Placement | Content Outsourcing | Cache Organization |
|---|---|---|---|---|
| Akamai | Content selection<br>• Full and partial-site delivery<br>Content Clustering<br>• Users' sessions based | Multi-ISP approach; Hotspot placement by allocating more servers to sites experiencing high load | Non-cooperative pull-based | Caching technique<br>• Intra and inter-cluster caching<br>Cache update<br>• Update propagation<br>• On-demand |
| Edge Stream | Content selection<br>• Partial-site delivery<br>Content Clustering<br>N/A | Single-ISP approach | Non-cooperative pull-based | Caching technique<br>• Inter-cluster caching<br>Cache update<br>N/A |
| Limelight Networks | Content selection<br>• Partial-site delivery<br>Content Clustering<br>N/A | Multi-ISP approach | Non-cooperative pull-based | Caching technique<br>• Intra-cluster caching<br>Cache update<br>• On-demand |
| Mirror Image | Content selection<br>• Partial-site delivery<br>Content Clustering<br>• URL based | Multi-ISP approach; Center placement following a concentrated "Superstore" architecture | Non-cooperative pull-based | Caching technique<br>• Intra-cluster caching<br>Cache update<br>• On-demand |

**Table 2.2** (continued)

| CDN Name | Content Selection and Delivery | Surrogate Placement | Content Outsourcing | Cache Organization |
|---|---|---|---|---|
| CoDeeN | Content selection<br>• Partial-site delivery<br>Content Clustering<br>N/A | Multi-ISP approach;<br>Topology-informed replica placement | Cooperative pull-based | Caching technique<br>• Intra and inter-cluster caching<br>Cache update<br>• On-demand |
| Coral | Content selection<br>• Full and partial-site delivery<br>Content Clustering<br>• Users' sessions based | Multi-ISP approach; Tree-based replica placement | Cooperative pull-based | Caching technique<br>• Intra and inter-cluster caching<br>Cache update<br>• Cache invalidation |
| Globule | Content selection<br>• Full and partial-site delivery<br>Content Clustering<br>N/A | Single-ISP approach; Best replica placement strategy is dynamically selected through regular evaluation of different strategies | Cooperative pull-based | Caching technique<br>• Intra and inter-cluster caching<br>Cache update<br>• Adaptive cache update |

DNS as a directory service. DNS-based mechanisms mainly consist of using a specialized DNS server in the name resolution process. Among other request-routing mechanisms, HTTP redirection is also highly used in the CDNs because of the finer level of granularity on the cost of introducing an explicit binding between a client and a replica server. Flexibility and simplicity are other reasons of using HTTP redirection for request-routing in CDNs. Some CDNs such as Mirror Image uses GSLB for request-routing. It is advantageous since less effort is required to add GSLB capability to the network without adding any additional network devices. Among the academic CDNs, Coral exploits overlay routing techniques, where indexing abstraction for request-routing is done using DSHT. Thus, it makes use of P2P mechanism for request redirection. As we mentioned earlier, the request-routing system of a CDN is composed of a request-routing algorithm and a request-routing mechanism. The request-routing algorithms used by the CDNs are proprietary in nature. The technology details of most of them have not been revealed. Our analysis of the existing CDNs indicates that Akamai and Globule use adaptive request-routing algorithm for their request-routing system. Akamai's adaptive (to flash crowds) request-routing takes into account server load and various network metrics; whereas Globule measures only the number of AS that a request needs to pass through. In case of CoDeeN, the request-routing algorithm takes into account request locality, system load, reliability, and proximity information. On the other hand, Coral's request-routing algorithm improves locality by exploiting on-the-fly network measurement and storing topology hints in order to increase the possibility for the clients to discover nearby DNS servers.

**Table 2.3** Request-routing taxonomy mapping

| CDN Name | Request-routing Technique |
|---|---|
| Akamai | • Adaptive request-routing algorithms which takes into account server load and various network metrics<br>• Combination of DNS-based request-routing and URL rewriting |
| EdgeStream | HTTP redirection |
| Limelight Networks | DNS-based request-routing |
| Mirror Image | Global Server Load Balancing (GSLB)<br>• Global awareness<br>• Smart authoritative DNS |
| CoDeeN | • Request-routing algorithm takes into account request locality, system load, reliability, and proximity information.<br>• HTTP redirection. |
| Coral | • Request-routing algorithms with improved locality by exploiting on-the-fly network measurement and storing topology hints<br>• DNS-based request-routing |
| Globule | • Adaptive request-routing algorithms considering AS-based proximity<br>• Single-tier DNS-based request-routing |

### 2.3.4 Performance Measurement Taxonomy Mapping

Table 2.4 shows the mapping of different performance measurement techniques to representative CDNs.

Performance measurement of a CDN through some metric estimation measures its ability to serve the customers with the desired content and/or services. A CDN's performance should be evaluated in terms of cache hit ratio, bandwidth consumption, latency, surrogate server utilization, and reliability. In addition, other factors such as storage, communication overhead, and scalability can also be taken into account. The estimation of performance metrics gives an indication of system conditions and helps for efficient request-routing and load balancing in large systems. It is important to a content provider to conduct performance study of a CDN for selecting the most appropriate CDN provider. However, the proprietary nature of the CDN providers does not allow a content provider to conduct performance measurement on them.

From Table 2.4, we can see that performance measurement of a CDN is done through internal measurement technologies as well as from the customer perspective. It is evident that, most of the CDNs use internal measurement based on network probing, traffic monitoring or the like. Akamai uses proactive traffic monitoring and network probing for measuring performance. In the academic domain, CoDeeN has the local monitoring ability that examines a service's primary resources, such as free file descriptors/sockets, CPU cycles, and DNS resolver service; Coral has the ability

**Table 2.4** Performance measurement taxonomy mapping

| CDN Name | Performance Measurement |
|---|---|
| Akamai | Internal measurement<br>• Network probing<br>• Traffic monitoring (proactive)<br>External measurement<br>• Performed by a third party (Giga Information group) |
| EdgeStream | Internal measurement<br>• Traffic monitoring through Real Time Performance Monitoring Service (RPMS) |
| Limelight Networks | N/A |
| Mirror Image | Internal measurement<br>• Network probing<br>• Traffic monitoring and reporting |
| CoDeeN | Internal measurement<br>• Local traffic and system monitoring |
| Coral | Internal measurement<br>• Traffic monitoring<br>• Liveness checking of a proxy via UDP RPC |
| Globule | Internal measurement<br>• Traffic monitoring<br>• Monitoring of server availability by the redirectors |

to perform a proxy's liveness check (via UDP remote procedure call (RPC)) prior to replying to a DNS query; whereas, Globule has monitoring ability implemented in its redirector servers which checks for the availability of other servers.

External performance measurement of CDN providers is not common because most of the operating CDNs are commercial enterprises, which are not run transparently, and there are commercial advantages to keep the performance metrics and methodologies undisclosed. Despite this, some CDNs such as Akamai allow a third-party to perform external measurements.

## 2.4 Discussion

As stated at the beginning of this chapter, a full-fledged CDN development requires addressing additional issues (other than the four core issues considered for the taxonomy) such as fault tolerance, security, and ability for Web application hosting. In this section, we present a brief discussion on them and assist the readers to comprehend respective fields by providing referral to relevant research materials.

CDNs being a complex fabric of distributed network elements, failures can occur at many places. Following a concentrated architecture such as local clustering may improve fault-tolerance. However, it creates a single-point of failure, when the ISP connectivity to the cluster is lost. This problem can be solved through deploying Web clusters in distributed locations (mirroring) or using multiple ISPs to provide connectivity (multihoming). While clustering, mirroring, or multihoming addresses the CDN robustness issue to some extent, they introduce additional problems. Clustering suffers from scalability, while mirroring requires each mirror to carry entire load, and multihoming requires each connection to carry the entire traffic. Commercial CDNs follow their own proprietary approaches to provide fault-tolerance and scalability. As for instance, Akamai developed a distributed monitoring service that ensures that server or network failures are handled immediately without affecting the end users. Other than this, there are numerous solutions available in literature, some of which are widely used in real systems. Interested readers are referred to [46, 77, 85] to find descriptions on the explicit fault-tolerance solutions in wide-area systems such as CDNs.

Ensuring security in CDNs pose extra challenges in system development. There are security concerns at different levels of a CDN such as network, routers, DNS or Web clusters. One common security threat is the DDoS attack. The DDoS attack can be aimed at (a) consumption of scarce resources such as network bandwidth or CPU; (b) destruction or modification of configuration information; and (c) physical destruction or modifications of network components [82]. Security threats also include attacks which exploit software vulnerabilities (intrusion attacks) and protocol inconsistencies (protocol attacks). There exist many prior works addressing various wide-area security problems. Extensive coverage and documentation of security related solutions are available in the literature [40, 50, 51, 53, 56, 57, 82].

Nowadays, commercial CDNs such as Akamai provide usage-based content and application delivery solutions to the end users. Akamai Edge Computing

Infrastructure (ECI) [1], Active Cache [19], and ACDN [80] replicate the application code to the edge servers without replicating the application data itself. Rather, the data is kept in a centralized server. It enables the Web tier applications to extend to the CDN platform so that end user requests for application object would execute at the replica server rather than at the origin. However, this approach suffers from increased wide-area latency due to excessive data access and bottleneck due to the concentration on a centralized server. To overcome these limitations, Sivasubramanian et al. [87] propose an approach for replicating Web applications on-demand. This approach employs partial replication to replicate data units only to the servers who access them often. In another work, application specific edge service architecture [39] is presented where the application itself is responsible for its replication with the compromise of a weaker consistency model. For more information on hosting wide-area applications readers are referred to [88].

## 2.5 Summary and Conclusions

In this chapter, we have analyzed and categorized CDNs according to the functional and non-functional attributes. We have developed a comprehensive taxonomy for CDNs based on four issues: CDN composition, content distribution and management, request-routing, and performance measurement. We further built up taxonomies for each of these paradigms to classify the common trends, solutions, and techniques in content networking. Additionally, we identify three issues, namely, fault tolerance, security, and ability for Web application hosting as to introduce challenges in CDN development. Hereby, we provided pointers to related research work in this context. Our taxonomy provides a basis for comparison of existing CDNs. In doing so, we assist the readers to gain insights into the technology, services, strategies, and practices that are currently followed in this field. We have also performed a mapping of the taxonomy to representative commercial and academic CDN systems. Such a mapping provides a basis to realize an in-depth understanding of the state-of-the-art technologies in content distribution space, and to validate the applicability and accuracy of the taxonomy.

Recently, the CDN industry is getting consolidated as a result of acquisitions and/or mergers. During the preparation of this chapter, we have experienced significant changes in the content distribution landscape due to this consolidation. Consequently, content distribution, caching, and replication techniques are gaining more attention in order to meet up the new technical and infrastructure requirements for the next generation CDNs. This may lead to new issues in the design, architecture, and development of CDNs. Present trends in content networking domain indicate that better understanding and interpretation of the essential concepts in this area is necessary. Therefore, we hope that the comprehensive comparison framework based on our taxonomy, presented in this chapter, will not only serve as a tool to understand this complex area, but also will help to map the future research efforts in content networking.

**Acknowledgements** We would like to acknowledge the efforts of all the developers of the commercial and academic CDNs surveyed in this paper. We thank the anonymous reviewers for their insightful comments and suggestions that have improved the presentation and correctness of this chapter. We also thank our colleagues at the University of Melbourne – James Broberg, Marcos Assunção, and Charity Lourdes for sharing thoughts and for making incisive comments and suggestions on this chapter. We would like to express our gratitude to Athena Vakali (Aristotle University of Thessaloniki, Greece), George Pallis (The University of Cyprus, Cyprus), Carlo Mastroianni (ICAR-CNR, Italy), Giancarlo Fortino (Università della Calabria, Italy), Christian Vecchiola (University of Genova, Italy), and Vivek Pai (Princeton University, USA) for their visionary comments on various parts of the taxonomy. We are also thankful to Fahim Husain (Akamai Technologies, Inc., USA), William Good (Mirror Image Internet, Inc., USA), and Lisa Amini (IBM T. J. Watson Research Center, USA) for providing valuable research papers, technical reports, white papers, and data sheet while preparing the manuscript.

## References

1. Akamai Technologies, 2007. www.akamai.com
2. CDNSim, A Content Distribution Network Simulator, 2007. http://oswinds.csd.auth.gr/∼cdnsim/
3. Keynote Systems—Web and Mobile Service Performance Testing Corporation, 2007. http://www.keynote.com/
4. Network simulators, 2007. http://www-nrg.ee.lbl.gov/kfall/netsims.html
5. PlanetLab Consortium, 2007. http://www.planet-lab.org/
6. The GigaWeb Corporation, 2007. http://www.gigaWeb.com/
7. The network simulator – ns-2, 2007. http://www.isi.edu/nsnam/ns/
8. Aberer, K. and Hauswirth, M. An overview on peer-to-peer information systems. In *Proc. of the Workshop on Distributed Data and Structures (WDAS)*, France, 2002.
9. Aggarwal, A. and Rabinovich, M. Performance of dynamic replication schemes for an Internet hosting service. Technical Report, HA6177000-981030-01-TM, AT&T Research Labs, Florham Park, NJ, USA, 1998.
10. Andrews, M., Shepherd, B., Srinivasan, A., Winkler, P., and Zane, F. Clustering and server selection using passive monitoring. In *Proc. of IEEE INFOCOM*, NY, USA, 2002.
11. Androutsellis-Theotokis, S. and Spinellis, D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4), ACM Press, NY, USA, pp. 335–371, 2004.
12. Ardaiz, O., Freitag, F., and Navarro, L. Improving the service time of Web clients using server redirection. *ACM SIGMETRICS Performance Evaluation Review*, 29(2), ACM Press, NY, USA, pp. 39–44, 2001.
13. Bakiras, S. and Loukopoulos, T. Combining replica placement and caching techniques in content distribution networks. *Computer Communications*, 28(9), pp. 1062–1073, 2005.
14. Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. Looking up data in P2P systems. *Communications of the ACM*, 46(2), ACM Press, NY, USA, pp. 43–48, 2003.
15. Barbir, A., Cain, B., Nair, R., and Spatscheck, O. Known content network request-routing mechanisms. Internet Engineering Task Force RFC 3568, 2003. www.ietf.org/rfc/rfc3568.txt
16. Bartal, Y. Probabilistic approximation of metric space and its algorithmic applications. In *Proc. of 37th Annual IEEE Symposium on Foundations of Computer Science*, 1996.
17. Brussee, R., Eertink, H., Huijsen, W., Hulsebosch, B., Rougoor, M., Teeuw, W., Wibbels, M., and Zandbelt, H. Content distribution network state of the art," Telematica Instituut, 2001.
18. Byers, J., Considine, J., and Mitzenmacher, J. Simple load balancing for distributed hash tables. In *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pp. 31–35, 2003.
19. Cao, P., Zhang, J., and Beach, K. Active cache: Caching dynamic contents on the Web. In *Proc. of the Middleware Conference*, pp. 373–388, 1998.

20. Cardellini, V., Casalicchio, E., Colajanni, M., and Yu, P. S. The state of the art in locally distributed Web-server systems. *ACM Computing Surveys*, 34(2), ACM Press, NY, USA, pp. 263–311, 2002.
21. Chen, Y., Katz, R. H., and Kubiatowicz, J. D. Dynamic replica placement for scalable content delivery. In *Proc. of International Workshop on Peer-to-Peer Systems (IPTPS 02)*, LNCS 2429, Springer-Verlag, pp. 306–318, 2002.
22. Chen, C. M., Ling, Y., Pang, M., Chen, W., Cai, S., Suwa, Y., Altintas, O. Scalable request-routing with next-neighbor load sharing in multi-server environments. In *Proc. of the 19th International Conference on Advanced Information Networking and Applications*, IEEE Computer Society, Washington, DC, USA, pp. 441–446, 2005.
23. Chen, Y., Qiu, L., Chen, W., Nguyen, L., and Katz, R. H. Efficient and adaptive Web replication using content clustering. *IEEE Journal on Selected Areas in Communications*, 21(6), pp. 979–994, 2003.
24. Cieslak, M., Foster, D., Tiwana, G., and Wilson, R. Web cache coordination protocol version 2. http://www.Web-cache.com/Writings/Internet-Drafts/draft-wilson-wrec-wccp-v2-00.txt
25. Clip2 Distributed Search Solutions, The Gnutella Protocol Specification v0.4. www.content-networking.com/papers/gnutella-protocol-04.pdf
26. Cooper, I., Melve, I., and Tomlinson, G. Internet Web replication and caching taxonomy. Internet Engineering Task Force RFC 3040, 2001. www.ietf.org/rfc/rfc3040.txt
27. Davison, B. D. Web caching and content delivery resources. http://www.Web-caching.com, 2007.
28. Delgadillo, K. Cisco DistributedDirector, Cisco Systems, Inc., 1997.
29. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing*, pp. 50–58, 2002.
30. Douglis, F. and Kaashoek, M. F. Scalable Internet services. *IEEE Internet Computing*, 5(4), pp. 36–37, 2001.
31. Emtage, A. and Deutsch, P. Archie: an electronic directory service for the Internet. In *Proc. of the Winter Usenix Conference*, San Francisco, CA, USA, pp. 93–110, January 1992.
32. Fei, Z., Bhattacharjee, S., Zugura, E. W., and Ammar, M. H. A novel server selection technique for improving the response time of a replicated service. In *Proc. of IEEE INFOCOM*, San Francisco, California, USA, pp. 783–791, 1998.
33. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. IDMaps: a global Internet host distance estimation service. *IEEE/ACM Transactions on Networking (TON)*, 9(5), ACM Press, NY, USA, pp. 525–540, 2001.
34. Freedman, M. J., Freudenthal, E., and Mazières, D. Democratizing content publication with Coral. In *Proc. of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, San Francisco, CA, USA, 2004.
35. Freedman, M. J., Lakshminarayanan, K., and Mazières, K. OASIS: anycast for any service. In *Proc. of 3rd Symposium of Networked Systems Design and Implementation (NSDI'06)*, Boston, MA, USA, 2006.
36. Fujita, N., Ishikawa, Y., Iwata, A., and Izmailov, R. Coarse-grain replica management strategies for dynamic replication of Web contents. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 45(1), pp. 19–34, 2004.
37. Gadde, S., Chase, J., and Rabinovich, M. Web caching and content distribution: a view from the interior. *Computer Communications*, 24(2), pp. 222–231, 2001.
38. Gadde, S., Rabinovich, M., and Chase, J. Reduce, reuse, recycle: an approach to building large Internet caches. In *Proc. of 6th Workshop on Hot Topics in Operating Systems*, pp. 93–98, 1997.
39. Gao, L., Dahlin, M., Nayate, A., Zheng, J., and Iyengar, A. Application specific data replication for edge services. In *Proc. of the Twelfth International World-Wide Web Conference*, Hungary, pp. 449–460, 2003.
40. Garg, A. and Reddy, A. L. N. Mitigating denial of service attacks using qos regulation. In *Proc. of International Workshop on Quality of Service (IWQoS)*, 2002.
41. Gayek, P., Nesbitt, R., Pearthree, H., Shaikh, A., and Snitzer, B. A Web content serving utility. *IBM Systems Journal*, 43(1), pp. 43–63, 2004.

42. Hamilton, M., Rousskov, A., and Wessels, D. Cache digest specification – version 5. 1998. http://www.squid-cache.org/CacheDigest/cache-digest-v5.txt
43. Harren, M., Hellerstein, J. M., Huebsch, R., Loo, B. T., Shenker, S., and Stoica, I. Complex queries in DHT-based peer-to-peer networks. In *Proc. of 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.
44. Hofmann, M. and Beaumont, L. R. *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129–134, 2005.
45. Huffaker, B., Fomenkov, M., Plummer, D. J., Moore, D., and Claffy, K. Distance metrics in the Internet. In *Proc. of IEEE International Telecommunications Symposium*, IEEE CS Press, Los Alamitos, CA, USA, 2002.
46. Jalote, P. *Fault Tolerance in Distributed Systems*. Prentice Hall, Englewood Cliffs, NJ, USA, 1994.
47. Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. On the placement of Internet instrumentation. In *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, pp. 295–304, 2000.
48. Jamin, S., Jin, C., Kure, A. R., Raz, D., and Shavitt, Y. Constrained mirror placement on the Internet. In *Proc. of IEEE INFOCOM*, Anchorage, Alaska, USA, 2001.
49. Johnson, K. L., Carr, J. F., Day, M. S., and Kaashoek, M. F. The measured performance of content distribution networks. *Computer Communications*, 24(2), pp. 202–206, 2001.
50. Jung, J., Krishnamurthy, B. and Rabinovich, M. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In *Proc. of the International World Wide Web Conference*, Hawaii, USA, pp. 252–262, 2002.
51. Jung, J., Paxson, V., Berger, A. W., and Balakrishnan, H. Fast portscan detection using sequential hypothesis testing. In *Proc. of IEEE Symposium on Security and Privacy*, Oakland, 2004.
52. Kahle, B. and Medlar, A. An information system for corporate users: wide area information servers. ConneXions—The Interoperability Report, 5(11), November 1991.
53. Kandula, S., Katabi, D., Jacob, M., and Berger, A. W. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *Proc. of Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, 2005.
54. Kangasharju, J., Roberts, J., and Ross, K. W. Object replication strategies in content distribution networks. *Computer Communications*, 25(4), pp. 367–383, 2002.
55. Karger, D., Sherman, A., Berkheimer, A., Bogstad, B., Dhanidina, R., Iwamoto, K., Kim, B., Matkins, L., and Yerushalmi, Y. Web caching with consistent hashing. *Computer Networks*, 31(11–16), pp. 1203–1213, 1999.
56. Kargl, F., Maier, J., and Weber, M. Protecting Web servers from distributed denial of service attacks, In *Proc. of the International World Wide Web Conference*, pages 514–524, Hong Kong, 2001.
57. Kim, Y., Lau, W. C., Chuah, M. C., and Chao, H. J. Packetscore: Statistics based overload control against distributed denial-of-service attacks. In *Proc. of INFOCOM*, Hong Kong, 2004.
58. Krishnamurthy, B., Willis, C., and Zhang, Y. On the use and performance of content distribution network. In *Proc. of 1st International Internet Measurement Workshop*, ACM Press, pp. 169–182, 2001.
59. Krishnan, P., Raz, D., and Shavitt, Y. The cache location problem. *IEEE/ACM Transaction on Networking*, 8(5), 2000.
60. Kung, H. T. and Wu, C. H. Content networks: taxonomy and new approaches. *The Internet as a Large-Scale Complex System*, (Kihong Park and Walter Willinger eds.), Oxford University Press, 2002.
61. Lazar, I. and Terrill, W. Exploring content delivery networking. *IT Professional*, 3(4), pp. 47–49, 2001.
62. Lee, J. An End-User Perspective on File-Sharing Systems. *Communications of the ACM*, 46(2), ACM Press, NY, USA, pp. 49–53, 2003.
63. Li, B., Golin, M. J., Italiano, G. F., Xin, D., and Sohraby, K. On the optimal placement of Web proxies in the Internet. In *Proc. of IEEE INFOCOM*, NY, USA, pp. 1282–1290, 1999.

64. Ma, W. Y., Shen, B., and Brassil, J. T. Content services network: architecture and protocols. In *Proc. of 6th International Workshop on Web Caching and Content Distribution (IWCW6)*, 2001.
65. Mao, Z. M., Cranor, C. D., Douglis, F., Rabinovich, M., Spatscheck, O., and Wang, J. A precise and efficient evaluation of the proximity between Web clients and their Local DNS servers. In *Proc. of the USENIX 2002 Annual Technical Conference*, Monterey, CA, USA, pp. 229–242, 2002.
66. Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., and Xu, Z. Peer-to-peer computing. Technical Report, HP Laboratories, Palo Alto, CA, HPL-2002-57, 2002. www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf
67. Ni, J. and Tsang, D. H. K. Large scale cooperative caching and application-level multicast in multimedia content delivery networks. *IEEE Communications*, 43(5), pp. 98–105, 2005.
68. Ni, J., Tsang, D. H. K., Yeung, I. S. H., and Hei, X. Hierarchical content routing in large-scale multimedia content delivery network. In *Proc. of IEEE International Conference on Communications (ICC)*, pp. 854–859, 2003.
69. Pai, V. S., Aron, M., Banga, G., Svendsen, M., Druschel, P., Zwaenepoel, W., Nahum, E. Locality-aware request distribution in cluster-based network servers. *ACM SIGPLAN Notices*, 33(11), ACM Press, NY, USA, pp. 205–216, 1998.
70. Pallis, G., Stamos, K., Vakali, A., Sidiropoulos, A., Katsaros, D., and Manolopoulos, Y. Replication-based on objects load under a content distribution network. In *Proc. of the 2nd International Workshop on Challenges in Web Information Retrieval and Integration (WIRI)*, Altanta, Georgia, USA, 2006.
71. Pallis, G. and Vakali, A. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1), ACM Press, NY, USA, pp. 101–106, 2006.
72. Pallis, G., Vakali, A., Stamos, K., Sidiropoulos, A., Katsaros, D., and Manolopoulos, Y. A latency-based object placement approach in content distribution networks. In *Proc. of the 3rd Latin American Web Congress (La-Web 2005)*, IEEE Press, Buenos Aires, Argentina, pp. 140–147, 2005.
73. Partridge, C., Mendez, T., and Milliken, W. Host anycasting service. Internet Engineering Task Force RFC 1546, 1993. www.ietf.org/rfc/rfc1546.txt
74. Pathan, M., Broberg, J., Bubendorfer, K., Kim, K. H., and Buyya, R. An Architecture for Virtual Organization (VO)-Based Effective Peering of Content Delivery Networks, UPGRADE-CN'07. In *Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, Monterey, California, USA, 2007.
75. Peng, G. CDN: Content distribution network. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003. http://citeseer.ist.psu.edu/peng03cdn.html
76. Pierre, G. and van Steen, M. Globule: a collaborative content delivery network. *IEEE Communications*, 44(8), 2006.
77. Pradhan, D. *Fault-Tolerant Computer System Design*. Prentice Hall, Englewood Cliffs, NJ, USA, 1996.
78. Qiu, L., Padmanabhan, V. N., and Voelker, G. M. On the placement of Web server replicas. In *Proc. of IEEE INFOCOM*, Anchorage, Alaska, USA, pp. 1587–1596, 2001.
79. Rabinovich, M. and Spatscheck, O. *Web Caching and Replication*, Addison Wesley, USA, 2002.
80. Rabinovich, M., Xiao, Z., and Agarwal, A. Computing on the edge: A platform for replicating internet applications. In *Proc. of the Eighth International Workshop on Web Content Caching and Distribution*, Hawthorne, NY, USA, 2003.
81. Radoslavov, P., Govindan, R., and Estrin, D. Topology-informed Internet replica placement. In *Proc. of Sixth International Workshop on Web Caching and Content Distribution*, Boston, Massachusetts, 2001.
82. Ranjan, S., Swaminathan, R., Uysal, M., and Knightly, E. DDoS-Resilient scheduling to counter application layer attacks under Imperfect Detection. In *Proc. of INFOCOM*, pp. 1–13, 2006

83. Rousskov, A. and Wessels, D. Cache digests. *Computer Networks and ISDN Systems*, 30(22), pp. 2155–2168, November 1998.
84. Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. An analysis of Internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36, pp. 315–328, 2002.
85. Schneider, F. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial, 1 *ACM Computing Surveys*, 22(4), pp.299–320, 1990.
86. Shaikh, A., Tewari, R., and Agrawal, M. On the effectiveness of DNS-based server selection." In *Proceedings of IEEE INFOCOM*, Anchorage, AK, USA, pp. 1801–1810, April 2001.
87. Sivasubramanian, S., Pierre, G., and van Steen, M. Replicating Web applications on-demand. In *Proc. of IEEE International Conference on Services Computing (SCC)*, pp. 227–236, China, 2004.
88. Sivasubramanian, S., Pierre, G., van Steen, M., and Alonso, G. Analysis of caching and replication strategies for Web applications. *IEEE Internet Computing*, 11(1), pp. 60–66, 2007.
89. Sivasubramanian, S., Szymaniak, M., Pierre, G., and Van Steen, M. Replication of Web hosting systems. *ACM Computing Surveys*, 36(3), ACM Press, NY, USA, 2004.
90. Stamos, K., Pallis, G., Thomos, C., and Vakali, A. A similarity-based approach for integrated Web caching and content replication in CDNs. In *Proc. of 10th International Databased Engineering and Applications Symposium (IDEAS 2006)*, IEEE Press, New Delhi, India, 2006.
91. Stamos, K., Pallis, G., and Vakali, A. Integrating caching techniques on a content distribution network. In *Proc. of 10th East-European Conference on Advances in Databases and Information Systems (ADBIS 2006)*, Springer-Verlag, Thessaloniki, Greece, 2006.
92. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. Chord: a scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Transactions on Networking (TON)*, 11(1), ACM Press, NY, USA, pp. 17–32, 2003.
93. Szymaniak, M., Pierre, G., and van Steen, M. Netairt: a DNS-based redirection system for apache. In *Proc. of International Conference WWW/Internet*, Algrave, Portugal, 2003.
94. Tse, S. S. H. Approximate algorithms for document placement in distributed Web servers. *IEEE Transactions on Parallel and Distributed Systems*, 16(6), pp. 489–496, 2005.
95. Vakali, A. and Pallis, G. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6), IEEE Computer Society, pp. 68–74, 2003.
96. Valloppillil, V. and Ross, K. W. Cache array routing protocol v1.0. Internet Draft, 1998.
97. Verma, D. C. *Content Distribution Networks: An Engineering Approach*, John Wiley & Sons, Inc., New York, 2002.
98. Vixie, P. and Wessels, D. Hyper text caching protocol (HTCP/0.0). Internet Engineering Task Force RFC 2756, 2000. www.ietf.org/rfc/rfc2756.txt
99. Wang, J. A survey of Web caching schemes for the Internet. *SIGCOMM Computer Communication Review*, 29(5), ACM Press, NY, USA, pp. 36–46, 1999.
100. Wang, L., Pai, V. S., and Peterson, L. The effectiveness of request redirection on CDN robustness. In *Proc. of 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, USA, pp. 345–360, 2002.
101. Wessels, D. and Claffy, K. Internet cache protocol (ICP) version 2. Internet Engineering Task Force RFC 2186, 1997. www.ietf.org/rfc/rfc2186.txt
102. Wu, B. and Kshemkalyani, A. D. Objective-optimal algorithms for long-term Web Prefetching. *IEEE Transactions on Computers*, 55(1), pp. 2–17, 2006.

# Chapter 3
# Dynamic, Scalable, and Efficient Content Replication Techniques

Yan Chen

## 3.1 Introduction

Exponential growth in processor performance, storage capacity, and network bandwidth is changing our view of computing. Our focus has shifted away from centralized, hand-choreographed systems to global-scale, distributed, self-organizing complexes – composed of thousands or millions of elements. Unfortunately, large pervasive systems are likely to have frequent component failures and be easily partitioned by slow or failed network links. Thus, use of local resources is extremely important – both for performance *and* availability. Further, pervasive streaming applications must tune their communication structure to avoid excess resource usage. To achieve both local access *and* efficient communication, we require flexibility in the placement of data replicas and multicast nodes.

One approach for achieving this flexibility while retaining strong properties of the data is to partition the system into two tiers of replicas [18] – a small, durable *primary* tier and a large, soft-state, *second-tier*. The primary tier could represent a Web server (for Web content delivery), the Byzantine inner ring of a storage system [6, 29], or a streaming media provider. The important aspect of the primary tier is that it must hold the most up-to-date copy of data and be responsible for serializing and committing updates. We will treat the primary tier as a black box, called simply "the data source". The second-tier becomes soft-state and will be the focus of this chapter. Examples of second-tiers include Content Delivery Networks (CDNs), file system caches, or Web proxy caches.

Because second-tier replicas (or just "replicas") are soft-state, we can dynamically grow and shrink their numbers to meet constraints of the system. We may, for instance, wish to achieve a Quality of Service (QoS) guarantee that bounds the maximum network latency between each client and replicas of the data that it is accessing. Since replicas consume resources, we will seek to generate as few replicas as possible to meet this constraint. As a consequence, popular data items may

Yan Chen
Department of EECS, Northwestern University, Evanston IL, USA,
e-mail: ychen@northwestern.edu

R. Buyya et al. (eds.), *Content Delivery Networks*,
© Springer-Verlag Berlin Heidelberg 2008

warrant hundreds or thousands of replicas, while unpopular items may require no replicas.

One difficult aspect of unconstrained replication is ensuring that content does not become stale. Slightly relaxed consistency, such as in the Web [20], OceanStore [29], or Coda [26], allows delay between the commitment of updates at the data source and the propagation of updates to replicas. None-the-less, update propagation must still occur in a timely manner. The potentially large number of replicas rules out direct, point-to-point delivery of updates to replicas. In fact, the extremely fluid nature of the second tier suggests a need to self-organize replicas into a multicast tree; we call such a tree a *dissemination tree* (d-tree). Since interior nodes must forward updates to child nodes, we will seek to control the *load* placed on such nodes by restricting the fanout of the tree.

The challenge of second-tier replication is to provide good QoS to clients while retaining *efficient* and *balanced* resource consumption of the underlying infrastructure. To tackle this challenge, we propose a self-organizing soft-state replication system called SCAN: the *Scalable Content Access Network*. Figure 3.1 illustrates a SCAN system. There are two classes of physical nodes shown in the network-plane of this diagram: *SCAN servers* (squares) and *clients* (circles). We assume that SCAN servers are placed in Internet Data Centers (IDC) of major ISPs with good connectivity to the backbone. Each SCAN server may contain replicas for a variety of data items. One novel aspect of the SCAN system is that it assumes SCAN servers participate in a distributed routing and location (DOLR) system, called Tapestry [22]. Tapestry permits clients to locate nearby replicas without global communication.

There are three types of data illustrated in Fig. 3.1: Data *sources* and *replicas* are the primary topic of this chapter and reside on SCAN servers. *Caches* are the images

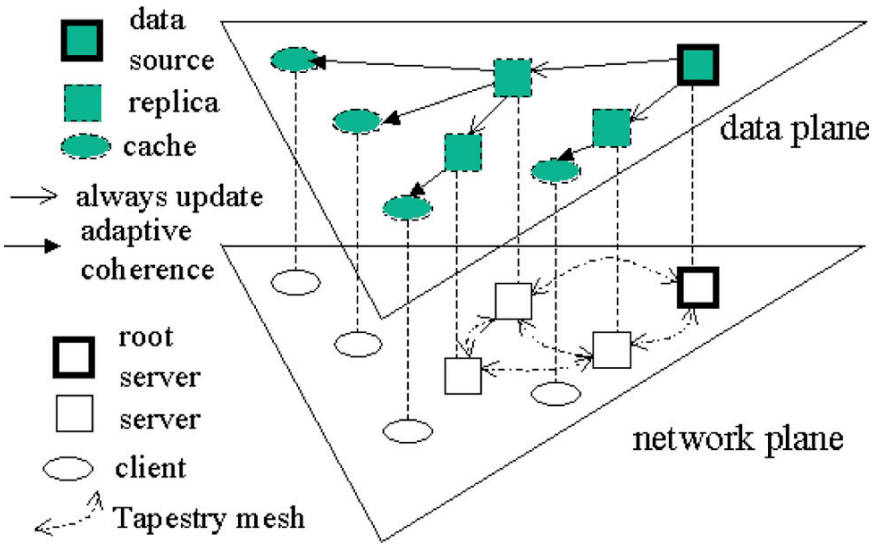



**Fig. 3.1** Architecture of a SCAN system

of data that reside on clients and are beyond our scope[1] Our goal is to translate client requests for data into replica management activities. We make the following contributions:

- We provide algorithms that dynamically place a minimal number of replicas while meeting client QoS and server capacity constraints.
- We self-organize these replicas into d-tree with small delay and bandwidth consumption for update dissemination.

The important intuition here is that the presence of the DOLR system enables simultaneous placement of replicas and construction of a dissemination tree without contacting the data source. As a result, each node in a d-tree must maintain state only for its parent and direct children.

The rest of this chapter is organized as follows. We first examine the related work in Sect. 3.2, then formulate the replica placement problem in Sect. 3.3. Next, we present our algorithms in Sect. 3.4, evaluation methodology in Sect. 3.5 and evaluation results in Sect. 3.6.

## 3.2 Previous Work

In this section, we first survey existing content distribution systems, namely Web caching (Sect. 3.2.1), uncooperative pull-based CDNs (Sect. 3.2.2), and cooperative push-based CDNs (Sect. 3.2.3). We compare these systems with SCAN, and summarize this in Table 3.1. Then we discuss the previous work on three building blocks of CDN: object location services (Sect. 3.2.4), and multicast techniques for update dissemination (Sect. 3.2.5). Finally, we summarize the limitations of previous work in Sect. 3.2.6.

### *3.2.1 Web Caching*

Caching can be *client-initiated* or *server-initiated*. Most caching schemes in wide-area, distributed systems are *client-initiated*, such as used by current Web browsers and Web proxies [32]. The problems with both of these solutions are myopic. A client cache does nothing to reduce traffic to a neighboring computer, and a Web proxy does not help neighboring proxies. Thus, the effectiveness of caching is ultimately limited to the low level of sharing of remote documents among clients of the same site [4]. A possible solution, *server-initiated caching*, allows servers to determine when and where to distribute objects [3, 4, 21]. Essentially, CDNs (including our approach) are server-initiated caching *with dedicated edge servers*. Previous server-initiated caching systems rely on unrealistic assumptions. Bestavros et al.

[1] Caches may be kept coherent in a variety of ways (for instance [44]).

**Table 3.1** Comparison of various Internet content delivery systems

| Properties | Web Caching (Client Initiated) | Web Caching (Server Initiated) | Uncooperative Pull-Based CDNs | Cooperative Push-Based CDNs | **SCAN** |
|---|---|---|---|---|---|
| Cache/replica sharing for efficient replication | No, uncooperative | Yes, cooperative | No, uncooperative | Yes, cooperative | **Yes, cooperative** |
| Scalability for request redirection | No redirection | OK, use Bloom filter [15] to exchange replica locations | Bad, centralized CDN name server | Bad, centralized CDN name server | **Good, decentralized DHT location services** |
| Granularity of replication | Per URL | Per URL | Per URL | Per Website | **Per cluster** |
| Distributed load balancing | No | No | Yes | No | **Yes** |
| Replica coherence | No | No | No | No | **Yes** |
| Network monitoring for fault-tolerance | No | No | Yes, but unscalable monitoring | No | **Yes, scalable monitoring** |

model the Internet as a hierarchy and any internal node is available as a service proxy [3, 4]. This assumption is not valid because internal nodes are routers, unlikely to be available as service proxies. Geographical push-caching autonomously replicate HTML pages based on the global knowledge of the network topology and clients' access patterns [21]. More recently, adaptive web caching [34] and summary cache [15] are proposed to enable the sharing of caches among Web proxies. Caches exchange content state periodically with other caches, eliminating the delay and unnecessary use of resources of explicit cache probing. However, each proxy server needs to send index update of cached contents to *all* other proxy servers, and needs to store the content indices of *all* other proxy servers. Thus, even with compact content index summary like the Bloom filter [15], the state maintenance and exchange overhead is still overwhelming and unscalable with the number of documents and number of cache servers. For instance, the target number of proxy servers is only in the order of 100 [15]. Furthermore, without dedicated infrastructure like CDN, caching proxies can not adapt to network congestion/failures or provide distributed load balancing.

### *3.2.2 Un-Cooperative Pull-Based CDNs*

Recently, CDNs have been commercialized to provide Web hosting, Internet content and streaming media delivery. Basically, the contents are pulled to the edge servers upon clients' requests. Various mechanisms, such as DNS-based redirection, URL rewriting, HTTP redirection, etc. [1], have been proposed to direct client requests for objects to one of the CDN servers (*a. k. a.* CDN nodes or edge servers). Most of the commercial CDN providers, such as Akamai [14], LimeLight Networks [31], and Mirror Image [35], use DNS-based redirection due to its transparency [28]. Figure 3.2 shows the CDN architecture using DNS-based redirection. Given the rapid growth of CDN service providers, such as Akamai (which already has more than 25,000 servers in about 900 networks spanning across 69 countries [14]), we assume that for each popular clients cluster, there is a CDN server as well as a local DNS server. The client cluster is the group of clients that are topologically close. The clients can be grouped by their BGP prefix [27] or by their local DNS servers. The latter is simple and adopted in practice, but it is not very accurate [33].

Figure 3.2 gives the sequence of operations for a client to retrieve a URL. The hostname resolution request is sent to the CDN name server via local DNS server. Due to the nature of centralized location service, the CDN name server cannot afford to keep records for the locations of each URL replica. Thus it can only redirect the request based on network proximity, bandwidth availability and server load. The CDN server that gets the redirected request may not have the replica. In that case, it will pull a replica from the Web content server, then reply to the client.

Due to the uncooperative nature, current CDNs often places more replicas than necessary and consumes lots of resources for storage and update. Simulations reveal that with reasonable latency guarantees, cooperative push-based CDN (defined in

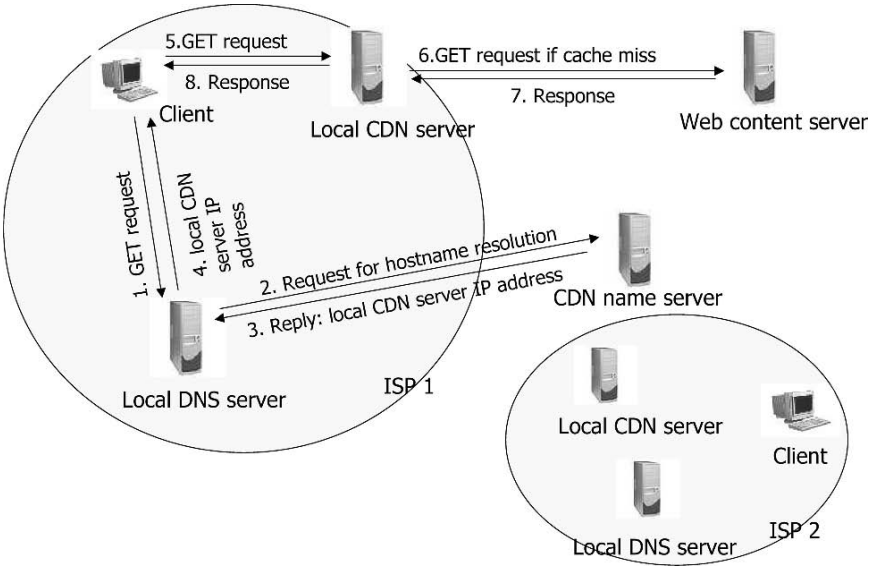



**Fig. 3.2** Un-cooperative pull-based CDN architecture

Sect. 3.2.3) only uses a small fractional number of replicas (6–8%) and less than 10% of the update dissemination bandwidth than the uncooperative schemes [10, 11].

As a research effort, Rabinovich and Aggarwal propose RaDaR, a global Web hosting service with dynamic content replication and migration [41]. However, it requires the DNS to give the complete path from the client to the server, which in practice is often unavailable.

### *3.2.3 Cooperative Push-Based CDNs*

Several recent works proposed to pro-actively push content from the origin Web server to the CDN edge servers or proxies according to users' access patterns and global network topology, and have the replicas cooperatively satisfy clients' requests [25, 30, 40, 48].

The key advantage of this cooperative push-based replication scheme over the conventional one does not come from the fact that we use push instead of pull (which only saves compulsory miss), but comes from the cooperative sharing of the replicas deployed. This cooperative sharing significantly reduces the number of replicas deployed, and consequently reduces the replication and update cost [10, 11].

We can adopt a similar CDN architecture as shown in Fig. 3.3 to support such a cooperative push-based content distribution. First, the Web content server incrementally pushes contents based on their hyperlink structures and/or some access history collected by CDN name server [10, 11]. The content server runs a "push"

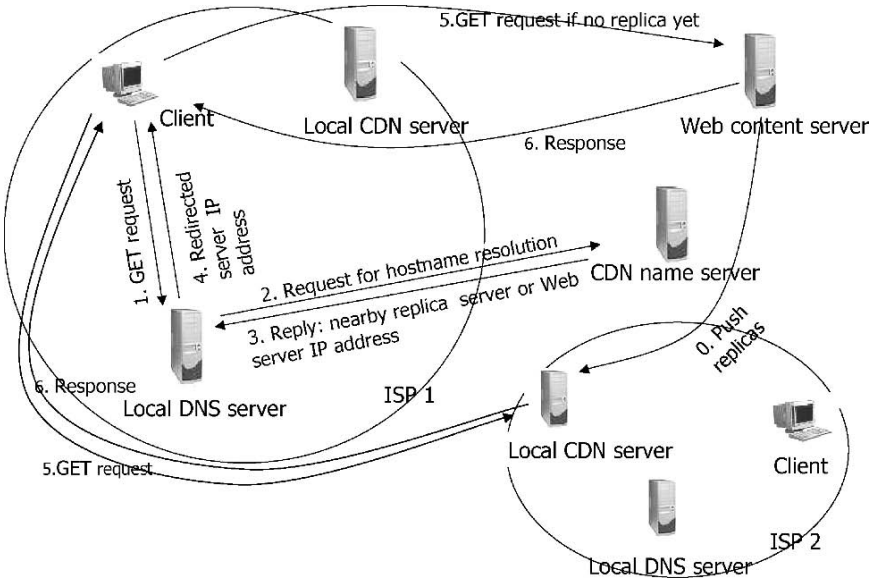



**Fig. 3.3** Cooperative push-based CDN architecture

daemon, and advertises the replication to the CDN name server, which maintains the mapping between content, identified by the host name in its (rewritten) URL, and their replica locations. The mapping can be coarse (e.g. at the level of Web sites if replication is done in units of Web sites), or fine-grained (e.g. at the level of URLs if replication is done in units of URLs).

With such replica location tracking, the CDN name server can redirect a client's request to its closest replica. Note that the DNS-based redirection allows address resolution on a per-host level. We combine it with content modification (e.g. URL rewriting) to achieve per-object redirection [1]. References to different objects are rewritten into different host names. To reduce the size of the domain name spaces, objects can be clustered as studied by Chen et al. [10, 11], and each cluster shares the same host name. Since the content provider can rewrite embedded URLs a priori before pushing out the objects, it does not affect the users' perceived latency and the one-time overhead is acceptable. In both models, the CDN edge servers are allowed to execute their cache replacement algorithms. That is, the mapping in cooperative push-based replication is soft-state. If the client cannot find the content in the redirected CDN edge server, either the client will ask the CDN name server for another replica, or the edge server pulls the content from the Web server and replies to the client.

Li et al. approach the proxy placement problem with the assumption that the underlying network topologies are trees, and model it as a dynamic programming problem [30]. While an interesting first step, this approach has an important limitation in that the Internet topology is not a tree. More recent studies [25, 40], based on evaluating real traces and topologies, have independently reported that a greedy placement algorithm can provide CDNs with performance that is close to optimal.

To simplify the assumption about detailed knowledge of global network topology and clients' distribution, topology-informed Internet replica placement was proposed to place replicas on the routers with big fanout [42]. They show that the router-level topology based replica placement can achieve average client latencies within a factor of 1.1–1.2 of the greedy algorithm, but only if the placement method is carefully designed.

## *3.2.4 Object Location Systems*

Networked applications are extending their reach to a variety of devices and services over the Internet. Applications expanding to leverage these network resources find that *locating objects* on the wide-area is an important problem. Further, the read-mostly model of shared access, widely popularized by the World-Wide-Web, has led to extensive object replication, compounding the problem of object location. Work on location services has been done in a variety of contexts [13, 19, 23, 50]. These approaches can be roughly categorized into the following three groups: *Centralized Directory Services (CDS)*, *Replicated Directory Services (RDS)*, and *Distributed Directory Services (DDS)*.

Extensive work on these directory services have been proposed as we will discuss in more detail in this subsection. However, to the best of our knowledge, there is no attempt to benchmark and contrast their performance.

### 3.2.4.1 Centralized and Replicated Directory Services

A *centralized directory service* (CDS) resides on a single server and provides location information for every object on the network (See Fig. 3.4). Because it resides on a single server, it is extremely vulnerable to DoS attacks. A variant of this is the *replicated directory service* (RDS) which provides multiple directory servers. An RDS provides higher availability, but suffers consistency overhead. Here we do not consider the partitioned directory service because it often requires extra meta directory server for maintaining the partitioning information, such as the root server of DNS.

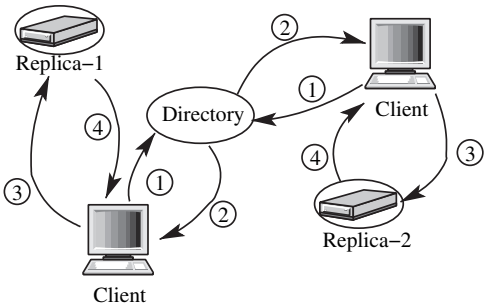



**Fig. 3.4** A Centralized Directory Service (CDS): Clients contact a single directory to discover the location of a close replica. Clients subsequently contact the replica directly. A Replicated Directory Service (RDS) provides multiple directories

### 3.2.4.2 Distributed Directory Services: The Tapestry Infrastructure

Networking researchers have begun to explore decentralized peer-to-peer location services with distributed hash table (DHT), such as CAN [43], Chord [47], Pastry [45] and Tapestry [50]. Such services offer a distributed infrastructure for locating objects quickly with guaranteed success. Rather than depending on a single server to locate an object, a query in this model is passed around the network until it reaches a node that knows the location of the requested object. The lack of a single target in decentralized location services means they provide very high availability even under attack; the effects of successfully attacking and disabling a set of nodes is limited to a small set of objects.

In addition, Tapestry exploits locality in routing messages to mobile endpoints such as object replicas; this behavior is in contrast to other structured peer-to-peer overlay networks [43, 45, 47]. Thus we leverage on Tapestry to build SCAN.

Tapestry is an IP overlay network that uses a distributed, fault-tolerant architecture to track the location of objects in the network. It has two components: a *routing mesh* and a *distributed location services*.

**Tapestry Routing Mesh** Figure 3.5 shows a portion of Tapestry. Each node joins Tapestry in a distributed fashion through nearby surrogate servers and set up *neighboring* links for connection to other Tapestry nodes. The neighboring links are shown as solid arrows. Such neighboring links provide a route from every node to every other node; the routing process resolves the destination address one digit at a time. (e.g., ***8 $\Longrightarrow$ **98 $\Longrightarrow$ *598 $\Longrightarrow$ 4598, where *'s represent wildcards). This routing scheme is based on the hashed-suffix routing structure originally presented by Plaxton et al. [39].

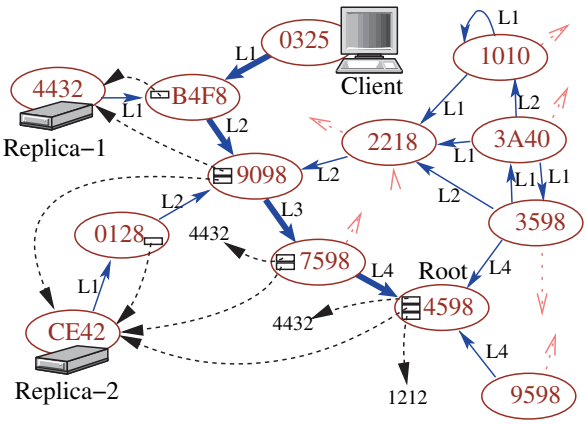



**Fig. 3.5** A Distributed Directory (Tapestry): Nodes connected via links (*solid arrows*). Nodes route to nodes one digit at a time: e.g. 1010 → 2218 → 9098 → 7598 → 4598. Objects are associated with one particular "root" node (e.g. 4598). Servers publish replicas by sending messages toward root, leaving back-pointers (*dotted arrows*). Clients route directly to replicas by sending messages toward root until encountering pointer (e.g. 0325 → *B4F*8 → 4432)

**Tapestry Distributed Location Service** Tapestry assigns a globally unique name (GUID) to every object. It then deterministically maps each GUID to a unique *root* node. Storage servers *publish* objects by sending messages toward the roots, depositing *location pointers* at each hop. Figure 3.5 shows two replicas and the Tapestry root for an object. These mappings are simply pointers to the server *s* where object *o* is being stored, and not a copy of the object itself. Thus for nearby objects, client search messages quickly intersect the path taken by publish messages, resulting in quick search results that exploit locality. It is shown that the average distance travelled in locating an object is *proportional* to the distance from that object [39].

### *3.2.5 Multicast for Disseminating Updates*

For update dissemination, IP multicast has *fundamental* problems as the architectural foundation for Internet distribution. For instance, it works only across space, not across time, while most content distribution on the Internet work across both [16]. Further, there is no widely available inter-domain IP multicast.

As an alternative, many application-level multicast (in short, ALM) systems have been proposed [7, 12, 16, 17, 38, 51]. Among them, some [7, 12, 38] target small group, multi-source applications, such as video-conferencing, while others [16, 17, 51] focus on large-scale, single-source applications, such as streaming media multicast. Bayeux [51] is also built on top of Tapestry. It uses the Tapestry location service to find the multicast root(s), and then uses Tapestry routing to route both the control (e.g. "join") and data messages. In contrast, we only use the Tapestry location mechanism to find the nearby replica.

Most ALM systems have scalability problems, since they utilize a central node to maintain states for all existing children [7, 12, 17, 38], or to handle all "join" requests [51]. Replicating the root is the common solution [17, 51], but this suffers from consistency problems and communication overhead. On the other hand, Scribe [46] and the update multicast system of SCAN (namely dissemination tree) leverage peer-to-peer routing and location services, and do not have the scalability problem. Scribe is a large-scale event notification system, using overlay DHT for both subscription and dissemination. The dissemination tree is more efficient because we use overlay DHT only for subscription, and use IP for dissemination directly.

### *3.2.6 Summary*

In summary, we find that previous work on CDNs and its related techniques have the following limitations.

1. Client-initiated web caching is myopic, while the server-initiated web caching has unscalable content state exchange overhead. Neither can adapt to network congestion/failures or provide distributed load balancing.

2. CDNs rely on centralized location services, thus they have to either apply inefficient and pull-based replication (uncooperative CDN), or replicate at the granularity of per Website and sacrifice the performance to clients (cooperative CDN).
3. There is no performance or DoS attack resilience benchmark for existing location services. This makes it difficult to compare the alternative proposals.
4. No coherence to replicas/caches: IP multicast doesn't exist in the Internet, while the existing application-level multicast has scalability problem.

In SCAN, the first two limitations are addressed with distributed location services, Tapestry, and we propose a network DoS resilience benchmark to contrast its performance with other alternatives [8]. For limitation 4, we dynamically place replicas and self-organize them into a scalable application-level multicast tree to disseminate updates as presented next.

## 3.3 Dynamic Replica Placement Problem Formulation

As shown in Fig. 3.1, replica placement is a key component of SCAN. According to users' requests, it dynamically places a minimal number of replicas while meeting client QoS and server capacity constraints. The location services discussed in last section are notified about the new replicas via Tapestry *PUBLISHOBJECT* API [50].

There is a large design space for modelling Web replica placement as an optimization problem and we describe it as follows. Consider a popular Web site or a CDN hosting server, which aims to improve its performance by pushing its content to some hosting server nodes. The problem is to dynamically decide where content is to be replicated so that some objective function is optimized under a dynamic traffic pattern and set of clients' QoS and/or resource constraints. The objective function can either minimize clients' QoS metrics, such as latency, loss rate, throughput, etc., or minimize the replication cost of CDN service providers, e.g. network bandwidth consumption, or an overall cost function if each link is associated with a cost. For Web content delivery, the major resource consumption in replication cost is the network access bandwidth at each Internet Data Center (IDC) to the backbone network. Thus, when given a Web object, the cost is linearly proportional to the number of replicas.

As Qiu et al. tried to minimize the total response latency of all the clients' requests with the number of replicas as constraint [40], we tackle the replica placement problem from another angle: minimize the number of replicas when meeting clients' latency constraints and servers' capacity constraints. Here we assume that clients give reasonable latency constraints as it can be negotiated through a service-level agreement (SLA) between clients and CDN vendors. Thus we formulate the Web content placement problem as follows.

Given a network $G$ with $C$ clients and $S$ server nodes, each client $c_i$ has its *latency constraint* $d_i$, and each server $s_j$ has its load/bandwidth/storage *capacity constraint* $l_j$. The problem is to find a smallest set of servers $S'$ such that the distance between

any client $c_i$ and its "parent" server $s_{c_i} \in S'$ is bounded by $d_i$. More formally, find the minimum $K$, such that there is a set $S' \subset S$ with $|S'| = K$ and $\forall\, c \in C$, $\exists\, s_c \in S'$ such that distance($c$, $s_c$) $\leq d_c$. Meanwhile, these clients $C$ and servers $S'$ self-organize into an application-level multicast tree with $C$ as leaves and $\forall\, s_i \in S'$, its fan-out degree (i.e. number of direct children) satisfies $f(s_i) \leq l_i$.

## 3.4 Replica Placement Algorithms

The presence of an underlying DOLR with routing locality can be exploited to perform simultaneous replica placement and tree construction. Every SCAN server is a member of the DOLR. Hence, new replicas are published into the DOLR. Further, each client directs its requests to its proxy SCAN server; this proxy server interacts with other SCAN servers to deliver content to the client.

Although we use the DOLR to locate replicas during tree building, we otherwise communicate through IP. In particular, we use IP between nodes in a d-tree for parents and children to keep track of one another. Further, when a client makes a request that results in placement of a new replica, the client's proxy keeps a cached pointer to this new replica. This permits direct routing of requests from the proxy to the replica. Cached pointers are soft state since we can always use the DOLR to locate replicas.

### *3.4.1 Goals for Replica Placement*

Replica placement attempts to satisfy both *client latency* and *server load* constraints. *Client latency* refers to the round-trip time required for a client to read information from the SCAN system. We keep this within a pre-specified limit. *Server load* refers to the communication volume handled by a given server. We assume that the load is directly related to the number of clients it handles and the number of d-tree children it serves. We keep the load below a specified maximum. Our goal is to meet these constraints while minimizing the number of deployed replicas, keeping the d-tree balanced, and generating as little traffic during update as possible. Our success will be explored in Sect. 3.6.

### *3.4.2 Dynamic Placement*

Our dynamic placement algorithm proceeds in two phases: *replica search* and *replica placement*. The replica search phase attempts to find an existing replica that meets the client latency constraint without being overloaded. If this is successful, we place a link in the client and cache it at the client's proxy server. If not, we proceed to the replica placement phase to place a new replica.

Replica search uses the DOLR to contact a replica "close" to the client proxy; call this the *entry* replica. The locality property of the DOLR ensures that the entry replica is a reasonable candidate to communicate with the client. Further, since the d-tree is connected, the entry replica can contact all other replicas. We can thus imagine three search variants: *Singular* (consider only the entry replica), *Localized* (consider the parent, children, and siblings of the entry replica), and *Exhaustive* (consider all replicas). For a given variant, we check each of the included replicas and select one that meets our constraints; if none meet the constraint, we proceed to place a new replica.

**procedure** DynamicReplicaPlacement_Naive($c$, $o$)
1 $c$ sends *JOIN* request to $o$ through DOLR, reaches entry server $s$. Request collects $\text{IP}_{s'}$, $dist_{overlay}(c,s')$ and $rc_{s'}$ for each server $s'$ on the path.
2 **if** $rc_s > 0$ **then**
  **if** $dist_{overlay}(c, s) \leq d_c$ **then** $s$ becomes parent of $c$, **exit**.
  **else**
3  $s$ pings $c$ to get $dist_{IP}(c, s)$.
4  **if** $dist_{IP}(\text{c, s}) \leq d_c$ **then** $s$ becomes parent of $c$, **exit**.
  **end**
**end**
5 At $s$, choose $s'$ on path with $rc_{s'} > 0$ *and* smallest $dist_{overlay}(t,c) \leq d_c$
**if** $\nexists$ *such* $s'$ **then**
6  for each server $s'$ on the path, $s$ collects $dist_{IP}(c,s')$ and chooses $s'$ with $rc_{s'} > 0$ *and* smallest $dist_{IP}(t,c) \leq d_c$.
**end**
7 $s$ puts a replica on $s'$ and becomes its parent, $s'$ becomes parent of $c$.
8 $s'$ publishes replica in DOLR, **exit**.

**Algorithm 1** Naive Dynamic Replica Placement. Notation: Object $o$. Client $c$ with latency constraint $d_c$. Entry Server $s$. Every server $s'$ has remaining capacity $rc_{s'}$ (additional children it can handle). The overlay distance ($dist_{overlay}$(x,y)) and IP distance ($dist_{IP}$(x,y)) are the round trip time (RTT) on overlay network and IP network, separately.

We restrict replica placement to servers visited by the DOLR routing protocol when sending a message from the client's proxy to the entry replica. We can locate these servers without knowledge of global IP topology. The locality properties of the DOLR suggest that these are good places for replicas. We consider two placement strategies: *Eager* places the replica as close to the client as possible and *Lazy* places the replica as far from the client as possible. If all servers that meet the latency constraint are overloaded, we replace an old replica; if the entry server is overloaded, we disconnect the oldest link among its d-trees.

**procedure** DynamicReplicaPlacement_Smart($c$, $o$)
1 $c$ sends *JOIN* request to $o$ through DOLR, reaches entry server $s$
2 From $s$, request forwarded to children ($sc$), parent ($p$), and siblings ($ss$)
3 Each family member $t$ with $rc_t > 0$ sends $rc_t$ to $c$. $c$ measures $dist_{IP}(t,c)$ through TCP three-way handshaking.
4 **if** $\exists$ $t$ *and* $dist_{IP}(t,c) \leq d_c$ **then**
5 $c$chooses $t$ as parent with biggest $rc_t$ and $dist_{IP}(t,c) \leq d_c$, **exit**.
**else**
6 $c$ sends *PLACEMENT* request to $o$ through DOLR, reaches entry server $s$
Request collects $IP_{s'}$, $dist_{overlay}(c,s')$ and $rc_{s'}$ for each server $s'$ on the path.
7 At $s$, choose $s'$ on path with $rc_{s'} > 0$ *and* largest $dist_{overlay}(t,c) \leq d_c$
**if** $\nexists$ *such* $s'$ **then**
8 for each server $s'$ on the path, $s$ collects $dist_{IP}(c,s')$ and chooses $s'$ with $rc_{s'} > 0$ *and* largest $dist_{IP}(t,c) \leq d_c$.
**end**
9 $s$ puts a replica on $s'$ and becomes its parent, $s'$ becomes parent of $c$.
10 $s'$ publishes replica in DOLR, **exit**.
**end**

**Algorithm 2** Smart Dynamic Replica Placement. Notation: Object $o$. Client $c$ with latency constraint $d_c$. Entry Server $s$. Every server $s'$ has remaining capacity $rc_{s'}$ (additional children it can handle). The overlay distance ($dist_{overlay}$(x,y)) and IP distance ($dist_{IP}$(x,y)) are the round trip time (RTT) on overlay network and IP network, separately.

#### 3.4.2.1 Dynamic Techniques

We can now combine some of the above options for search and placement to generate dynamic replica management algorithms. Two options that we would like to highlight are as follows.

- *Naive Placement:* A simple combination utilizes *Singular* search and *Eager* placement. This heuristic generates minimal search and placement traffic.
- *Smart Placement:* A more sophisticated algorithm is shown in Algorithm 2. This algorithm utilizes *Localized*search and *Lazy* placement.

Note that we try to use the overlay latency to estimate the IP latency in order to save "ping" messages. Here the client can start a daemon program provided by its CDN service provider when launching the browser so that it can actively participate in the protocols. The locality property of Tapestry naturally leads to the locality of d-tree, i.e. the parent and children tend to be close to each other in terms of the number of IP hops between them. This provides good delay and multicast bandwidth consumption when disseminating updates, as measured in Sect. 3.6. The tradeoff between the naive and smart approaches is that the latter consumes more "join" traffic to construct a tree with fewer replicas, covering more clients, with less delay and multicast bandwidth consumption. We evaluate this tradeoff in Sect. 3.6.

### 3.4.2.2 Static Comparisons

The replica placement methods given above are unlikely to be optimal in terms of the number of replicas deployed, since clients are added sequentially and with limited knowledge of the network topology. In the static approach, the root server has complete knowledge of the network and places replicas *after* getting all the requests from the clients. In this scheme, updates are disseminated through IP multicast. Static placement is not very realistic, but may provide better performance since it exploits knowledge of the client distribution and global network topology.

The problem formulated in Sect. 3.3 can be converted to a special case of the capacitated facility location problem [24] defined as follows. Given a set of locations $i$ at which facilities may be built, building a facility at location $i$ incurs a cost of $f_i$. Each client $j$ must be assigned to one facility, incurring a cost of $d_j c_{ij}$ where $d_j$ denotes the demand of the node $j$, and $c_{ij}$ denotes the distance between $i$ and $j$. Each facility can serve at most $l_i$ clients. The objective is to find the number of facilities and their locations yielding the minimum total cost.

To map the facility location problem to ours, we set $f_i$ always 1, and set $c_{ij}$ 0 if location $i$ can cover client $j$ or $\infty$ otherwise. The best approximation algorithm known today uses the primal-dual schema and Lagrangian relaxation to achieve a guaranteed factor of 4 [24]. However, this algorithm is too complicated for practical use. Instead, we designed a greedy algorithm that has a logarithmic approximation ratio.

Besides the previous notations, we define the following variables: set of covered clients by $s$: $C_s$, $C_s \subseteq C$ and $\forall\, c \in C_s$, $dist_{IP}(c, s) \leq d_c$; set of possible server parents for client $c$: $S_c$, $S_c \subseteq S$ and $\forall\, s \in S_c$, $dist_{IP}(c, s) \leq d_c$.

**procedure** ReplicaPlacement_Greedy_DistLoadBalancing($C$, $S$)
**input** : Set of clients to be covered: $C$, total set of servers: $S$
**output**: Set of servers chosen for replica placement: $S'$
**while** $C$ *is not empty* **do**
    Choose $s \in S$ which has the largest value of min(cardinality $|C_s|$, remaining capacity $rc_s$)
    $S' = S' \bigcup \{s\}$
    $S = S - \{s\}$
    **if** $|C_s| \leq rc_s$ **then** $C = C - C_s$
    **else**
        Sort each element $c \in C_s$ in increasing order of $|S_c|$
        Choose the first $rc_s$ clients in $C_s$ as $C_{sChosen}$
        $C = C - C_{sChosen}$
    **end**
    recompute $S_c$ for $\forall\, c \in C$
**end**
return $S'$.

**Algorithm 3** Static Replica Placement with Load Balancing.

We consider two types of static replica placement:

- *IP Static:* The root has global IP topology knowledge.
- *Overlay Static:* For each client $c$, the root only knows the servers on the Tapestry path from $c$ to the root which can cover that client (in IP distance).

The first of these is a "guaranteed-not-to-exceed" optimal placement. We expect that it will consume the least total number of replicas and lowest multicast traffic. The second algorithm explores the best that we could expect to achieve gathering all topology information from the DOLR system.

### *3.4.3 Soft State Tree Management*

Soft-state infrastructures have the potential to be extremely robust, precisely because they can be easily reconfigured to adapt to circumstances. For SCAN we target two types of adaptation: fault recovery and performance tuning.

To achieve fault resilience, the data source sends periodic *heartbeat* messages through the d-tree. Members know the frequency of these heartbeats and can react when they have not seen one for a sufficiently long time. In such a situation, the replica initiates a *rejoin* process – similar to the replica search phase above – to find a new parent. Further, each member periodically sends a *refresh* message to its parent. If the parent does not get the refresh message within a certain threshold, it invalidates the child's entry. With such soft-state group management, any SCAN server may crash without significantly affecting overall CDN performance.

Performance tuning consists of pruning and re-balancing the d-tree. Replicas at the leaves are pruned when they have seen insufficient client traffic. To balance the d-tree, each member periodically rejoins the tree to find a new parent.

## 3.5 Evaluation Methodology

We implement an event-driven simulator for SCAN because *ns*2 [5] can only scale up to one thousand nodes. This includes a packet-level network simulator (with a static version of the Tapestry DOLR) and a replica management framework. The soft-state replica layer is driven from simulated clients running workloads. Our methodology includes evaluation metrics, network setup and workloads.

### *3.5.1 Metrics*

Our goal is to evaluate the replica schemes of Sect. 3.4.2. These strategies are dynamic naive placement (*od_naive*), dynamic smart placement (*od_smart*), overlay static placement (*overlay_s*), and static placement on IP network (*IP_s*). We compare the efficacy of these four schemes via three classes of metrics:

- *Quality of Replica Placement*: Includes number of deployed replicas and degree of load distribution, measured by the ratio of the standard deviation vs. the mean of the number of client children for each replica server.

- *Multicast Performance*: We measure the relative delay penalty (RDP) and the bandwidth consumption which is computed by summing the number of bytes multiplied by the transmission time over every link in the network. For example, the bandwidth consumption for 1 K bytes transmitted in two links (one has 10 ms, the other 20 ms latency) is 1 KB $\times$ (10+20)ms = 0.03(KB.sec).
- *Tree Construction Traffic:* We count both the number of application-level messages sent and the bandwidth consumption for deploying replicas and constructing d-tree.

In addition, we quantify the effectiveness of capacity constraints by computing the *maximal load* with or without constraints. The maximal load is defined as the maximal number of client cache children on any SCAN server. Sensitivity analysis are carried out for various client/server ratios and server densities.

### *3.5.2 Network Setup*

We use the GT-ITM transit-stub model to generate five 5000-node topologies [49]. The results are averaged over the experiments on the five topologies. A packet-level, priority-queue based event manager is implemented to simulate the network latency. The simulator models the propagation delay of physical links, but does not model bandwidth limitations, queuing delays, or packet losses.

We utilize two strategies for placing SCAN servers. One selects all SCAN servers at random (labelled *random SCAN*). The other preferentially chooses transit and gateway nodes (labelled *backbone SCAN*). This latter approach mimics the strategy of placing SCAN servers strategically in the network.

To compare with a DNS-redirection-based CDN, we simulate typical behavior of such a system. We assume that every client request is redirected to the closest CDN server, which will cache a copy of the requested information for the client. This means that popular objects may be cached in every CDN server. We assume that content servers are allowed to send updates to replicas via IP multicast.

### *3.5.3 Workloads*

To evaluate the replication schemes, we use both a synthetic workload and access logs collected from real Web servers. These workloads are a first step toward exploring more general uses of SCAN.

Our synthetic workload is a simplified approximation of *flash crowds*. Flash crowds are unpredictable, event-driven traffic surges that swamp servers and disrupt site services. For our simulation, all the clients (not servers) make requests to a given hot object in random order.

Our trace-driven simulation includes a large and popular commercial news site, MSNBC [36], as well as traces from NASA Kennedy Space Center [37]. Table 3.2

**Table 3.2** Statistics of Web site access logs used for simulation

| Web Site | Period | # Requests Total – Simulated | # Clients Total | # Client Groups Total – Simulated | # Objects Simulated |
|---|---|---|---|---|---|
| MSNBC | 10–11 am, 8/2/99 | 1604944–1377620 | 139890 | 16369–4000 | 4186 |
| NASA | All day, 7/1/95 | 64398–64398 | 5177 | 1842–1842 | 3258 |

shows the detailed trace information. We use the access logs in the following way. We group the Web clients based on BGP prefixes [27] using the BGP tables from a BBNPlanet (Genuity) router [2]. For the NASA traces, since most entries in the traces contain host names, we group the clients based on their domains, which we define as the last two parts of the host names (e.g. a1.b1.com and a2.b1.com belong to the same domain). Given the maximal topology we can simulate is 5000 (limited by machine memory), we simulate all the clients groups for NASA and 4000 top client groups (cover 86.1% of requests) for MSNBC. Since the clients are unlikely to be on transit nodes nor on server nodes, we map them randomly to the rest of nodes in the topology.

## 3.6 Evaluation Results

In this section, we evaluate the performance of the SCAN dynamic replica management algorithms. What we will show is that:

- For realistic workloads, SCAN places close to an optimal number of replicas, while providing good load balance, low delay, and reasonable update bandwidth consumption relative to static replica placement on IP multicast.
- SCAN outperforms the existing DNS-redirection based CDNs on both replication and update bandwidth consumption.
- The performance of SCAN is relatively insensitive to the SCAN server deployment, client/server ratio, and server density.
- The capacity constraint is quite effective at balancing load.

We will first present results on synthetic workload, and then the results of real Web traces.

### *3.6.1 Results for the Synthetic Workload*

We start by examining the synthetic, flash crowd workload. 500 nodes are chosen to be SCAN servers with either “random” or “backbone” approach. Remaining nodes are clients and access some hot object in a random order. We randomly choose one non-transit SCAN server to be the data source and set as 50 KB the size of the hot object. Further, we assume the latency constraint is 50 ms and the load capacity is 200 clients/server.

### 3.6.1.1 Comparison Between Strategies

Figure 3.6 shows the number of replicas placed and the load distribution on these servers. *Od_smart* approach uses only about 30–60% of the servers used by *od_naive*, is even better than *overlay_s*, and is very close to the optimal case: *IP_s*. Also note that *od_smart* has better load distribution than *od_naive* and *overlay_s*, close to *IP_s* for both *random* and *backbone SCAN*.

Relative Delay Penalty (RDP) is the ratio of the overlay delay between the root and any member in d-tree vs. the unicast delay between them [12]. In Fig. 3.7, *od_smart* has better RDP than *od_naive*, and 85% of *od_smart* RDPs between any member server and the root pairs are within 4. Figure 3.8 contrasts the bandwidth consumption of various replica placement techniques with the optimal IP static placement. The results are very encouraging: the bandwidth consumption of *od_smart* is quite close to *IP_s* and is much less than that of *od_naive*.

The performance above is achieved at the cost of d-tree construction (Fig. 3.9). However, for both *random* and *backbone SCAN*, *od_smart* approach produces less than three times of the messages of *od_naive* and less than six times of that for optimal case: *IP_s*. Meanwhile, *od_naive* uses almost the same amount of bandwidth as *IP_s* while *od_smart* uses about three to five times that of *IP_s*.

In short, the smart dynamic algorithm has performance that is close to the ideal case (static placement with IP multicast). It places close to an optimal number of replicas, provides better load distribution, and less delay and multicast bandwidth consumption than the naive approach – at the price of three to five times as much tree construction traffic. Since d-tree construction is much less frequent than data access and update this is a good tradeoff.

Due to the limited number and/or distribution of servers, there may exist some clients who cannot be covered when facing the QoS and capacity requirements. In this case, our algorithm can provide hints as where to place more servers. Note that experiments show that the naive scheme has many more uncovered clients than the smart one, due to the nature of its unbalanced load. Thus, we remove it from consideration for the rest of synthetic workload study.

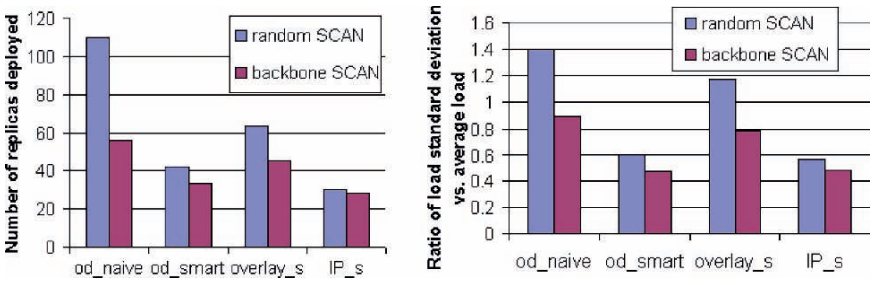



**Fig. 3.6** Number of replicas deployed (*left*) and load distribution on selected servers (*right*) (500 SCAN servers)

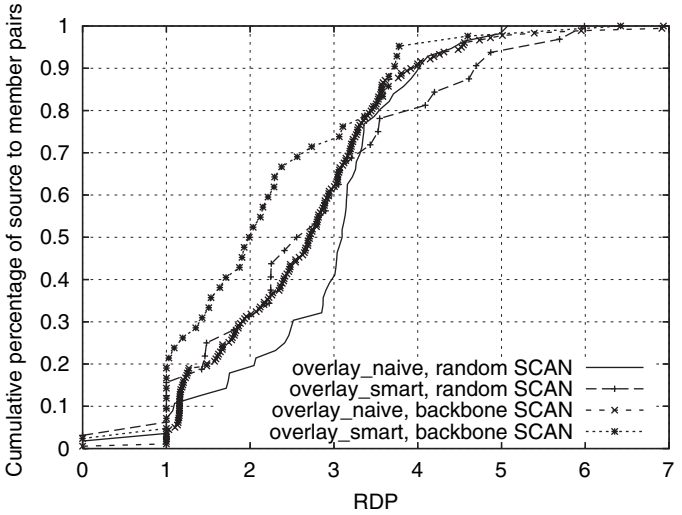



**Fig. 3.7** Cumulative distribution of RDP (500 SCAN servers)

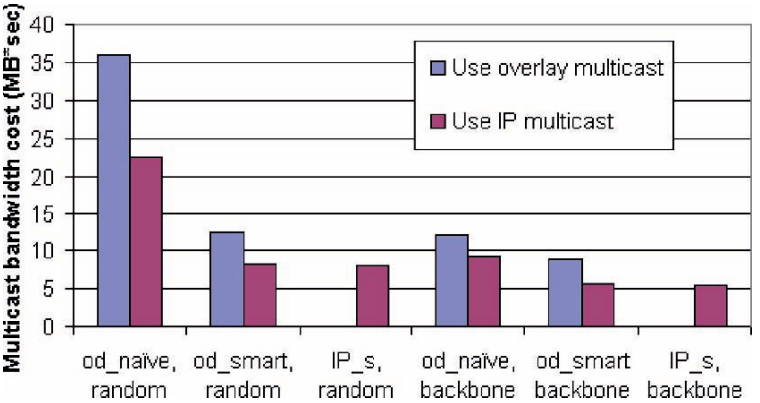



**Fig. 3.8** Bandwidth consumption of 1MB update multicast (500 SCAN servers)

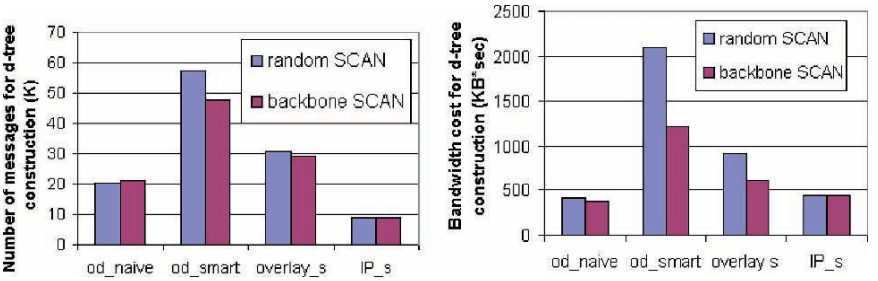



**Fig. 3.9** Number of application-level messages (*left*) and total bandwidth consumed (*right*) for d-tree construction (500 SCAN servers)

#### 3.6.1.2 Comparison with a CDN

As an additional comparison, we contrast the overlay smart approach with a DNS-redirection-based CDN. Compared with a traditional CDN, the overlay smart approach uses a fraction of the number of replicas (6–8%) and less than 10% of bandwidth for disseminating updates.

#### 3.6.1.3 Effectiveness of Distributed Load Balancing

We study how the capacity constraint helps load balancing with three client populations: 100, 1000 and 4500. The former two are randomly selected from 4500 clients. Figure 3.10 shows that lack of capacity constraints (labelled *w/o LB*) leads to hot spot or congestion: some servers will take on about 2–13 times their maximum load. Performance with load balancing is labelled as *w/ LB* for contrast.

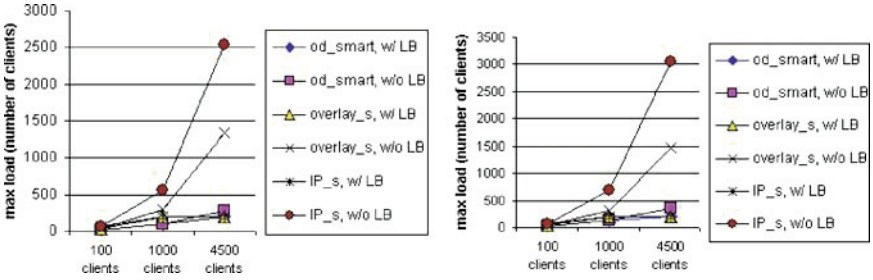



**Fig. 3.10** Maximal load measured with and without load balancing constraints (LB) for various numbers of clients (*left*: 500 random servers, *right*: 500 backbone servers)

#### 3.6.1.4 Performance Sensitivity to Client/Server Ratio

We further evaluate SCAN with the three client populations. Figure 3.11 shows the number of replicas deployed. When the number of clients is small, *w/ LB* and *w/o LB* do not differ much because no server exceeds the constraint. The number of replicas required for *od_smart* is consistently less than that of *overlay_s* and within the bound of 1.5 for *IP_s*. As before, we also simulate other metrics, such as load distribution, delay and bandwidth penalty for update multicast under various client/server ratios. The trends are similar, that is, *od_smart* is always better than *overlay_s*, and very close to *IP_s*.

#### 3.6.1.5 Performance Sensitivity to Server Density

Next, we increase the density of SCAN servers. We randomly choose 2500 out of the 5000 nodes to be SCAN servers and measure the resulting performance. Obviously,

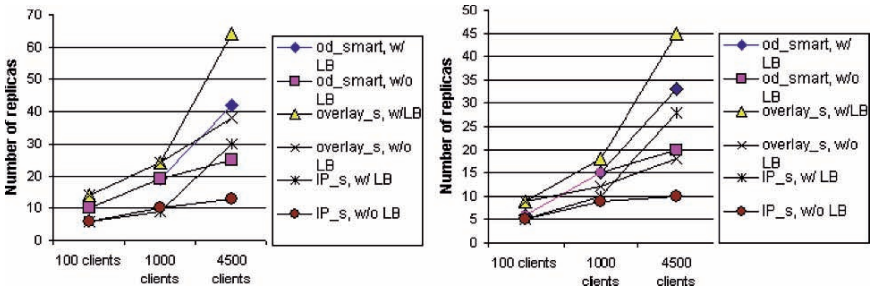



**Fig. 3.11** Number of replicas deployed with and without load balancing constraints (LB) for various numbers of clients (*left*: 500 random servers, *right*: 500 backbone servers)

this configuration can support better QoS for clients and require less capacity for servers. Hence, we set the latency constraint to be 30 ms and capacity constraint 50 clients/server. The number of clients vary from 100 to 2500.

With very dense SCAN servers, our *od_smart* still uses less replicas than *overlay_s*, although they are quite close. *IP_s* only needs about half of the replicas, as in Fig. 3.12. In addition, we notice that the load balancing is still effective. That is, overloaded machines or congestion cannot be avoided simply by adding more servers while neglecting careful design.

In summary, *od_smart* performs well with various SCAN server deployments, various client/server ratios, and various server densities. The capacity constraint based distributed load balancing is effective.

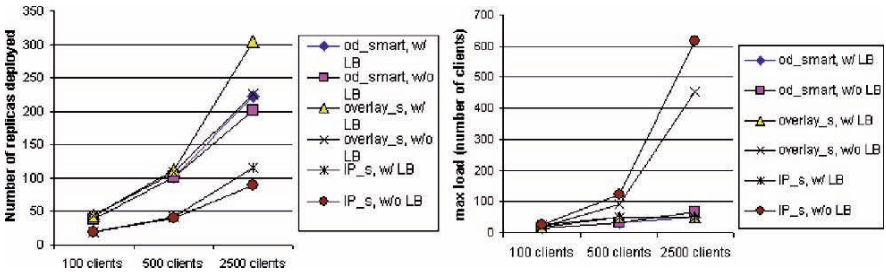



**Fig. 3.12** Number of replicas deployed (*left*) and maximal load (*right*) on 2500 random SCAN servers with and without the load balancing constraint (LB)

### *3.6.2 Results for Web Traces Workload*

Next, we explore the behavior of SCAN for Web traces with documents of widely varying popularity. Figure 3.13(a) characterizes the request distribution for the two traces used (note that the *x*-axis is logarithmic.). This figure reveals that the request number for different URLs is quite unevenly distributed for both traces.

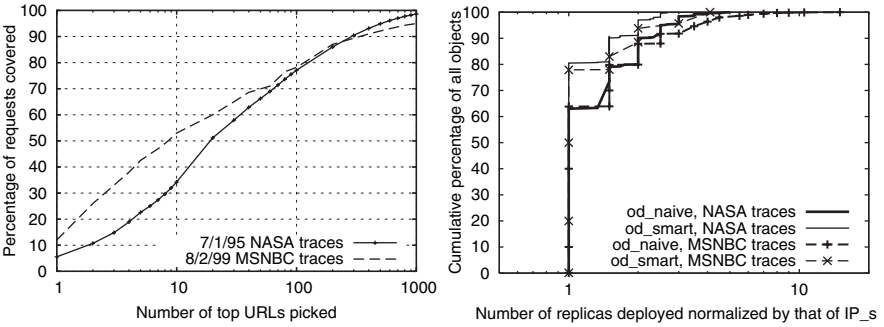



**Fig. 3.13** Simulation with NASA and MSNBC traces on 100 backbone SCAN servers. (**a**) Percentage of requests covered by different number of top URLs (*left*); (**b**) the CDF of replica number deployed with *od_naive* and *od_smart* normalized by the number of replicas using *IP_s* (*right*)

For each URL in the traces, we compute the number of replicas generated with *od_naive*, *od_smart*, and *IP_s*. Then we normalize the replica numbers of *od_naive* and *od_smart* by dividing them with the replica number of *IP_s*. We plot the CDF of such ratios for both NASA and MSNBC in Fig. 3.13(b). The lower percentage part of the CDF curves are overlapped and close to 1. The reasons are most of the URLs have very few requests, and we only simulate a limited period, thus the number of replicas deployed by the three methods are very small and similar. However, *od_smart* and *od_naive* differ significantly for popular objects, exhibited in the higher percentage part. *Od_smart* is very close to *IP_s*, for all objects, the ratio is less than 2.7 for NASA and 4.1 for MSNBC, while the ratio for *od_naive* can go as high as 5.0 and 15.0, respectively.

In addition, we contrast the bandwidth consumption for disseminating updates. Given an update of unit size, for each URL, we compute the bandwidth consumed by using (1) overlay multicast on an *od_naive* tree, (2) overlay multicast on an *od_smart* tree, and (3) IP multicast on an *IP_s* tree. Again, we have metric (1) and (2) normalized by (3), and plot the CDF of the ratios. The curves are quite similar to Fig. 3.13(b).

In conclusion, although *od_smart* and *od_naive* perform similarly for infrequent or cold objects, *od_smart* outperforms dramatically over *od_naive* for hot objects which dominate overall requests.

### *3.6.3 Discussion*

How does the distortion of topology through Tapestry affect replica placement? Notice that the overlay distance through Tapestry, on average, is about 2–3 times more than the IP distance. Our simulations in Sect. 3.6, shed some light on the resulting penalty: *Overlay_s* applies exactly the same algorithm as *IP_s* for replica placement, but uses the static Tapestry-level topology instead of IP-level topology. Simulation

results show that *overlay_s* places 1.5–2 times more replicas than $IP_s$. For similar reasons, *od_smart* outperforms *overlay_s*. The reason is that *od_smart* uses "ping" messages to get the real IP distance between clients and servers. This observation also explains why *od_smart* gets similar performance to *IP_s*. One could imagine scaling overlay latency by an expected "stretch" factor to estimate real IP distance – thereby reducing ping probe traffic.

## 3.7 Conclusions

The importance of adaptive replica placement and update dissemination is growing as distribution systems become pervasive and global. In this chapter, we present SCAN, a scalable, soft-state replica management framework built on top of a distributed object location and routing framework (DOLR) with locality. SCAN generates replicas on demand and self-organizes them into an application-level multicast tree, while respecting client QoS and server capacity constraints. An event-driven simulation of SCAN shows that SCAN places close to an optimal number of replicas, while providing good load distribution, low delay, and small multicast bandwidth consumption compared with static replica placement on IP multicast. Further, SCAN outperforms existing DNS-redirection based CDNs in terms of replication and update cost. SCAN shows great promise as an essential component of global-scale peer-to-peer infrastructures.

**Acknowledgements** Some of the materials presented in this chapter appeared in a preliminary form at Pervasive'02 (the first International Conference on Pervasive Computing) [9]. I would like to thank other co-authors who contributed to the previous form of this work: Prof. Randy H. Katz and Prof. John D. Kubiatowicz from UC Berkeley and Prof. Lili Qiu from UT Austin.

## References

1. Barbir, A., Cain, B., Douglis, F., Green, M., Hofmann, M., Nair, R., Potter, D., and Spatscheck, O. Known CN request-routing mechanisms. http://www.ietf.org/internet-drafts/draft-ietf-cdi-known-request-routing-00.txt.
2. BBNPlanet. telnet://ner-routes.bbnplanet.net.
3. Bestavros, A. Demand-based document dissemination to reduce traffic and balance load in distributed information systems. In *Proceedings of the IEEE Symposium on Parallel and Distributed Processing* (1995).
4. Bestavros, A., and Cunha, C. Server-initiated document dissemination for the WWW. In *IEEE Data Engineering Bulletin* (Sep. 1996).
5. Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y., and Yu, H. Advances in network simulation. *IEEE Computer 33*, 5 (May 2000), 59–67.
6. Castro, M., and Liskov, B. Proactive recovery in a byzantine-fault-tolerant system. In *Proceedings of USENIX Symposium on OSDI* (2000).
7. Chawathe, Y., McCanne, S., and Brewer, E. RMX: Reliable multicast for heterogeneous networks. In *Proceedings of IEEE INFOCOM* (2000).

8. Chen, Y., Bargteil, A., Bindel, D., Katz, R. H., and Kubiatowicz, J. Quantifying network denial of service: A location service case study. In *Proceeding of Third International Conference on Information and Communications Security (ICICS)* (2001).
9. Chen, Y., Katz, R. H., and Kubiatowicz, J. D. SCAN: a dynamic scalable and efficient content distribution network. In *Proceedings of the First International Conference on Pervasive Computing* (Aug. 2002).
10. Chen, Y., Qiu, L., Chen, W., Nguyen, L., and Katz, R. H. Clustering Web content for efficient replication. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)* (2002).
11. Chen, Y., Qiu, L., Chen, W., Nguyen, L., and Katz, R. H. Efficient and adaptive Web replication using content clustering. *IEEE Journal on Selected Areas in Communications (J-SAC), Special Issue on Internet and WWW Measurement, Mapping, and Modeling 21*, 6 (2003), 979–994.
12. Chu, Y., Rao, S., and Zhang, H. A case for end system multicast. In *Proceedings of ACM SIGMETRICS* (June 2000).
13. Czerwinski, S., Zhao, B., Hodes, T., Joseph, A., and Katz, R. An architecture for a secure service discovery service. In *Proceedings of ACM/IEEE MobiCom Conference* (1999).
14. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing* (September/October 2002), 50–58.
15. Fan, L., Cao, P., Almeida, J., and Broder, A. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proceedings of ACM SIGCOMM Conference* (1998).
16. Francis, P. Yoid: Your own Internet distribution. Technical report, ACIRI, http://www.aciri.org/yoid, April, 2000.
17. Gifford, D. K., Johnson, K. L., Kaashoek, M. F., and O'Toole, J. W. Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of USENIX Symposium on OSDI* (2000).
18. Gray, J., Helland, P., O'Neil, P., and Shasha, D. The dangers of replication and a solution. In *Proceedings of ACM SIGMOD Conference* (June 1996), *25*, 2, pp. 173–182.
19. Guttman, E., Perkins, C., Veizades, J., and Day, M. Service Location Protocol, Version 2. IETF Internet Draft, November 1998. RFC 2165.
20. Gwertzman, J., and Seltzer, M. World-Wide Web Cache consistency. In *Proceedings of the 1996 USENIX Technical Conference* (1996).
21. Gwertzman, J., and Seltzer, M. An analysis of geographical push-caching. In *Proceedings of International Conference on Distributed Computing Systems* (1997).
22. Hildrum, K., Kubiatowicz, J., Rao, S., and Zhao, B. Distributed data location in a dynamic network. In *Proceedings of ACM SPAA* (2002).
23. Howes, T. A. The lightweight directory access Protocol: X.500 Lite. Tech. Rep. 95–8, Center for Information Technology Integration, U. Mich., July 1995.
24. Jain, K., and Varirani, V. Approximation algorithms for metric facility location and *k*-median problems using the primal-dual schema and lagrangian relaxation. In *Proceedings of IEEE FOCS* (1999).
25. Jamin, S., Jin, C., Kurc, A., Raz, D., and Shavitt, Y. Constrained mirror placement on the Internet. In *Proceedings of IEEE Infocom* (2001).
26. Kistler, J., and Satyanarayanan, M. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems 10*, 1 (Feb. 1992), 3–25.
27. Krishnamurthy, B., and Wang, J. On network-aware clustering of Web clients. In *Proceedings of SIGCOMM* (2000).
28. Krishnamurthy, B., Wills, C., and Zhang, Y. On the use and performance of content distribution networks. In *Proceedings of SIGCOMM Internet Measurement Workshop* (2001).
29. Kubiatowicz, J., et al. Oceanstore: An architecture for global-scale persistent storage. In *Proceeedings of 9th ASPLOS* (2000).
30. Li, B., Golin, M. J., Italiano, G. F., Deng, X., and Sohraby, K. On the optimal placement of Web proxies in the Internet. In *Proceedings of IEEE INFOCOM* (1999).
31. Limelight Networks Inc. `http://www.limelightnetworks.com/`.
32. Luotonen, A., and Altis, K. World-Wide Web proxies. In *Proceedings of the First International Conference on the WWW* (1994).

33. Mao, Z. M., Cranor, C., Douglis, F., Rabinovich, M., Spatscheck, O., and Wang, J. A precise and efficient evaluation of the proximity between Web clients and their local DNS servers. In *Proceedings of USENIX Technical Conference* (2002).
34. Michel, S., Nguyen, K., Rosenstein, A., Zhang, L., Floyd, S., and Jacobson, V. Adaptive Web caching: Towards a new caching architecture. In *Proceedings of 3rd International WWW Caching Workshop* (June, 1998).
35. Mirror Image Internet Inc. `http://www.mirror-image.com`.
36. MSNBC. http://www.msnbc.com.
37. NASA kennedy space center server traces. `http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html`.
38. Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M. ALMI: An application level multicast infrastructure. In *Proceedings of 3rd USENIX Symposium on Internet Technologies* (2001).
39. Plaxton, C. G., Rajaraman, R., and Richa, A. W. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the SCP SPAA* (1997).
40. Qiu, L., Padmanabhan, V. N., and Voelker, G. M. On the placement of Web server replica. In *Proceedings of IEEE INFOCOM* (2001).
41. Rabinovich, M., and Aggarwal, A. RaDaR: A scalable architecture for a global Web hosting service. In *Proceedings of WWW* (1999).
42. Radoslavov, P., Govindan, R., and Estrin, D. Topology-informed Internet replica placement. In *Proceedings of the International Workshop on Web Caching and Content Distribution* (2001).
43. Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM* (2001).
44. Rodriguez, P., and Sibal, S. SPREAD: Scalable platform for reliable and efficient automated distribution. In *Proceedings of WWW* (2000).
45. Rowstron, A., and Druschel, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of ACM Middleware* (2001).
46. Rowstron, A., Kermarrec, A.-M., Castro, M., and Druschel, P. SCRIBE: The design of a large-scale event notification infrastructure. In *Proceedings of International Workshop on Networked Group Communication (NGC)* (2001).
47. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM* (2001).
48. Venkataramani, A., Yalagandula, P., Kokku, R., Sharif, S., and Dahlin, M. The potential costs and benefits of long term prefetching for content distribution. In *Proceedings of Web Content Caching and Distribution Workshop 2001* (2001).
49. Zegura, E., Calvert, K., and Bhattacharjee, S. How to model an Internetwork. In *Proceedings of IEEE INFOCOM* (1996).
50. Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications* (2003).
51. Zhuang, S. Q., Zhao, B. Y., Joseph, A. D., Katz, R. H., and Kubiatowicz, J. D. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of ACM NOSSDAV* (2001).

# Chapter 4
# Content Delivery and Management

Claudia Canali, Valeria Cardellini, Michele Colajanni and Riccardo Lancellotti

## 4.1 Introduction

The Web has evolved in the last decade from a mean to distribute content with marginal interest to a major communication media, where critical content and services are delivered to the users. This success was mainly driven by the concerns of content providers about the user-perceived performance of content delivery. When high availability, scalability, and performance are required, a common solution for content providers is to exploit third-party services to improve the performance of content and service delivery. The technical goal of Content Delivery Network (CDN) providers is to guarantee adequate delivery performance even when the incoming request load is overwhelming for the content provider alone.

CDNs have been originally proposed to primarily distribute static Web content and some limited streaming audio/video content over the Internet. First-generation CDNs were designed primarily to ease network congestion caused by the delivery of static Web pages through congested public peering points. However, the current context for content delivery is very different from that of the inception of these infrastructures, which date back to almost 10 years ago. Indeed, the Web scenario is currently characterized by an increased sophistication and complexity in delivered content. Modern Web contents are often dynamically generated and personalized according to the user preferences and needs. Traditional content delivery technologies designed for static content are not able to meet the new needs, as there are inherent challenges and limitations in the delivery of dynamic content that need to

Claudia Canali
University of Modena and Reggio Emilia, 41100 Modena, Italy, e-mail: claudia.canali@unimore.it

Valeria Cardellini
University of Roma "Tor Vergata", 00133 Roma, Italy, e-mail: cardellini@ing.uniroma2.it

Michele Colajanni
University of Modena and Reggio Emilia, 41100 Modena, Italy,
e-mail: michele.colajanni@unimore.it

Riccardo Lancellotti
University of Modena and Reggio Emilia, 41100 Modena, Italy,
e-mail: riccardo.lancellotti@unimore.it

R. Buyya et al. (eds.), *Content Delivery Networks*,
© Springer-Verlag Berlin Heidelberg 2008

be overcome. In the last years, CDN started to evolve towards the support for the delivery of dynamically generated content, allowing the content providers to exploit the benefits of CDNs for modern Web applications and services.

This chapter explores the issues of content delivery through CDNs, with a special focus on the delivery of dynamically generated and personalized content. We describe the main functions of a modern Web system and we discuss how the delivery performance and scalability can be improved by replicating the functions of a typical multi-tier Web system over the nodes of a CDN. For each solution, we present the state of the art in the research literature, as well as the available industry-standard products adopting the solution. Furthermore, we discuss the pros and cons of each CDN-based replication solution, pointing out the scenarios that provide the best benefits and the cases where it is detrimental to performance.

The rest of this chapter is organized as follows. Section 4.2 provides some background material, presenting the logical layers of a Web system for delivering dynamic and personalized content and classifying the main caching and replication solutions for the logical layers. Sections 4.3, 4.4, and 4.5 discuss in details how the logical layers can be mapped over the nodes of a CDN in order to accelerate the delivery of static and dynamic resources. Section 4.6 discusses the issues related to the management and replication of user profile information over a distributed delivery infrastructure. Finally, Sect. 4.7 concludes the chapter with some final remarks and outlines the open research directions.

## 4.2 Systems for Web Content Delivery

In this section, we provide some background material on the architecture of the systems employed to generate and deliver Web content. We first review in Sect. 4.2.1 the logical layers of a multi-tier Web system; we then describe in Sect. 4.2.2 the architecture and the main functionalities of a CDN. Finally, in Sect. 4.2.3 we classify the caching and replication approaches for the logical layers of a Web system that can be employed by CDNs to accelerate the generation and delivery of dynamic and personalized content.

### *4.2.1 Logical Layers of a Web System*

The large majority of modern Web systems is based on a multi-tier logical architecture, that separates the HTTP interface, the application (or business) logic, the data repository and, when existing, the user-related information for authentication and content personalization. These logical architecture layers are often referred to as *front-end*, *application*, *back-end*, and *user profile* layers [8]. Figure 4.1 shows the structure of a typical system providing Web-based services. We recognize the logical components of the system and the fundamental interactions between the layers.

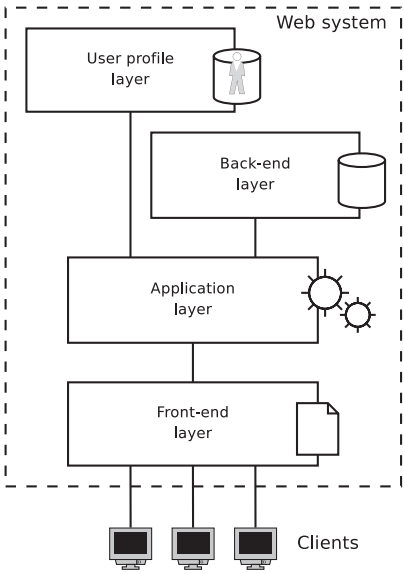



**Fig. 4.1** Logical layers of a Web system

The front-end layer is the interface of the Web-based service. It accepts HTTP connection requests from the clients, serves static content from the file system, and represents an interface towards the application logic of the middle layer. The delivery of static content is a straightforward operation. A static Web content is typically stored in a file system and client requests for this type of content are managed by the HTTP server, which retrieves the resources from the file system and sends them back to the client in HTTP responses.

Examples of static content that are handled by the front-end layer are:

- *Web objects embedded in a Web page*. Typical embedded objects are images, style sheets, and active components such as flash animations, Java applets, and ActiveX controls.
- *Multimedia content*. Audio and video streams are static content handled by the front-end layer. To allow a smooth consumption of multimedia content by the client, the common approach is to rely on *HTTP streaming*, that is to divide the resources into chunks of data that are delivered in sequence to the client. The client can start the playback as soon as the first data chunk has arrived, without waiting for the delivery of the whole resource [26].
- *Page fragments*. They are portions of a Web page with a distinct theme or functionality [14]. Each fragment is considered as an independent information entity. For example, the page of a Web portal is typically described as composed by fragments such as latest news, feature articles, link bars, and advertisements. The use of fragments in the management of static content aims to improve the re-usability of Web content, because some fragments are common to multiple pages. However, when fragment-based management of static content is used, the front-end layer is also responsible for the *assembly* of fragments to build the Web page prior to its delivery to the user.

The application layer is at the heart of a Web-based service: it handles all the business logic and computes the information which is used to build responses with dynamically generated content. Content generation often requires interactions with the back-end and user profile layers: hence the application layer must be capable of interfacing the application logic with the data storage of the back-end and must be able to access the user profile when personalized content needs to be generated. Dynamic content is generated on-the-fly as a response to client requests. Examples of dynamic content generated by the application layer are:

- Answers retrieved from an organized source of information, such as the shopping cart page or searches in an e-commerce site.
- Web content generated dynamically to separate the content from its representation. For example, content management systems[1] or XML-based technologies [47] provide mechanisms for separating structure and representation details of a Web document. In these systems, the content (even when its lifecycle is relatively long) is generated dynamically from a template on-the-fly. The burden of dynamic data generation, that requires a computational effort due to data retrieval from databases, (optional) information processing and construction of the HTTP output, is outweighted by the convenience of handling data through software specifically designed to this aim, such as a DBMS.
- Web content generated by user social behavior. For example, the pages of forums or blogs provide an exchange place for messages written by the Web users.

The back-end layer manages the main information repository of a Web-based service. It typically consists of a database server and storage of critical information that is a source for generating dynamic content. If we refer to the examples of dynamic content generation from the application layer, we can identify the following data repositories:

- In the case of an e-commerce site, we use a database for storing the product lists, accessed for searching in the product catalog. Furthermore, also the user interactions are managed using a database for shopping cart status or list of purchases.
- In the case of a content management system, the dynamic generation of content accesses the database to retrieve both the Web page templates and the actual contents during the generation of Web resources.
- For Web sites such as blogs or forums, articles, comments, and posts are typically stored in a database.

The user profile layer stores information on the user preferences and context [16]. This information is accessed in the generation of dynamic content to provide personalized content. The information stored in the user profile may be originated from multiple sources such as:

- Information supplied by the user, that is usually provided through a fill-in form to add/edit user preferences. This profile communication may occur when the user registers itself for the access to a Web-based service or may be filled/modified later.

[1] For a list of the most popular CMS software, the reader may refer to http://www.cmsmatrix.org.

- Information inferred from the analysis of user behavior that is typically obtained from data mining of Web logs [20, 21, 27]. Typical examples of Web-based services that rely on information gathered through data mining are the recommendation systems for e-commerce [27] or the advertisements tailored on the user preferences.

### *4.2.2 A Simplified CDN Architecture*

A CDN's architecture aims to achieve high performance and scalability by leveraging the principle of replicating the system resources (that is, the Web servers) to guarantee a high level of performance in the service of a huge amount of content. Replication occurs both at local and geographic level. In the case of local replication of system resources, the servers used for the service of user requests are tightly connected. They are placed on the same LAN and usually share a single upstream link connecting the system to the rest of the Internet. The common term to describe such a system is *cluster*. Servers within a cluster provide increased computing power thanks to the replication of system resources. They can interact in a fast and effective way [11]. Moreover, the replication may improve fault tolerance because a faulty node can be easily bypassed.

LAN-based systems have many pros, but they have scalability problems related to efficient generation and delivery of resources when the Web site is highly popular. The first problem that affects replication on a local scale is the so called first mile, i.e. the network link connecting the cluster to the Internet. This link can represent the system bottleneck for the end-to-end performance; moreover, it is a potential single point of failure. Traffic on the Web-based cluster zone, failures on an external router, and Denial-of-Service (DoS) attacks may cause the service to be unreachable independently of the computational power of the cluster platform. When better scalability and performance are needed, it is useful to replicate some elements of the infrastructure over a geographic scale.

A simplified view of a CDN's geographically distributed architecture is shown in Fig. 4.2. We distinguish two types of servers in a typical CDN, namely, *edge servers* and the *core server* [18, 36]. Edge servers are replicated on the so-called *network edge*, which means that these servers are as close as possible to the clients, typically in the Points of Presence (POP) of multiple Interner Service Providers (ISPs), and are mainly responsible for the interaction with clients. Client requests are typically diverted from the origin server to the edge server by means of DNS-based redirection [12, 36]. This approach is based on modified DNS servers that resolve queries for the site hostname with the IP address of a suitable edge server (the algorithm used to detect the most suitable edge server is usually complex and takes into account geographic and network distance, network link and edge server status).

The core server is a logical entity that handles the functions that are related to the management of the infrastructure, coordination of request distribution policies,

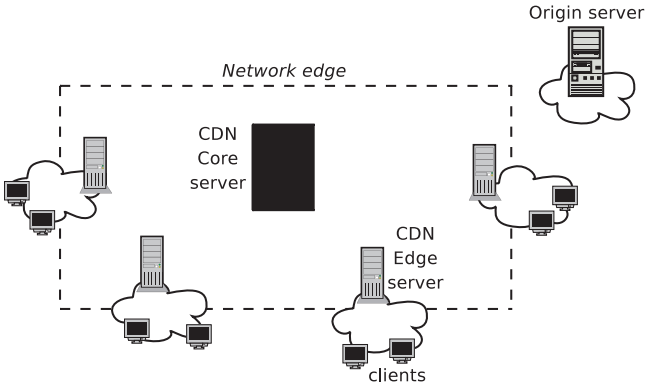



**Fig. 4.2** A simplified CDN architecture

and billing. It can be implemented as a single powerful server or, more often, as a multi-cluster, that is a set of clusters that cooperate and behave like a single virtual computer with high availability and computational power.

### *4.2.3 Accelerating Content Generation and Delivery*

The trend of Web evolution towards an ever increasing demand of scalable and high performance content delivery requires the content provider to rely on CDNs. On the other hand, CDNs should develop techniques to accelerate the delivery of content on behalf of the content provider.

To analyze how a CDN can accelerate the delivery of Web content and applications, we focus our attention on the origin server and on the edge servers, that are the elements of the Web infrastructure most involved in the content delivery process. The directions to address scalability and performance issues are the classical two: caching and replication. Indeed, CDNs replicate some logical layers of the origin server on their edge servers. Since we have four logical levels in the Web system, we envision four mapping approaches, as illustrated in Fig. 4.3.

- **Replication of front-end layer**. The edge server is responsible only for the management of static content. This approach is typical of the first generation of CDNs, where the edge servers, called *surrogate* servers, behave like reverse proxies to accelerate the delivery of content that can be stored at the file system level [36, 49]. The replicated Web content may be whole Web objects, for example when a CDN is used for delivering embedded objects or multimedia resources, or the replication may consider a more fine-grained approach, replicating Web fragments [14].
- **Replication of application layer**. A CDN is used to improve the delivery performance of dynamically generated content. This approach, called *edge computing*

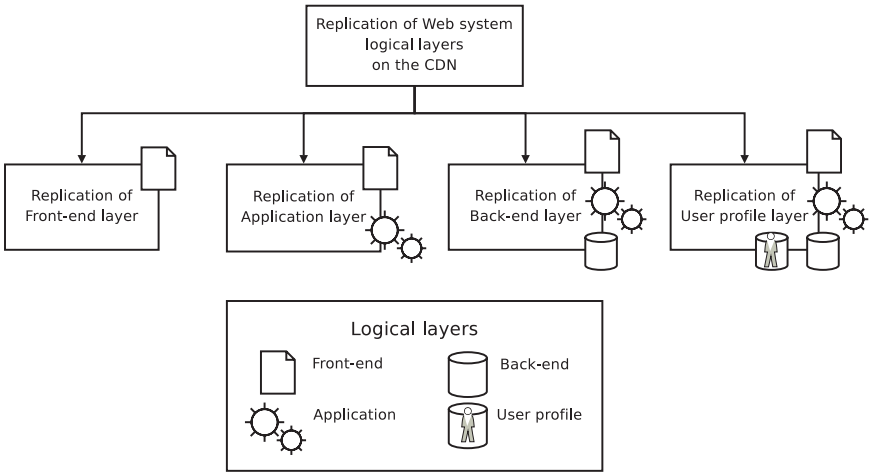



**Fig. 4.3** Possible mapping of Web system logical layers on a CDN infrastructure

[44], moves Web application programs or components directly on the edge server [18, 37] with the aim of generating dynamic Web content close to the clients.

- **Replication of back-end layer**. The edge server provides both the functions for generating dynamic content and hosts data involved in the content generation. The origin server (or the core server of the CDN) is only responsible for the management of the infrastructure and acts as a master copy of the data.
- **Replication of user profile layer**. The edge server hosts also the data repository used for the generation of personalized content [42].

## 4.3 Front-End Layer Replication

The replication of the front-end layer aims to improve performance and scalability in the delivery of static content, as shown in Fig. 4.4. Such content is *cached* on the CDN edge servers. Moving the delivery of static content on the network edge addresses scalability issues, because it avoids the risk of network congestion in peering points and WAN links, that provides a major contribution to network-related delays [36].

Accelerating the delivery of static content using a third-party infrastructure is a common approach for improving the performance of content delivery, and dates back to the first generation of CDNs, such as the Adero CDN or the Akamai media delivery service [2]. However, delivery of this content is still a critical task, due to the growing amount of rich-media content [50] that is becoming a significant fraction of Web traffic. Moving the delivery of such media content close to the clients may have an important benefit for two reasons. First, due to the large size

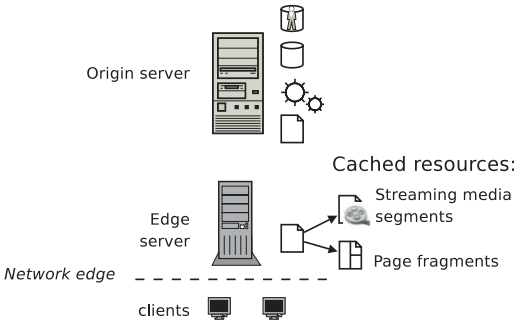



**Fig. 4.4** Replication of the front-end layer on the edge server

of these content, network-related delays at peering points may have a significant impact on the user-perceived performance. Second, due to the techniques of HTTP streaming, which is commonly used, reducing the variance in delivery time results in smoother playback [26].

Due to the large size of multimedia content, it is common to cache on the edge server only the most popular fraction of each multimedia content instead of storing the whole resource (this is usually referred as *segment caching*) [15, 25], as shown in Fig. 4.4. The popularity of each fragment within a multimedia resource depends on the user access patterns. In the case of sequential access, the common approach is to rely on *sequential caching*, that is storing the first part of the resource to reduce buffering time. On the other hand, when the access patterns involve a significant amount of seek operation within the media, different caching techniques, such as *interleaved caching*, may be more effective [25].

The approach of dividing streaming content into segments has been proposed also for Web resources, in the case of the delivery of Web content assembled from *fragments* (represented among the cached resources in Fig. 4.4). This solution requires more effort from the edge server, because the front-end layer must include the functions for the separate caching and the assembly of fragments. Being an independent information entity, each fragment can have its own cacheability profile, which describes the ability to cache it and its Time-To-Live (TTL), thus allowing to manage the content freshness and lifetime at a fragment granularity rather than at the Web page level.

Upon a user requests the Web page (template), the edge server examines its cache for the included page fragments and assembles the page on-the-fly. Only those fragments either stale or deemed impossible to cache are fetched from the origin server. Therefore, using fragment-based caching and dynamic assembly on the edge servers, the origin server obtains two advantages: first, it does not have to assemble the page; second, it is typically required to deliver only a small fraction of the page, corresponding to stale or non-cacheable fragments. As regards the user-perceived performance, fragment-based caching has been proved to be effective in improving response time by serving most of the resources that comprise a dynamically generated page at the edge of the Internet, close to the end user [38, 51]. Furthermore, fragment-based caching has also beneficial effects on the edge servers. Indeed, it improves the disk space utilization because fragments that are shared across different

Web pages need to be stored only once; furthermore, it reduces the amount of invalidation at the edge server, because only those parts of the Web page that expire need to be invalidated.

The common standard for fragment-based caching is represented by Edge Side Includes (ESI) [19], which is an XML-based markup language that enables to distinguish via XML cacheable and non-cacheable content. The content provider designs and develops the business logic to form and assemble the pages by using the ESI specification within its development environment. Besides the primary functionality for including fragments within a page (even in a conditional way), the other key functionalities provided by ESI include the support for handling exceptions due to fragments unavailability and the support for explicit invalidation of cached fragments in such a way that it guarantees a stronger consistency than that provided by a TTL-based mechanism [19, 29].[2]

Fragment-based publishing and caching of Web pages have been adopted by companies and commercial products, including the EdgeSuite network of Akamai [2] based on the ESI specification and IBM's WebSphere Edge Server [28]. A large-scale deployment of a Web publishing system based on the fragment-based approach and compatible with ESI has been presented by Challenger et al. in [14]. This system is able to construct complex objects from fragments and has been developed to handle major sporting events at Web sites hosted by IBM. The authors also addressed the problem of detecting and updating all Web pages affected by one or more fragment changes. The proposal is to adopt different algorithms based on graph traversal that can be used according to the consistency requirements. A comparative study of four different offloading and caching strategies at the edge servers has been conducted by Yuan et al. in [51] using a representative e-commerce benchmark. Their results show that a simple strategy of offloading the functionality of composing Web pages from fragments can be very effective in terms of latency and server load reduction.

Most edge servers that support fragment-based caching do not provide any support for cooperation among the individual edge caches, i.e. these are treated as completely independent entities. This limitation does not allow to take full advantage of the potential caching capabilities of the edge servers that can be exploited through cooperation. Some effort toward this direction has been taken in the Akamai EdgeSuite network, which however includes only a limited cooperation among its edge servers. A recent work by Ramaswamy et al. [39] has addressed some significant challenges in designing a large-scale cooperative network of edge servers. Their proposal presents low-cost cooperative techniques based on dynamic hashing-based document lookup and update protocols and considers also how to provide failure resilience of individual edge servers.

The major drawbacks of the fragment-based solution are related to its applicability with respect to the type of dynamic content being delivered and to the task of fragmenting a Web page. Indeed, fragment-based caching can be effectively applied if the stream of requests is characterized by a high locality and if updates in

[2] For an analysis of cache consistency mechanisms in CDNs, the reader may refer to Chap. 5 of this book.

the content of the origin server are not frequent. This condition ensures that the fragment cacheability profiles are sufficient for managing the content freshness, thus relieving the origin server from the task to explicitly invalidate cached fragments. Furthermore, this technique suffers from lack of transparency, since caching, fragmentation, and assembling should be implemented on a per-application basis. For example, ESI requires a complete revision of the Web page code, because ESI code must be added over the original code, and its performance is dependent on the page structure. This manual identification and markup of page fragments is also hardly manageable for edge servers which deliver content from multiple providers. To overcome the manual fragmentation of Web pages, in [38] Ramaswamy et al. have proposed a scheme to automatically detect fragments in a Web page. Their approach depends upon a careful analysis of the dynamic Web pages with respect to their information sharing behavior and change patterns.

## 4.4 Application Layer Replication

A performance bottleneck in CDNs that replicate only the front-end layer is represented by the application layer in the origin server, which is responsible for the generation of dynamic content according to the Web application logic. Replication of application layer, commonly known as *edge computing* [18, 45], aims to improve the delivery of dynamically generated content by offloading this task from the origin server. The application code is replicated at multiple edge servers, while the data layer is still kept centralized. The computation is pushed to the edge of the network, as illustrated in Fig. 4.5.

In edge computing, each edge server has a full copy of the application code while the back-end layer is still centralized in the origin server, i.e. the edge servers

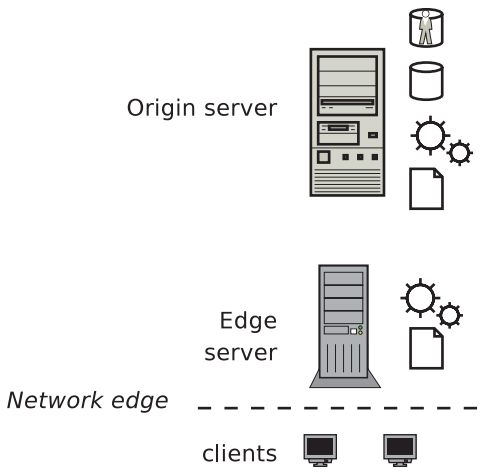



**Fig. 4.5** Replication of the application layer on the edge server

continue to share a centralized database. By pushing the computation task to the edge of the network, the load on the origin server can be reduced and the CDN can achieve better efficiency and performance and higher availability with respect to the front-end only replication approach, where the application and data layers are centrally managed.

We can identify two architectural solutions depending on the ability of the edge server to distinguish between transactional and non-transactional requests. A transactional request is an atomic set of database operations that usually involve lock on part of the database and perform some update to the database records, while non-transactional requests have a read-only behavior on the data. If the edge server cannot distinguish the type of request, the Web server at the edge server forwards all requests to its local application layer, where they are executed. The local application logic then makes calls for database access to the centralized data layer located in the CDN core. Otherwise, if the edge server is able to distinguish between transactional and non-transactional requests, the edge server redirects only non-transactional requests to the local application layer, while transactional requests are directly forwarded to the application layer at the origin server, that then executes the transaction and accesses the centralized database.

In the application replication approach, the CDN core typically plays a coordinator role, being in charge for migrating and/or replicating the applications on the edge servers and keeping track of the replicas. It can be also responsible for maintaining the application replicas consistent with the primary copy. The CDN core may accomplish this functionality using a simple server-based invalidation that is, updating the application on the edge servers when the developer changes the primary copy.

Edge computing has been proposed and applied in a variety of commercial products and academic projects. For example, it is the heart of the EdgeComputing product from Akamai [3], which hosts customer-supplied J2EE components on edge-side application servers. Akamai EdgeComputing employs a two-level model for replicating the application layer: JSPs and servlets that contain the presentation logic are deployed on the edge servers of the Akamai network, while the business tier components that are tightly coupled with back-end applications or a database typically remain in the CDN core at the origin server.

Process migration issues have been addressed by many years of research; an example of complex system that could be employed in the CDN context is vMatrix [9], which migrates the entire dynamic state of the application from one server to another. However, Web applications do not require a real application migration at an arbitrary time but only at request boundaries [37]. Therefore, a significant simplification applicable in the CDN context is the automatic deployment of the application at the edge servers. ACDN (where the acronym stands for Application CDN) by Rabinovich et al. [37] is an application distribution framework that exploits this concept of automatic deployment; the application is dynamically replicated by the central coordinator on the basis of the observed demand. The framework implementation is based on a meta-file, which contains the list of the files comprising the application and an initialization script.

The DotSlash framework by Zhao and Schulzrinne [52, 53] is another academic project that adopts a dynamic script replication technique to manage dynamic content. DotSlash was not designed for large-scale CDNs, but it rather provides a system to handle sudden load spikes that affect the performance of Web sites through the dynamic provisioning of rescue servers which act as caching proxies.

The application layer replication is characterized for enabling the customization of concrete and specific applications. The application replication approach is neither generic nor transparent. Indeed, it requires customization on a per-application basis, because a manual configuration is needed to choose the components to be offloaded and where to deploy applications. For example, in ACDN [37], applications can be deployed and re-deployed dynamically, but manual administration is still involved, such as creating the meta-file for each application that has to be replicated. This application customization increases substantially the total cost of ownership, and it is prone to codification errors. Some effort for automatically deciding how to replicate Web applications has been proposed in [33]. However, these studies are mainly focused on a small scale scenario and may be not suitable for a large scale CDN, with tens of thousands of edge servers.

Further disadvantages of the application layer replication approach steam from keeping the data centralized at the origin server. This architectural choice determines two drawbacks. First, if the edge servers are located worldwide, as in large-scale CDNs, then each data access incurs a WAN delay; second, the centralized database may quickly become a performance bottleneck, as it needs to serve all database requests from the whole system. Therefore, the application replication solution is suitable only for those Web sites that require few accesses to the centralized database in order to generate the requested content.

The remaining approaches discussed in the next section aim to mitigate the centralized data layer bottleneck, which limits the overall CDN scalability. Therefore, the further steps in offloading the functionalities of the origin server to the edge servers exploit caching and replication techniques for the data layer.

## 4.5 Back-End Layer Replication

The edge computing approach may not solve every scalability problem, since in some Web applications the bottleneck lies in the back-end layer [13] instead of the application layer. In this case, scalability issues can be addressed by assigning to a third party (i.e. a CDN) the management of application data. A CDN provides answers to the queries of the application layer hosted by the edge servers on behalf of the back-end tier of the origin server.

The available solutions for replicating a data storage have been widely studied in the context of databases [23]. In this chapter, we will limit the scope of our analysis to the replication of data in the back-end layer of a Web system. In this scenario, the available approaches are summarized in [43]: the replication of the data stored in the back-end layer may be complete or partial, as illustrated in Fig. 4.6. The

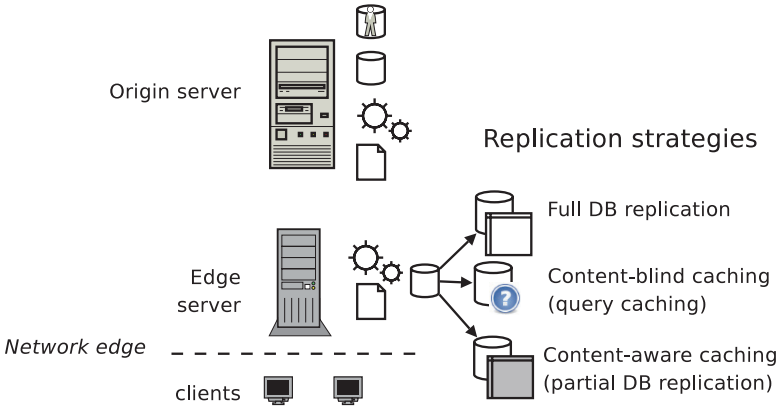



**Fig. 4.6** Replication of the back-end layer on the edge server

partial replication of data can be obtained by exploiting a caching mechanism of the most popular queries to the data storage (*Content-Blind Caching*) or by actively replicating portions of the back-end data, selected on the basis of usage patterns, network, and infrastructure status [44] (*Content-Aware Caching*).

We anticipate that there is no clear winner among these alternatives, due to the different access patterns of the Web application to the database. Indeed, a work by Gao et al. [22] propose different replication strategies depending on the nature of the Web applications considered.

### *4.5.1 Content Blind Caching*

When content-blind caching is adopted, edge servers cache the results of previous queries to the database. In such a way, the server may process locally future identical or similar queries, thus improving performance and relieving the load on the origin server back-end layer.

The approach of caching query results to replicate the back-end layer is highly popular. For example, the GlobeCGC [40] system explicitly aims to cache queries on the edge servers of a geographically distributed systems such as a CDN. Recently, the idea of dynamically replicating the back-end tier using a query cache to improve scalability has been proposed. For example, the QCache module of the DotSlash framework [53] proposes an agreement of cooperating Web sites that can temporarily enable a distributed query cache facility to alleviate the overload conditions in case of unexpected traffic surges.

The effectiveness of the query results caching depends on the achievable cache hit rate. To improve the amount of queries that can be serviced accessing the query cache, the characteristic of content blindness of the caching mechanism may be relaxed. To this aim, sophisticated query matching engines can be used so that a new

query can be answered using a union of already cached queries instead of contacting the origin server. Support for this enhanced query matching engine is a distinctive feature of DBproxy [4]. An efficient way to merge cached queries has been proposed in [30], where each query contributes to populate an (initially empty) copy of the original back-end database. DBCache [10] supports database caching at the level of tables, allowing to cache at the edge node either the entire content or a subset of the tables of the centralized database server.

Caching mechanisms should guarantee consistency of the cached data. Since a traditional TTL-based approach is not suitable for every Web application, some specific consistency enforcing mechanism has been proposed. For example, Olston et al. rely on a multicast messaging system to ensure that invalidation messages are sent to every query cache [32], while Tolia et al. [48] use hash functions to guarantee that no stale data are served from the cache.

The query-caching support on the edge server is an important feature that is available in multiple commercial products, including BEA WebLogic and IBM WebSphere. In particular, IBM WebSphere supports query caching through the so-called *Materialized Query Tables*. A materialized query table (MQT) is a table that materializes the pre-computed result of a query involving one or more tables. After the MQT is created and populated, an arbitrary subsequent query may be satisfied by the MQT, if the MQT matches all or a part of the query. A similar feature is provided by BEA WebLogic by means of EJBs.

Even if some consistency enforcing mechanism is adopted, the network layers on the geographic infrastructure can lead to hosting data at the edge servers that are stale with respect to the current state at the centralized data layer. This might not be a problem for read-mostly scenarios, where the Web applications do not need transactional semantics. However, for an important class of applications (e.g. when payment operations are involved) transactional semantics is a must and database updates are frequent. In these cases, query caching may not be a viable option. Furthermore, database caching techniques are suited only for those applications which repeatedly issue the same queries to the data layer. For applications which do not exhibit this temporal locality, it can be more efficient to replicate partially or entirely the data layer at the edge servers.

### *4.5.2 Content Aware Caching*

In the case of content-aware caching, each edge server runs its own database server, which contains a partial view of the centralized database. The typical approach for partial data replication is to push section of the database close to the edge, according to access patterns. Since the aim is to improve the response time perceived by the end user, the algorithms for replica placement (such as HotZone) usually include network latency in the performance model [46].

A significant example of replication mechanism is provided by GlobeDB [43], that uses partially replicated databases based on data partition to reduce update traffic. However, this solution relies on one special server, which holds the full

database, to execute complex queries. Thereby, it may suffer from scalability because of the new throughput bottleneck represented by the special server. GlobeTB [24] improves the approach of GlobeDB with the goal of not only reducing the latency but also to increase the throughput of the replicated database. To this aim, GlobeTP relaxes the need for a single centralized master database, thus avoiding the risk of bottleneck in the origin server back-end.

As in the case of query caching, also partial database replication may suffer from consistency problems. Ganymed [35] explicitly addresses the issue of how to guarantee data consistency when the replicated back-end tiers are subject to changes (i.e. when update, delete or insert queries are issued). To this aim, Ganymed separates updates from read-only transactions, and routes updates to a main centralized database server and queries to read-only database copies.

The support for partial replication of databases is also available in multiple commercial products. For example, the mySQL DBMS supports a scheme for partitioning data among multiple replicas. Similar features have been also introduced into IBM DB2 and Oracle. However, in most cases partial replication schemes in databases are designed to manage a local replication of the resources (i.e. database clustering), and require a centralized manager that handles and distributes queries and transactions over the database partitions. This approach cannot be directly applied to the context of large-scale geographical replications, because the presence of a centralized manager would hinder the scalability of the system. For this reason, most commercial products rely more on query caching rather than on database replication schemes.

### *4.5.3 Full Database Replication*

Full database replication maintains identical copies of the database at multiple locations. By moving a copy of the database to the edge servers and keeping the database copies coordinated among them, it becomes possible to completely deliver dynamic content at the edge of the network, without the need to modify each deployed application. However, the management of database replication introduces severe consistency management problems, that are particularly critical to solve when the client requests trigger frequent updates on persistent data. This is a well know issue that the database community has being addressed for a long time.

Traditionally, data replication is achieved through either *lazy* or *eager* write update propagation [23]. In the eager (or synchronous) approach, the coordination among the replicas occurs before the transaction commits, while in the lazy approach updates are only propagated after the transaction commits. The eager approach favors fault-tolerance and offers the same correctness guarantees as a single database. However, it suffers from severe limitations regarding performance and scalability that may render it impractical [23]. On the other hand, the lazy approach favors performance and scales very well; therefore, commercial replication products typically use it. However, the lazy approach introduces new problems, because transactions can read stale data and conflicts between updating transactions can be detected late, introducing the need for conflict resolution.

The simplest solution to manage database replication in Web environments is based on a centralized primary copy at the origin server and replicated secondary copies at the edge servers. Read-only transactions can be executed completely at the edge by accessing the local secondary database copy. However, for transactions that require updating operations (as in write-mostly scenarios), all database accesses are redirected to the database primary copy located at the centralized origin server. The primary database propagates any update to the secondary databases on a regular basis. A first drawback of this approach is that the edge servers must be aware of the application semantics, because it has to know whether a request triggers an update or a read-only transaction. Moreover, in this solution the consistency of the replicated data is maintained through a lazy update propagation scheme, which presents two negative effects. First, the data at the edge servers might be stale. Second, a crash might cause a data loss.

The exploitation of full database replication in the Web environment poses a number of challenging problems. Indeed, most database replication techniques proposed up to now assume that the database replicas are interconnected through a LAN. In recent years, the database community has proposed many replication protocols that provide both data consistency and good performance in LANs. As we focus on Web environments, we only mention some works that have addressed database replication in the context of locally distributed Web systems. The interested reader may also refer to [31] for a more comprehensive analysis on database replication systems based on group communications. A lazy replication solution that provides serializability and throughput scaling through the reduction of the number of conflicts has been proposed by Amza et al. in [5]; this earlier work has been improved through the introduction of distributed versioning, which provides strong consistency and avoids deadlock problems [6]. A recent work by the same authors investigates how to combine query result caching and cluster replication solutions [7]. A middleware tool that supports consistent and scalable data replication has been presented in [34].

In a CDN the database replicas are geographically spread in a WAN. If the Web application generates a significant number of database updates, a large amount of traffic may overload the wide-area network and impact negatively on performance, because each update needs to be propagated to all the other replicas to maintain the consistency of the replicated data. A performance analysis of data replication techniques that provide strong consistency in wide-area networks through group communications has been presented in [31]. However, the scalability analysis performed in this work is limited to eight replicas. Therefore, we can conclude that scalability and performance for database replication in WANs are largely an open issue that call for further research efforts.

## 4.6 User Profile Layer Replication

The user profile layer relies on a database for data storage, like the back-end layer. Hence, the possible solutions for replicating the user profile correspond to that

already described in Sect. 4.5. However, the access patterns for this layer are quite different if compared to the back-end layer.

In particular, the user typically interacts with only one edge server, hence the profile of a given user is accessed by one edge server for the whole duration of a user session. This access pattern has a significant impact on consistency and replication policies. Indeed, the whole dataset of user profiles can be partitioned and distributed over the edge nodes depending on the user access patterns. Since no replication is needed, consistency issues are limited to guarantee that the user profiles on the edge servers are consistent with the data on the origin server. The main approaches to manage the user profiles are therefore restricted to content blind or content-aware data caching, because full database replication is clearly unnecessary.

However, it is worth to note that, even if the user accesses only one edge server for the whole duration of its session, user migration among multiple edge servers may occur between consecutive session. Therefore, it is necessary to guarantee that the user profile data migrates following the user, as shown in Fig. 4.7. The support for this behavior is not explicitly optimized in most replication strategies for back-end data. Some proposals to handle this profile migration have emerged in the last years. CONCA [41] is a generic data caching framework that aims to support user mobility by allowing data to follow the user. This framework has been extended by the same authors to explicitly support the presence of personal data in Tuxedo [42].

Besides the replication of user-related information, a further critical operation that must be carried out by the user profile layer is the actual creation and update of such information. Currently, the user profile is either manually updated by the user through Web-based forms or is automatically updated by the Web system on the basis of the user behavior. The information stored in the user profile and the way to collect them depends on the Web-based services that are to be deployed. We present and discuss some significant examples of personalized content generation.

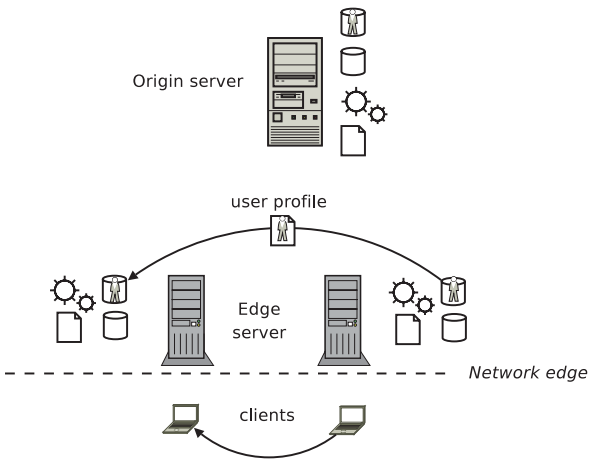



**Fig. 4.7** Replication of the user profile layer on the edge server

- *Generation of personalized content through aggregation of external data sources*. This service is common to multiple personalized portals (e.g. myYahoo, iGoogle) and provides the user with a personalized news feed, typically retrieved from heterogeneous sources by means of XML-RSS feeds. The user profile contains information on which feeds are of interest for the users and about how the personalized content is to be presented. The users provide information about the subscription to news feeds and on the preferred presentation layout through filled-in forms during the registration to the personalized portal.
- *Collaborative filtering*. This type of service supports the interaction of users providing feedback on other users or topics. This type of personalized content generation is often used in recommendation systems that provide suggestions on goods to purchase, based on analysis of similar user behaviors, or to rank the reputation of a user in a social network. Information about the user is collected both through explicit user inputs (e.g. in the case where the user reputation is based on feedback from other users) or through implicit information collection, for example by mining the user purchases to cluster the user population according to pre-defined profiles [27].
- *Location and surrounding-based services*. These services generate personalized content on the basis of the user geographic location. The user position is determined through the analysis of data-related information or is explicitly supplied by the user when accessing the service. The user location is compared with geographic data and the generation and delivery of static and dynamic content (e.g. queries) is carried out according to the user location and surrounding, possibly combined with user preferences.

These examples show that, even if some information may be provided explicitly from the users, a significant fraction of the data stored in the user profiles are inferred through data mining of log files, cookies, and user click history. With the available information collection technologies, it is possible to extract interesting information related to the users including sensitive data, such as political, physical, and sexual features. Furthermore, most techniques are almost transparent to the users which are often completely unaware. Unauthorized users information collection occurred in the last years, for example, by the doubleclick.com commercial advertisement service. Several commercial services, including search engines, were associated with doubleclick.com. The commercial sites used cookies to monitor their visitors activities, and any information collected were stored in doubleclick.com databases. These user profiles were then used by doubleclick.com to select the advertisement banners more suitable for the users.

The examples of misusing personal information raised the interest towards the issues of whether and how to inform users about personal data collection. Concerns about privacy due to log data mining and cookie analysis [1] motivate the efforts of defining novel mechanisms to negotiate what information can be derived from user behavior and how they are to be used. The Platform for Privacy Preferences (P3P) [17] is an example of a proposal aiming to address this issue: each site compliant with the P3P standard must provide an XML-encoded description of which data are collected, how they are managed, and where they are stored. Full compliance

with the P3P standard imposes some restriction to the automatic replication of user profiles, because we must ensure that the adequate level of privacy is guaranteed for every replica of the user profile.

## 4.7 Conclusions and Open Issues

The delivery of static and dynamically generated content can be accelerated through a third party, i.e. a CDN infrastructure, that replicates some of the layers of a Web system. Throughout this chapter we have analyzed the replication of every logical layer composing a Web system. For each layer, we have discussed the research proposals in the field of content delivery and we have illustrated how the CDN industry is leveraging the replication to improve the effectiveness of content delivery. In particular, our analysis shows that replication of the front-end layer is suitable when the content provider aims to accelerate the delivery of static, possibly multimedia, content. When the CDN is used to accelerate the delivery of dynamic content, replication of the application layer is required. The achievable performance gain from this approach depends on the access patterns to the data, that may still determine a bottleneck in the back-end layer for some Web applications, thus forcing the replication of this latter layer also.

The research field in content delivery presents several open issues that are yet to be addressed. Indeed, even if some proposals to accelerate the delivery of dynamically generated content have been made and adopted by the industry, the effectiveness of the proposed solutions is still highly dependent on the access patterns of the applications. In particular, the risk of creating a bottleneck in the back-end layer is still one of the main issues that hinder the scalability of dynamic Web content delivery. This problem is likely to remain a major issue even in the next years, due to the evolution of Web content and applications. The Web 2.0 is shifting the Web towards two main trends: an ever-increasing amount of personalization, and the new Web usage patterns with large upload streams. Personalized (and uncacheable) content and high frequency of content refresh reduce the effectiveness of caching mechanisms and determine a growth in the overhead due to data consistency protocols. Furthermore, the presence of personal user information introduces bounds in the possibility of user profile replication, because the content provider must preserve the privacy of user sensitive information. The complexity of the scenario is further increased by the convergence of Web 2.0 with user mobility, that disrupts access locality due to the migration of users among the edge nodes. We believe that coping with this evolution will be the next challenge for CDN operators and researchers studying solutions for content delivery.

## References

1. Agostini, A., Bettini, C., Riboni, D.: Loosely coupling ontological reasoning with an efficient middleware for context-awareness. In: Proc. of Mobiquitous 2005. San Diego, CA (2005)
2. Akamai: (2007). `http://www.akamai.com/`

3. Akamai EdgeComputing: (2007). http://www.akamai.com/html/technology/edgecomputing.html
4. Amiri, K., Park, S., Tewari, R., Padmanabhan, S.: DBProxy: A dynamic data cache for Web applications. In: Proc. of 19th IEEE Int'l Conf. on Data Engineering, pp. 821–831. Bangalore, India (2003)
5. Amza, C., Cox, A., Zwaenepoel, W.: Conflict-aware scheduling for dynamic content applications. In: Proc. of 4th USENIX Symp. on Internet Technologies and Systems (2003)
6. Amza, C., Cox, A., Zwaenepoel, W.: Distributed versioning: Consistent replication for scaling back-end databases of dynamic content web sites. In: Proc. of ACM/IFIP/Usenix Middleware Conf. (2003)
7. Amza, C., Cox, A., Zwaenepoel, W.: A comparative evaluation of transparent scaling techniques for dynamic content servers. In: Proc. of IEEE Int'l Conf. on Data Engineering (2005)
8. Andreolini, M., Colajanni, M., Mazzoni, F., Lancellotti, R.: Fine grain performance evaluation of e-commerce sites. ACM Performance Evaluation Review **32**(3) (2004)
9. Awadallah, A., Rosenblum, M.: The vMatrix: A network of virtual machine monitors for dynamic content distribution. In: Proc. of 7th Int'l Workshop on Web Content Caching and Distribution (2002)
10. Bornhovd, C., Altinel, M., Mohan, C., Pirahesh, H., Reinwald, B.: Adaptive database caching with DBCache. IEEE Data Engineering Bulletin **27**(2), 11–18 (2004)
11. Cardellini, V., Casalicchio, E., Colajanni, M., Yu, P.S.: The state of the art in locally distributed web-server systems. ACM Computing Surveys **34**(2) (2002)
12. Cardellini, V., Colajanni, M., Yu, P.: Request redirection algorithms for distributed web systems. IEEE Tran. on Parallel and Distributed Systems **14**(5) (2003)
13. Cecchet, E., Chanda, A., Elnikety, S., Marguerite, J., Zwaenepoel, W.: Performance comparison of middleware architectures for generating dynamic Web content. In: Proc. of 4th ACM/IFIP/USENIX Middleware (2003)
14. Challenger, J., Dantzig, P., Iyengar, A., Witting, K.: A fragment-based approach for efficiently creating dynamic Web content. ACM Transactions on Internet Technology **5**(2), 359–389 (2005)
15. Chen, S., Shen, B., Wee, S., Zhang, X.: Adaptive and lazy segmentation based proxy caching for streaming media. In: Proc. of ACM NOSSDAV (2003)
16. Colajanni, M., Lancellotti, R., Yu, P.S.: Scalable architectures and services for ubiquitous web access. In: Tutorial notes in 2006 World Wide Web Conf. (2006)
17. Cranor, L.: Web Privacy with P3P. O'Reilly (2002)
18. Davis, A., Parikh, J., Weihl, B.: EdgeComputing: Extending enterprise applications to the edge of the Internet. In: Proc. of 2004 World Wide Web Conf. (2004)
19. Edge Side Includes: (2007). http://www.esi.org/
20. Eiriniaki, M., Vazirgiannis, M.: Web mining for Web personalization. ACM Transactions on Internet Technology **3**(1) (2003)
21. Flesca, S., Greco, S., Tagarelli, A., Zumpano, E.: Mining user preferences, page content and usage to personalize Website navigation. World Wide Web **8**(3), 317–345 (2005)
22. Gao, L., Dahlin, M., Nayate, A., Zheng, J., Iyengar, A.: Improving availability and performance with application-specific data replication. IEEE Transactions on Knowledge and Data Engineering **6**(1), 106–120 (2005)
23. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: Proc. of ACM SIGMOD Int'l Conf. on Management of Data, pp. 173–182 (1996)
24. Groothuyse, T., Sivasubramanian, S., Pierre, G.: GlobeTP: Template-based database replication for scalable Web applications. In: Proc. of 2007 World Wide Web Conf. (2007)
25. Guo, H., Chen, S., Xiao, Z., Zhang, X.: DISC: Dynamic interleaved segment caching for interactive streaming. In: Proc. of the 25th International Conference on Distributed Computing Systems (2005)
26. Guo, L., Chen, S., Xiao, Z., Zhang, X.: Analysis of multimedia workloads with implications for Internet streaming. In: Proc. of 14th Int'l World Wide Web Conf. (2005)
27. Ho Ha, S.: Helping online customers decide through Web personalization. IEEE Intelligent systems **17**(6) (2002)

28. IBM WebSphere Edge Server: (2007). http://www-3.ibm.com/software/Webservers/edgeserver/
29. Iyengar, A., Ramasvamy, L., Schroeder, B.: Techniques for efficiently serving and caching dynamic Web content. In: S. Chanson, X. Tang, J. Xu (eds.) Web Content Delivery. Springer (2005)
30. Larson, P., Goldstein, J., Guo, H., Zhou, J.: MTCache: Mid-tier database caching for SQL server. IEEE Data Engineering Bulletin **27**(2), 35–40 (2004)
31. Lin, Y., Kemme, B., Patiño-Martínez, M., Jiménez-Peris, R.: Consistent data replication: Is it feasible in WANs? In: Proc. of Europar Conf. (2005)
32. Olston, C., Manjhi, A., Garrod, C., Ailamaki, A., Maggs, B., Mowry, T.: A scalability service for dynamic Web applications. In: Proc. of Innovative Data Systems Research, pp. 56–69. Asilomar, CA (2005)
33. Pacifici, G., Spreitzer, M., Tantawi, A., Youssef, A.: Performance management of cluster based Web services. IEEE Journal on Selected Areas in Communications **23**, 2333–2343 (2005)
34. Patiño-Martínez, M., Jiménez-Peris, R., Kemme, B., Alonso, G.: Consistent database replication at the middleware level. ACM Transactions on Computer Systems **23**(4), 1–49 (2005)
35. Plattner, C., Alonso, G.: Ganymed: Scalable replication for transactional Web applications. In: Proc. of ACM/IFIP/USENIX Int'l Middleware Conf. Toronto, Canada (2004)
36. Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley (2002)
37. Rabinovich, M., Xiao, Z., Aggarwal, A.: Computing on the edge: A platform for replicating Internet applications. In: Proc. of 8th Int'l Workshop on Web Content Caching and Distribution. Hawthorne, NY (2003)
38. Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F.: Automatic fragment detection in dynamic Web pages and its impact on caching. IEEE Transactions on Knowledge and Data Engineering **17**(6), 859–874 (2005)
39. Ramaswamy, L., Liu, L., Iyengar, A.: Scalable delivery of dynamic content using a cooperative edge cache grid. IEEE Transactions on Knowledge and Data Engineering **19**(5), 614–630 (2007)
40. Rilling, L., Sivasubramanian, S., Pierre, G.: High availability and scalability support for Web applications. In: Proc. of 2007 IEEE/JSP Int'l Symp. on Applications and the Internet. Washington, DC (2007)
41. Shi, W., Karamcheti, V.: Conca: An architecture for consistent nomadic content access. In: Proc. of Workshop on Caching, Coherence, and Consistency. Sorrento, Italy (2001)
42. Shi, W., Shah, K., Mao, Y., Chaudhary, V.: Tuxedo: A peer-to-peer caching system. In: Proc. of 2003 Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (2003)
43. Sivasubramanian, S., Alonso, G., Pierre, G., van Steen, M.: GlobeDB: Autonomic data replication for Web applications. In: Proc. of 14th Int'l World Wide Web Conf. Chiba, Japan (2005)
44. Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: Analysis of caching and replication strategies for Web applications. IEEE Internet Computing **11**(1), 60–66 (2007)
45. Sivasubramanian, S., Szymaniak, M., Pierre, G., van Steen, M.: Replication for Web hosting systems. ACM Computing Surveys **36**(3) (2004)
46. Szymaniak, M., Pierre, G., van Steen, M.: Latency-driven replica placement. IPSJ **47**(8) (2006)
47. The Apache Cocoon project (2007). http://cocoon.apache.org/
48. Tolia, N., Satyanarayanan, M.: Consistency-preserving caching of dynamic database content. In: Proc. of 16th Int'l World Wide Web Conf., pp. 311–320 (2007)
49. Vakali, A., Pallis, G.: Content delivery networks: Status and trends. IEEE Internet Computing **7**(6) (2003)
50. Williams, A., Arlitt, M., Williamson, C., Barker, K.: Web workload characterization: Ten years later. In: S. Chanson, X. Tang, J. Xu (eds.) Web Content Delivery. Springer (2005)
51. Yuan, C., Chen, Y., Zhang, Z.: Evaluation of edge caching/offoading for dynamic content delivery. IEEE Transactions on Knowledge and Data Engineering **16**(11) (2004)

52. Zhao, W., Schulzrinne, H.: DotSlash: Handling Web hotspots at dynamic content Web sites. In: Proc. of IEEE Global Internet Symposium. Miami, FL (2005)
53. Zhao, W., Schulzrinne, H.: Enabling on-demand query result caching in DotSlash for handling Web hotspots effectively. In: Proc. of Int'l Workshop on Hot Topics in Web Systems and Technologies. Boston, MA (2006)

# Chapter 5
# Caching Techniques on CDN Simulated Frameworks

Konstantinos Stamos, George Pallis and Athena Vakali

## 5.1 Introduction

It is evident that in the new Web era, content volume and services availability play a major role, leaving behind typical static pages which have solely text and images. The majority of the business oriented service providers are concerned for the Quality of Services (QoS), in terms of content delivery. In this context, proxy servers and Content Delivery Networks (CDNs) have been prosposed as different technologies, dealing with this concern. Their common goal is to bring content close to the users, reducing the response time.

Both technologies demonstrate different advantages and disadvantages. CDNs are characterized by robustness in serving huge amounts of requests and content volumes. However, their main shortcoming is that due to replication and distribution cost, replica placements should be static for a large amount of time. This leads to unoptimized storage capacity usage since the surrogate servers would contain redundant, possibly outdated, or unwanted content. On the other hand, proxy servers adapt content caching according to varying access patterns, using cache replacement algorithms. However, proxy servers do not scale well for serving large volumes of data or user populations. In an effort to combine the advantages of both, earlier recent work [2, 20, 29, 30] investigated different approaches that enable Web caching in CDNs, taking proxy servers' characteristics into account. As new caching ideas emerge, the need for a CDN testbed, suitable for performance evaluation and stress testing, becomes evident. Such a testbed should provide a networking environment incorporating CDN components, clients, traffic, and sufficient support for caching schemes deployment.

While the ideal case would be to examine caching schemes in real networks and CDNs, this is not always feasible or appropriate. Setting up a real CDN environment

Konstantinos Stamos
Department of Informatics, Aristotle University of Thessaloniki, e-mail: kstamos@csd.auth.gr

George Pallis
Department of Computer Science, University of Cyprus, e-mail: gpallis@cs.ucy.ac.cy

Athena Vakali
Department of Informatics, Aristotle University of Thessaloniki, e-mail: avakali@csd.auth.gr

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

from scratch is unfeasible since it introduces high infrastructure cost. Moreover, its configuration is a cumbersome task because it involves many parameters (traffic patterns, link speeds, network topologies, and protocols). Incorporating a new caching scheme requires large scale modifications to the execution environments of the various network elements. Furthermore, commercial CDNs are of proprietary nature and they are not usually accessible for research purposes. Finally, it is not straightforward to carry out experimentation in a real world framework, since it involves uncontrollable events (such as random noise and external network traffic), rendering the experiments unreproducible.

To overcome the difficulties imposed by the real world models, one may build simulated models. A simulated model, in our case a Web caching enabled CDN, introduces a new set of challenges. Dealing with the model itself, balance between accurate real world model representation and reasonable resources management (execution times and memory consumption) must be achieved. Furthermore, the model should provide base for incorporating CDN components, clients, traffic, services, content types, and especially caching schemes. The variety of possible network configurations and diversity of the caching schemes impose a large tree of implementation cases. Therefore the best choice is to adopt an open architecture, by maintaining a reasonable level of abstraction in the simulated entities.

Currently, there is quite limited number of CDN simulation environments and there is no standard roadmap for a practitioner to design and implement such a complex environment. The motivation of this chapter originates to these difficulties which emphasize the need for developing widely available and open CDN simulation environments. More specifically, the core contributions of this chapter are:

- *To provide sufficient background* for issues related to Web caching in the context of CDNs;
- *To identify the simulation requirements* of a Web caching enabled CDN;
- *To analyze and model the simulation* of various caching schemes in an actual CDN simulator; and
- *To suggest a roadmap* for the practitioner who would like to clarify performance issues related to such simulated frameworks.

In summary, the main goal of this chapter is to offer a solid design methodology and share implementation experiences, while covering most of the topics related to Web caching in a CDN simulation framework.

The rest of this chapter is structured as follows: we start by presenting issues related to the content delivery in Web via CDNs and proxy servers. Then, the potential of integrating caching characteristics of both CDNs and proxy servers are examined. A categorization of dynamic content along with several techniques are provided, followed by solutions to the problem of cache consistency. We continue with an in depth examination on how the mentioned caching schemes can be modeled and implemented in a simulated environment.

## 5.2 Content Delivery on the Web

Distributing information to users over the Internet in an efficient and cost-effective manner is a challenging problem. Web data caching and replication techniques have become key practices for addressing this problem, due to their ability to offer increased scalable solutions [25]. *Web caching* is mainly implemented by proxy servers, whereas *content replication* is the main practice on CDNs. Broadly speaking, the intention of Web caching and content replication is to shift the workload away from overloaded content providers and satisfy user requests from the intermediaries (proxy servers or CDN servers). Internet Service Providers (ISPs) use proxies to store the most frequently or most recently requested content. In addition, Web content providers may sign a contract with a CDN provider (e.g. Akamai) in order to offer their sites content over the CDN servers. In the following subsections, we overview the main characteristics of these two intermediary infrastructures for the Web.

### *5.2.1 Proxy Servers*

Proxy servers are deployed by ISPs to deal with increased Web traffic and optimize the content delivery on the Web [33]. In particular, proxy servers act as an intermediator between users and content providers, serving user requests from local storage. Users make their connections to proxy applications running on their hosts. At each request, the proxy server is contacted first to find whether it has a valid copy of the requested object. If the proxy has the requested object and it is updated, this is considered as a cache hit; otherwise a cache miss occurs and the proxy must forward the request on behalf of the user. Upon receiving a new object, the proxy services a copy to the end user and keeps another copy to its local storage.

Thus, the intermediate caching of objects reduces bandwidth consumption, network congestion, and network traffic. Also, because it delivers cached objects from proxy servers, it reduces external latency (the time it takes to transfer objects from the origin server to proxy servers). Finally, proxy caching improves fault-tolerance because users can obtain a cached copy even if the remote server is unavailable or uncacheable.

On the other hand, using a shared proxy cache has three significant drawbacks: If proxy is not properly updated, a user might receive stale data, and, as the number of users grows, content providers typically become bottlenecks. Furthermore, caching is problematic in terms of not improving availability during “flash crowd” events. The third drawback is related to the limited system resources of cache servers (i.e. memory space, disk storage, I/O bandwidth, processing power, and networking resources).

The above problems stem from the fact that proxy servers have been designed to work on a local basis. Thus, when a proxy server cannot satisfy a user request (cache miss), it should connect with the underlying Web content provider in order to fetch

the requested content. However, this may lead to Denial of Service (DoS), since Web content provider cannot serve a huge amount of requests (each Web content provider supports a limited number of HTTP connections). Moreover, the communication between a Web content provider and a proxy server may cause increased latency. For instance, consider the scenario where a user from Australia requests a Web page, and its Web content provider is located in USA. In such a case, a large number of TCP connections should be setup in order to communicate the proxy server with the content provider.

### 5.2.2 *Content Delivery Networks*

Figure 5.1 depicts how content is delivered on the Web using proxy and CDNs infrastructure. In case of cache misses, the proxy servers communicate with CDN servers in order to fetch the requested content. Specifically, a CDN maintains multiple *Points of Presence* (PoP) with Web server replicas (called *surrogate servers*) that store copies of the same content, and uses information about the user and the content requested to "route" the user request to the most appropriate site. The customers of a CDN are organizations that wish to offer their site content to a geographically

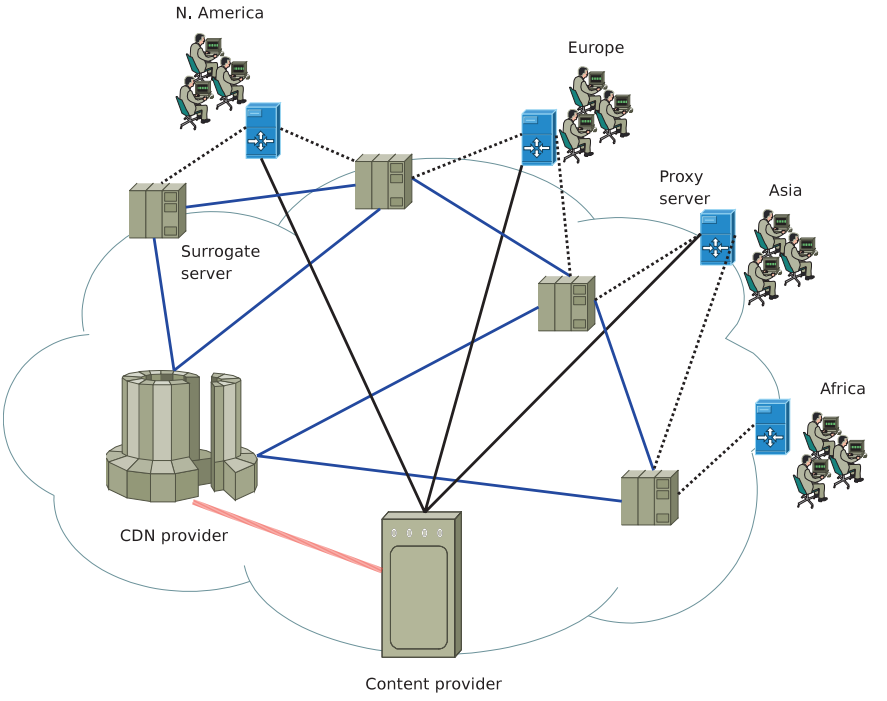



**Fig. 5.1** Content delivery on the Web

**Table 5.1** Proxy servers vs. CDNs

| Features | Proxy Server | CDN |
|---|---|---|
| Key practice | Web caching | content replication |
| Cached content | dynamically changes; content requested by users of an ISP | predefined content from the CDN-supported content providers |
| Scalability | low | high |
| Performance | vulnerable to flash crowd events | stable; suitable for resource-hungry applications (e.g. streaming media) |

distributed and potentially large audience. A CDN usually co-locates its surrogate servers within strategic data centers, using multiple network providers, on a globally distributed basis. Table 5.1 summarizes the main difference between proxy servers and CDNs. A comprehensive taxonomy with a broad coverage of CDNs in terms of organizational structure, content distribution mechanisms, request redirection techniques, and performance measurement methodologies can be found in Chap. 2 of this book.

## 5.3 Emerging Web Data Caching Techniques in CDNs

CDNs host distributed global information resources which are related to a large spectrum of applications. Users interact with (or within) companies, organizations, governmental agencies, and educational or collaborative environments. The popularity of the CDNs originates from its potential to efficiently deliver dynamic, distributed, heterogeneous, and unstructured data all over the world. Therefore, the need of various Web data caching techniques and mechanisms on CDNs has become obligatory towards improving information delivery over the Web.

### *5.3.1 Caching in CDNs*

As we mentioned in the previous Section, Web caching and content replication have been developed as two distinct approaches in order to meet the increasing demand of user requests:

- *Web caching approach*: Proxy servers store the Web objects into their caches. However, the cached objects are determined by a cache replacement policy. The cache replacement policies refer to deciding which objects will evict from the cache to accommodate new objects. In such a policy, each object is defined by a "value", the so-called *cache utility value* (CUV). The objects with the smallest

utility outcome will be the first candidates to evict from the cache. Podlipnig and Bszrmenyi in [23] conducted an extended survey of the existing cache replacement strategies.

- *Content replication approach*: Surrogate servers keep replicas of the Web objects on behalf of content providers. Contrary to proxy servers, the replicated content in CDNs remains static.

However, content replication practices of CDNs include inherent limitations. The major limitation is that a CDN infrastructure does not manage the replicated content in an efficient way. Moreover, replica placement is static for a considerable amount of time. The static nature of the outsourced content leads to inefficient storage capacity usage since the surrogate servers cache may contain unnecessary objects after a period of time. As a result, if user access patterns change, the replicas in surrogate servers could not satisfy the user requests.

A solution to the above issue would be to integrate both caching and replication policies to the storage space of surrogate servers. The experimental results reported by Stamos et al. [30] show that an integration scheme outperforms the stand-alone Web caching and static content replication implementations.

To formally define the integration approach, consider a Web site representative $W$ who has signed a contract with a CDN provider. The Web site contains $N$ objects initially located only at the content provider (outside of the CDN). The total size of $W$ is $W^s$ and is given by the following equation:

$$W^s = \sum_{k=1}^{N} U_k^s, \tag{5.1}$$

where $U_k^s$ is the size of the $k$-th $(1 \leq k \leq N)$ object.

Let $M$ be the number of surrogate servers consisting the CDN. Each surrogate server $M_i$ $(1 \leq i \leq M)$ has a total cache size $M_i^s$ dedicated for replicating the content of $W$. The original copies are located in the content provider. For simplicity, we consider that the surrogate servers are homogeneous (same storage capacity $M_i^s = M^s$ $(1 \leq i \leq M)$) and do not contain content from other Web sites.

As depicted in Fig. 5.2, the cache of surrogate server could be partitioned into two partitions:

- *Static cache partition*: Dedicated for static content replication. To formally define the static cache partition, we consider that its size is a percentage $r$ $(r \in [0..1])$ of $M^s$. Therefore, the replicated objects, in static cache of a surrogate server, obey the following constraint:

$$\sum_{k=1}^{N} (f_{ik} U_k^s) \leq r M^s, \tag{5.2}$$

  where $f_{ik}$ is a function denoting whether an object $k$ exists in the cache of surrogate server $i$. Specifically, $f_{ik} = 1$, if the k-th object is placed at the i-th surrogate server and $f_{ik} = 1$, otherwise. The content of the static cache is identified by applying a content replication algorithm. A wide range of content replication

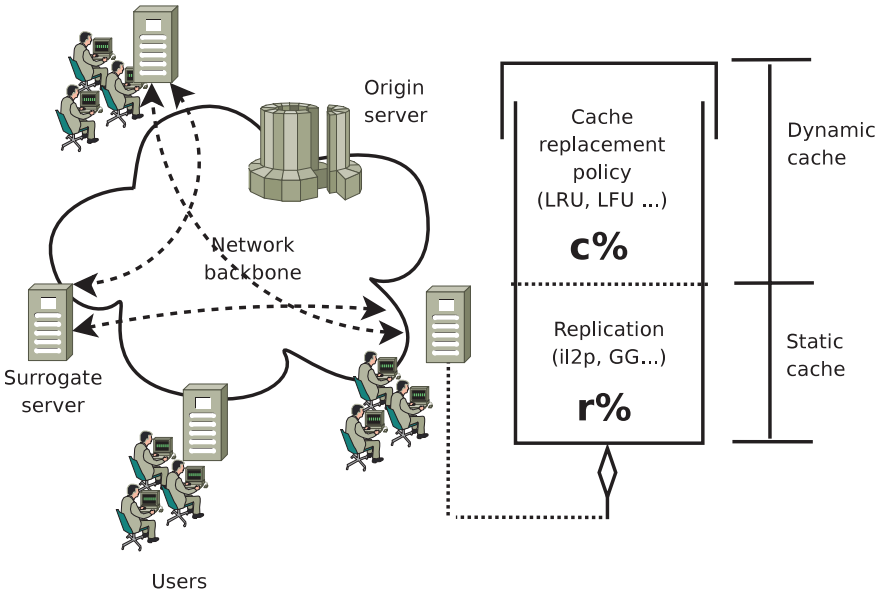



**Fig. 5.2** Integrating caching in a CDN

algorithms have been proposed in literature [12, 19, 21, 32, 37]. Kangasharju et al. [12] use four heuristic methods: (1) random, (2) popularity, (3) greedy-single, and finally (4) greedy-global. The experiments show that the greedy-global outperforms all other approaches. However, the greedy approaches are not feasible to implement on real applications due to their high complexity. Tse [32] study the content placement problem from another point of view. Specifically, the author presents a set of greedy approaches where the placement is occurred by balancing the loads and sizes of the surrogate servers. A quite similar approach is also presented in Zhuo et al. [37]. Pallis et al. [21] present a self-tuning, parameterless algorithm (called Lat-cdn) for placing outsourced objects in CDN surrogate servers, which is based on network latency. Finally, in [19], Pallis et al. partition the content placement placement problem into two sub-problems. The first one defines the pairs of outsourced object - surrogate server which achieve the lowest latency. The second one determines which objects to replicate based on the users workload. This approach is called il2p.

- *Dynamic cache partition*: Reserved for Web caching using cache replacement policies. To formally define the dynamic cache partition, we consider that the size reserved for dynamic caching is a percentage $c$, $(c \in [0..1])$ of $M^s$. More specifically, the stored objects respect the following storage capacity constrain:

$$\sum_{k=1}^{N} (f_{ik} U_k^s) \leq cM^s \tag{5.3}$$

Initially, the dynamic cache is empty since it is filled with content at run-time according to the selected cache replacement policy. Thus, the surrogate servers would have the replicas with the best CUV in their dynamic cache partition. Other than the traditional cache replacement policies (e.g. LRU, LFU), Aioffi et al. [1] use an on-line heuristic algorithm in order to decide whether to add a new content replica or remove an existing one. The proposed algorithm (called on-line MDCDN) is based on a statistical forecasting method, called Double Exponential Smoothing (DES). Taking the user demand variations into account, MDCDN predicts the future demand at each surrogate server. These predictions determine the CUV of the the cached objects. Chen et al. [6] use an application-level multicast tree as a cache replacement policy for each CDN surrogate server. Presti et al. [24] determine the CUV of replicas by a non-linear integer programming formulation. In [3], Bartolini et al. decide whether to add a new content replica or remove an existing one using a semi-Markov decision process.

Given the above cache segmentation scheme, the percentages $(r,c)$ must obey is the following:

$$r+c=1 \tag{5.4}$$

The challenge for such an approach is to determine the surrogate server size which would be devoted to caching and replication as well. In other words, we should determine the percentages $(r,c)$. Considering that this problem is NP complete [2], several heuristic approaches have been considered to efficiently integrate static and dynamic cache in CDN surrogate servers. Bakiras and Loukopoulos [2] propose a greedy *hybrid* algorithm that combines an LRU cache replacement policy with static content replication on a CDN. More specifically, initially the storage capacity of each surrogate server is reserved for Web caching and at each iteration of the algorithm, objects are placed to surrogate servers maximizing a benefit value. The *hybrid* gradually fills the surrogate servers caches with static content at each iteration, as long as it contributes to the optimization of response times. Stamos et al. [29] have developed a placement similarity approach (the so called *SRC*) evaluating the level of integration of Web caching with content replication. According to this approach, a similarity measure is used to determine the surrogate server size which would be devoted to caching and replication. Finally, Pallis et al. [20] use a logistic sigmoid function in order to classify the surrogate server cache into two parts. The proposed approach, called R-P, classifies the replicas with respect to their quality values. In particular, the quality value of each replica is expressed by the users interest (increasing its value) or the lack of users interest (decreasing its value) for the underlying replica.

### *5.3.2 Caching Dynamic Content*

Dynamic content can be classified into three categories, as depicted in Fig. 5.3, based on how frequently Web objects change and whether these changes can be

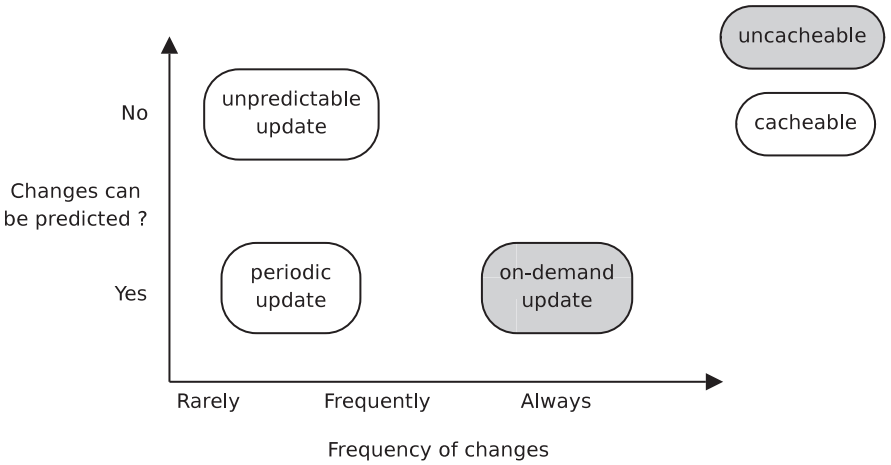



**Fig. 5.3** Categorization of dynamic content

predicted. The *periodic-update* category includes objects that the content provider updates at specified time intervals. For instance, consider a news Web page which is updated in every 5 min. The *on-demand-update* category consists of objects which are generated on demand and may have different attributes depending on the requesting user (e.g. the query forms). The *unpredictable-update* category includes objects that change unpredictably. The objects in *periodic-update* and *unpredictable-update* categories can be cached, whereas, the objects in the *on-demand-update* category are uncacheable.

Efficient distribution of dynamic content to end users is an important issue due to the growing number of dynamic data on the Web. A wide range of caching techniques have been proposed in order to accelerate the delivery of dynamic content to users [5, 27]. Fragment caching is an effective technique to accelerate current Web applications which usually generates heterogeneous contents with complex layout.

*A fragment can be defined as a portion of a Web page which has a specific theme or functionality and is distinguishable from the other parts of the page.* A Web page has references to these fragments, which are stored independently on the content provider or the surrogate servers. Challenger et al. [5] represent the relationships between Web pages and fragments by object dependence graphs.

The fragment-based approach has also been implemented in commercial CDN providers. For instance, the EdgeSuite network of Akamai is based on a fragment-based policy using the ESI (Edge Side Includes) specification accepted by the World Wide Web consortium. Specifically, the ESI specification defines an XML-based mark-up language for defining templates and identifying page fragments. A fragment-based policy is also used by the IBM Websphere [5], where the Web pages can be decomposed into a hierarchy of complex, atomic fragments.

Fragment-based approaches cannot be effectively applied on the objects which belong to the *on-demand-update* category, since these objects cannot be cached.

Specifically, they perform well if the temporal locality of requests is high and if the underlying database is updated rarely. Applications that do not exhibit these behavior require more sophisticated techniques [28]. Therefore, instead of caching fragments of Web pages, another approach is to replicate a full copy of the application code at the surrogate servers [26]. In such an approach (known as *Edge Computing*), each surrogate server may connect with a centralized database. So, all database queries are forwarded to the content provider. Although this technique allows to distribute the computations to generate pages, it is limited by the latency incurred for each query, and by the throughput bottleneck of the origin database [28]. To address this issue, another approach is to keep a partial replica of the database (known as *Content-Aware Caching (CAC)* approach). In such an approach, the application programmers can choose the data replication that are best suited for the application. This approach can yield considerable gains in performance and availability, provided that the selected strategies are well suited for the application [28]. However, this is quite difficult since it requires significant insight of the application programmers in domains such as fault-tolerance and weak cache consistency. In this context, another technique (known as *Content-Blind query Caching (CBC)*) has been proposed to cache the results of database queries at the surrogate servers. Consistency of cached results must be maintained when the underlying database is updated. This technique allows to reduce the database query latency since a number of queries can be answered locally. The total system throughput is also increased because less queries are addressed to the content provider [28].

### *5.3.3 Cache Consistency Mechanisms*

Considering the dynamic nature of Web content, an important issue that must be addressed by CDNs is the consistency maintenance [36]. To prevent stale content from being transmitted to end users, the surrogate server must ensure that the locally cached data is consistent with that stored on servers. The exact cache consistency mechanism and the degree of consistency employed by a CDN depends on the nature of the cached data. Consequently, a CDN should ensure the consistency of replicas with the content provider by employing suitable mechanisms.

The problem of consistency maintenance has been well studied in the context of proxy servers. Particularly, in proxy servers the Time to Live (TTL) concept is widely used [26]. According to this, the content provider, when serving a cacheable object to the proxy, supplies an explicit TTL value. Then, the proxy considers that object valid during its TTL period. In the context of a CDN, the TTL concept should be employed in each individual surrogate server. In such a case, each surrogate server is responsible for maintaining consistency of data stored in its cache. Therefore, each one interacts with the content provider to do so independently of the other surrogate servers. However, this approach is impractical/unfeasible to be implemented in a large-scale network infrastructure. Considering that a typical CDN usually consists of a large number of surrogate servers (i.e. the Akamai – the leading

CDN provider – has more than 25,000 surrogate servers around the world), the content provider will need to individually interact with a large number of surrogate servers. Thus, such an approach is not scalable from the perspective of the content providers.

To formally define the degree of consistency that a CDN can support, let $CP_k^t$ and $S_k^t$ denote the version of the object $k$ at the content provider and the surrogate server respectively at time $t$. In this context, an object $k$ is said to be:

- *Strongly consistent* with that at the content provider if the version at the surrogate server is always up-to-date with the content provider. That is, $\forall$t, $CP_k^t = S_k^t$. Strong consistency ignores network delays incurred in propagating updates to the surrogate server.
- *Delta consistent* with that at the content provider if the version at the surrogate server is identical for $\Delta$ time units, where $\Delta$ is a configurable parameter. That is, $\forall$t, $\exists\tau\ 0 \leq \tau \leq \Delta$ such that $CP_k^{t-\tau} = S_k^t$.
- *Weak consistent* with that at the content provider if the version at the surrogate server is not always up-to-date with the content provider.

A consistency degree may also be defined for multiple objects; it is known as *mutual consistency*. To formally define this degree of consistency, consider two objects $a$ and $b$ that are related to each other. Cached versions of objects $a$ and $b$ at time t ($S_a^t$ and $S_b^t$ respectively) are defined to be mutually consistent in the time domain ($M_t$-consistent) if the following condition holds: If $CP_a^t = S_a^{t_1}$ and $CP_b^t = S_b^{t_2}$ then $|t_1 - t_2| \leq \delta$, where $\delta$ is the tolerance on the consistency guarantees. For $\delta = 0$, it requires that the objects should have simultaneously existed on the content provider at some point in the past.

There exists a wide range of mechanisms [16, 17, 31] that have been used to provide efficient cache consistency in CDNs. These can be broadly categorized as follows:

- Server-driven consistency (also referred to as *server-based invalidation*): the content provider notifies the surrogate servers when the content changes. This approach substantially reduces the number of control messages exchanged between the content provider and the surrogate server since messages are sent only when an object is modified. However, this results in inefficient usage of the distribution network for content delivery and inefficiency in managing consistency at surrogate servers, since the content provider should maintain a list of all surrogate servers that cache the object. Several new protocols have been proposed recently to provide consistency using server-based invalidation. Web cache invalidation protocol (WCIP) [14] is one such proposal for propagating server invalidation using application-level multicast. Web Content Distribution protocol (WCDP) [31] is another proposal that enables server-driven consistency. Using the WCDP, the content provider can dynamically control the propagation and visibility of an object update. WCDP supports different levels of consistency (i.e. strong, delta, weak, and mutual).
- Client-driven consistency (also referred to as *client polling*): the updated version of a Web object is delivered to all the surrogate servers whenever a change

is made to the object at the content provider. The advantage is that it does not require any list to be maintained at the content provider. However, such an approach may generate significant levels of unnecessary traffic if the objects are updated more frequently than accessed. Mikhailov and Wills [16] proposed a client-driven consistency approach, called MONARCH (Management of Objects in a Network using Assembly, Relationships and Change cHaracteristics). The MONARCH guarantees the cache consistency by collecting snapshots of content from sites of interest. This content is then used as input to a simulator to evaluate several cache consistency policies over a range of access patterns.

- Leases approach: Consistency is achieved by associating leases with each object that get replicated to surrogate servers. Specifically, lease is a time period where its duration denotes the interval of time during which the content provider agrees to notify the surrogate server if the object is modified. After the expiration of the lease, the surrogate server must send a message requesting renewal of the lease. This approach is a combination of server-driven and client-driven consistency. If the lease duration is zero, the cache consistency scheme degenerates into pure client-driven consistency. On the other hand, if the lease duration is infinite, the cache consistency scheme degenerates into a pure server-driven consistency. The concept of a lease was first proposed in the context of cache consistency in distributed file systems [10]. The use of leases for Web proxy caches was first presented in [15]. Duvvuri et al. in [8] present extensions to the HTTP protocol in order to incorporate leases. Ninan et al. [17] presented a variation of the lease approach for CDNs, called cooperative leases, by using $\Delta$-consistency semantics. Specifically, $\Delta$-consistency requires that a cached version of an object is never out-of-date by more than $\Delta$ time units with its server version. The value of $\Delta$ determines the nature of the provided guarantee. Therefore, the larger the value of $\Delta$ is, the weaker the consistency is.

## 5.4 Caching Techniques on CDNsim

This section is focused on introducing cache replacement policies in CDNs by using an actual CDN simulator. For this purpose we use CDNsim[1] as the main paradigm. First of all, the necessity of such a simulated environment is investigated, along with other simulation solutions. Then the requirements of a simulated cache, in terms of scientific issues and resource requirements, are defined. Finally, several issues related to the actual development of such caching framework are discussed.

[1] http://oswinds.csd.auth.gr/∼cdnsim

### 5.4.1 The Need for CDN Simulated Environments

There have been several attempts to create private academic CDNs such as CoDeeN [18], Coral [9], and Globule [22] for research and every day purposes. However, the reproducibility of a given experiment is impossible since we are dealing with real networks. Moreover, it is hard to implement and evaluate new policies due to the required large scale alterations of the whole system. Consequently, the necessity of a simulation environment for performing experiments, still remains. Towards this direction, there have been several implementations of a simulated CDN [4, 7, 12, 34] which fit the individual needs of each research work. Most of them do not take several critical factors into account, such as the bottlenecks that are likely to occur in the network, the number of sessions that each network element can serve (e.g. router, surrogate server) and ignore the TCP/IP protocol. Finally, the most important disadvantage is the unavailability of a platform for examining caching techniques in CDNs.

Filling this gap, CDNsim is developed as a general purpose CDN simulator. It is extensible and open source, written in C++ using the OMNET++ and INET libraries.[2] It is a parallel discrete event trace driven network simulation package that provides libraries, utilities, and interfaces for content delivery on the Web. CDNsim models a CDN including basic network components such as users, surrogate servers, content providers, and routers. It takes the characteristics of Internet infrastructure into account by simulating the TCP/IP. CDNsim has been designed to support research in broad-coverage CDN services. It has also the ability to simulate Peer-to-Peer (P2P) services as well as various internetwork configurations. The experience gained from the development of such a tool is reported in the following paragraphs.

### 5.4.2 CDNsim's Caching Framework Requirements

This subsection includes the specifications of the surrogate servers' caches in CDNsim which are taken into account at design time, and both research and performance issues are addressed. A requirement is to support the integrated caching schemes as reported in these works [2, 20, 29, 30]. Cache consistency mechanisms must also be supported. Moreover, it is required to support complex content types such as video, and treat dynamic content by fragmentation. Finally, support for non cacheable content should be enabled.

The diversity of cache replacement algorithms leads to confusing branches of implementation cases. For instance, LFU and SIZE use different attributes for replacing objects. Therefore the detection of a common denominator is necessary. More specifically, given a set of primitive generic operations and content types one should be able to implement any flavor of the mentioned methodologies. Therefore, a strict requirement is to prepare a set of interfaces that can be used as building blocks of

[2] http://www.omnetpp.org/

the various caching schemes. An appropriate exposure of the cache content to the network should be considered. This would enable both user access for downloading content and CDN access for management.

The execution of a CDN simulation includes high activity in the surrogate servers' caches. A typical simulation scenario involves a set of users performing requests for objects (Web pages, multimedia content, etc.) to the CDN. The surrogate servers manage the content of their caches and attempt to satisfy the requests. By increasing the number of requests, the required CPU time (of the host running the simulation) is increased as well. Moreover, an increment to the caches' capacity leads to more objects being stored and thus to higher RAM requirements. It is evident that depending on the various simulation scenarios and configurations the caches may become performance bottlenecks.

The primary performance concern is to optimize the cache function in terms of CPU requirements. In order to define a satisfactory performance threshold, the available operations of a cache need to be identified:

- *Search*: This operation involves the procedure of browsing though the cache's contents until a specified object is found. For instance, a user requests a Web page and the surrogate server examines the cache to check whether it is stored or not. The performance penalty of such operation depends on the organization of the content. Generally, the *search* complexity at an average case should always be better than $O(n)$, where $n$ refers to the number of objects residing in cache. This is critical since the *search* operation may be executed several million times during a simulation, involving caches with several thousands of objects.
- *Insertion*: This operation refers to the procedure of inserting a new object in cache. It is responsible to maintain the proper organization of the cache and update other attributes such as the remaining storage space. Likewise, it must perform better that $O(n)$. Every cache replacement policy includes *insertion* operations as part of their algorithm. It is expected to be executed many times during a simulation and therefore it is an essential optimization parameter.
- *Deletion*: It is the procedure to remove a specific object from the cache. Every cache replacement algorithm replaces objects by performing *deletion* operations to free up storage space. Therefore, $O(n)$ performance should be upper bound.
- *Update*: The case of an object's update can be expressed as a combination of *deletion* of the previous version object and *insertion* of the updated object. The *update*. operation takes place when cache consistency is applied.

Summarizing, the cache speed is closely related to the content organization in the RAM of the host. Most of the cache replacement algorithms include the *Search-Insertion-Deletion-Update (SIDU)* operations. Therefore, the optimization requirement is the design of efficient content manipulation operations.

The secondary performance concern is the optimization of the memory footprint. It is important that the simulation's execution environment fits in RAM. Absolutely no memory must be swapped to the hard drive of the host, or else the execution performance will be reduced. Therefore, it is required to adopt a conservative design that saves memory. Simulating large networks with many surrogate servers, caches,

and objects require several gigabytes of RAM. However, the memory optimization is usually a secondary requirement because the aforementioned problem can be easily solved with sufficient RAM.

### 5.4.3 CDNsim's Cache Architecture

In order to meet the previously discussed requirements, we define an architectural design. This design is used for the actual implementation of CDNsim. It is an effort to shape an abstraction to the real entities in a CDN. More specifically we model the cache organization of a CDN as a 3-level architectural component, depicted in Fig. 5.4.

The first level deals with the notion of content by ignoring all the special characteristics that identifies it. *Cacheable content*, in general, is considered as raw *fragments*, namely objects. The objects are stored into the cache, which is merely a storage medium that keeps up-to-date information such as available storage capacity and provides the SIDU interface, as discussed previously. The objects can be classified into two categories: (a) volatile objects and (b) non-volatile objects. The first term refers to the objects that are stored inside the cache suggesting static caching, while the later defines the objects devoted for cache replacement algorithms. By flagging the objects, the integrated static and dynamic caching algorithms described in [2, 20, 29, 30] can be implemented.

One level up, we deal with the organization of the objects. The specific characteristics of each object are ignored. In this context, a set of cache replacement policies

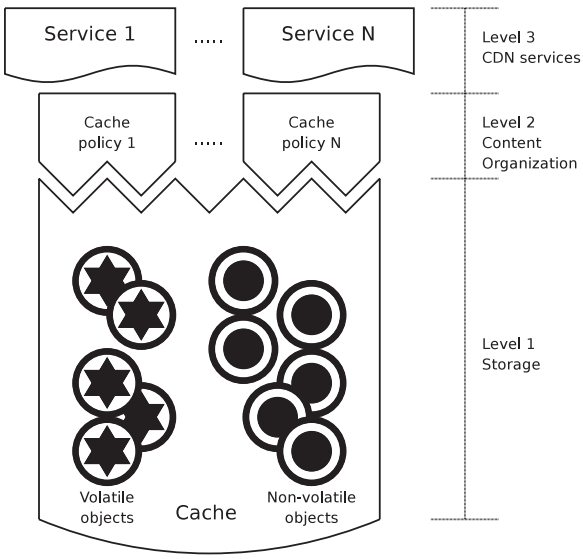



**Fig. 5.4** 3-level cache organization

is defined, managing the content of the cache. Each policy maintains an ordering of the objects according to a set of attributes. The attributes may refer to objects' characteristics such as size, TTL, and last access time. For instance, the LRU cache policy should maintain a buffer that keeps the objects sorted according to last access time, enabling the replacement of the objects. The upper level uses the SIDU interface which is provided by the cache replacement policies. At this point, we may introduce the concept of cache partitioning and especially *Static cache partition* and *Dynamic cache partition*.

The third level, defines a higher level logic of content management. In contrast to the lower levels, we are interested in the actual content type and special characteristics of the objects. Each content type effectively can be expressed as a group of objects (fragments) forming a hierarchy. For instance, a dynamic page can be expressed as a set of objects, representing identified cacheable page fragments. Therefore, the object abstraction of the lower levels provides a unified approach for dealing with the content diversity. Another high level construct is the service, which represents any operation at CDN level. The services can be of internal use only (e.g. surrogate server cooperation) and are available only to the CDN. Otherwise, the services are public, such as dynamic Web pages manifestation and serving to the users. Therefore, cache consistency can be addressed by implementing appropriate services that manage the cache content. Additionally, *uncacheable content (on-demand-update)* can be handled by implementing services capable of composing content on-the-fly.

Table 5.2 summarizes the mapping between various caching issues and the architectural components in CDNsim. Specifically, CDNsim supports directly the static, dynamic, and integrated caching scheme. Each can be modeled as partition of the cache. CDNsim offers generic input for any kind of static replica placement algorithm, while by default it supports the LRU dynamic caching. Cache consistency is managed by a set of services that signal the various content updates and commit the changes. Cache consistency lays at level-3. By default CDNsim implements strong cache consistency. Complex content types such as audio, video, Web pages, and streaming content are supported in the form of objects' hierarchies. Each type can be represented by a set of objects that can be cached. Combined with cache

**Table 5.2** Mapping of caching issues to CDNsim's architectural components

| Caching Issue | Architectural Component | CDNsim Default |
|---|---|---|
| Static/Dynamic cache partitioning | Cache partition, volatile/non-volatile | Generic support/LRU |
| Strong/Delta/ Weak/Mutual consistency | Service | Strong |
| Complex content | Object hierarchy | Video, Web pages, etc |
| Unpredictable/periodic – cacheable dynamic content | Object | Generic support |
| On-demand-update – uncacheable dynamic content | Service | Unspecified |

consistency services we can enable the caching of dynamic content updates. Uncacheable content is dealt separately by implementing a set of specialized services at level-3.

### 5.4.4 Implementation Considerations

This subsection covers the actual implementation of a caching scheme in CDNsim, which can be of use to an actual software practitioner. Specifically, we use a representative example where the surrogate servers contain caches with partitioned storage capacity for static and dynamic caching, as reported by Stamos et al. [30]. The static part of the cache is filled using a replica placement algorithm while the dynamic part of the cache obeys to the LRU. We select LRU as it is a well known and easy to follow algorithm. Our goal is to implement such a cache by following the described architecture, while keeping up with the performance requirements.

**Level 1.** The primary concern in this level is to implement the classes object and cache. Since we plan to create a caching scheme that incorporates dynamic and static caching, a class of volatile objects, and a class for non-volatile objects is defined. We consider an object belonging to the non-volatile class, to be stored in the static part of the cache. Specifying an object as volatile leads to be stored at runtime in the dynamic part and potentially be removed by a cache replacement policy.

Low memory consumption is defined as a secondary requirement. An implementation approach needs to be followed that balances information compression and the ability to perform cache replacement:

- *Full compression – no information*: Bloom filters [13] are proposed as a method for simulating a cache. A bloom filter is a bitarray that packs information. In the context of a cache, this information refers to the ids of the objects. The operations permitted in the bitarray are: reading, enabling, and disabling bits. A set of different hash functions map each inserted id to a set of bits in the array. This approach has several advantages. The operation is fast, because it includes AND and OR operations, native to the CPU of the host and the memory consumption is low. However, the use of hash functions causes collisions. For different object ids the same bits may be suggested leading to inaccurate content description. Furthermore, the information related to each object is stripped. We cannot store attributes such as last access time and thus we are unable to implement cache replacement algorithms like LRU.
- *Partial compression – partial information*: As the name suggests, this approach makes partially use of the bloom filters technique and the full representation of the objects [13]. In full representation, each object is an actual C++ object that stores all the necessary attributes, like the size and the last access time. However, the bloom filters result in information loss and thus LRU cannot be implemented.
- *No compression – full information*: The full representation is suitable for implementing LRU since all the objects' attributes are available. Consider the following example, we need 16 bytes (4 for the id, 8 for the last access time and 4 for

the size, in a 32 bits environment) we still can store roughly about 130 million objects in 2 GB RAM. Therefore, despite the increased memory usage several millions of objects can be managed using a standard host. The use of lossless compression schemes is prohibited, because they involve time consuming decompression and re-compression leading to performance penalty. Therefore, the suggestion is to use a $1-1$ mapping of ids-objects. All the SIDU operations are $O(1)$.

**Level 2.** It manages the organization of the content in the form of cache policies. Two distinct cache policies are identified, the static caching and the LRU. The content of the static part of the cache remains unchanged by default, therefore, we are not required to maintain some kind of ordering in the stored objects. A $1-1$ mapping of the object ids to the objects will suffice and the SIDU performance is $O(1)$.

On the other hand, LRU requires special treatment. LRU removes the least recently used objects in favor of others most recently used. Therefore, It is necessary to enable an ordering of the objects according to the last access time. CDNsim uses a buffer containing objects sorted by the last access time, depicted in Fig. 5.5. This buffer does not store the actual objects, which is the responsibility of the cache itself. Instead, the buffer is used only as an ordering structure of object ids. The *search* operation requires $O(n)$ in the worst case, since we have to browse through the buffer to detect the required object. The *insertion* operation requires $O(1)$; the new object is inserted at the beginning of the buffer (head) and if necessary, several objects at the end of the buffer (tail) are being removed to free up storage capacity. The *deletion* operation requires $O(1)$ time. As long as the object for deletion is searched, we just crop the object from the buffer without the need for adjusting any ordering. Finally the *update* operation is also $O(1)$. Provided that we have searched the object, we update without changing the ordering. It is evident that the *search* operation is involved in most of the SIDU operations. The time complexity of the search in the worst case is $O(n)$ and may reduce the speed of the cache. However, this can be safely ignored for two reasons: (a) data structure can be cached out easily in the CPU cache, leading to small performance overhead and (b) we tend to search for recently used objects, so only a few objects are being checked at the beginning of the buffer. Although in practice it gives satisfactory performance, we can further improve the *search* operation. This can be achieved by maintaining an extra index that points directly to the objects inside the buffer achieving $O(1)$ performance.

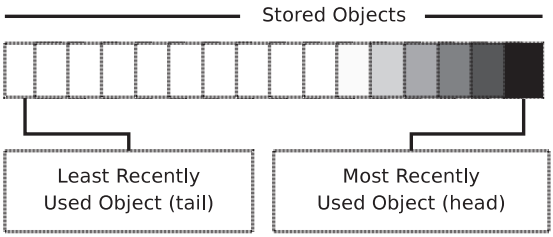



**Fig. 5.5** The LRU buffer

Another issue is the possible resource deadlocks and content inconsistency by accessing simultaneously the content. This is handled by creating private copies of the requested content. Therefore, the services of the upper level deal only with private copies of the content.

**Level 3.** At this level we are free from the caching issues, as they are handled by the lower level. The software practitioner is free to implement services that serve and distribute content.

### *5.4.5 Indicative Experimentation Results*

The effectiveness of the storage partitioning scheme for Web caching and content replication is supported by a set of experiments conducted in this work [30]. In this subsection we demonstrate briefly a few results that capture the behavior of this scheme. The examined criteria are:

- *Mean response time*: This is the expected time for a request to be satisfied. It is the summation of all request times divided by their quantity. Low values denote that content is close to the end user.
- *Response time CDF*: The Cumulative Distribution Function (CDF) in our experiments denotes the probability of having a response times lower or equal to a given response time. The goal of a CDN is to increase the probability of having response times around the lower bound of response times.
- *Hit ratio*: It is defined as the fraction of cache hits to the total number of requests. A high hit ratio indicates an effective cache replacement policy and defines an increased user servicing, reducing the average latency.
- *Byte hit ratio*: It is the hit ratio expressed in bytes. It is defined as the fraction of the total number of bytes that were requested and existed in cache to the number of bytes that were requested. A high byte hit ratio improves the network performance (i.e. bandwidth savings, low congestion, etc.).

The tested caching schemes include the LRU, LFU, and SIZE algorithms at various levels of integration $(r,c)$ with static replication. The $(r,c)$, as already defined, represent the percentage of the storage capacity used for static replication and Web caching respectively. The used static replication algorithm is il2p [19] which takes the server load into account. Specifically, il2p using two phases selects which object should be placed and where. During the first phase for each object the appropriate surrogate is selected minimizing network latency. Given the candidate pairs of (object, surrogate server), at the second phase, the one that yields the maximum utility value (depended on server load) is selected. This selection process is iterated until all caches are full. For completion, the cases of full mirroring (entire Web site is copied to the caches) and empty disks (simulating the absence of CDN) are included. Meeting the experimentation needs, the Stanford's Web site[3] is used. In

[3] http://www.stanford.edu/∼sdkamvar/research.html

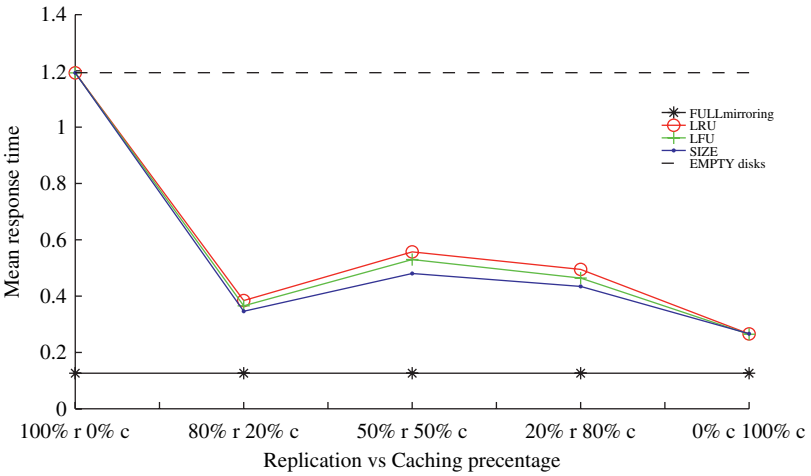



**Fig. 5.6** Mean response time

this context a CDN was built using CDNsim, simulating several thousand users and network elements.

To begin with, Fig. 5.6 illustrates how the mean response time of the requests is affected by modifying the level of integration between static replication and caching. Using static replication only ($r = 100\%$) we receive the highest mean response time. This can be explained by the fact that a fixed replica placement cannot cope with the changing users' demands for content. Two distinct performance peaks can be identified; the first for $r = 80\%, c = 20\%$ and the second for $c = 100\%$. Increasing the dynamic partition of the cache only by 20% (first peak) leads to significant performance improvement. This behavior is logical since we keep a part of the cache open for new content to be stored, based on the users' requests, while maintaining a sufficient amount of static replicas suggested by il2p. As the percentage of the dynamic partition increases the good attributes of replication are gradually lost. This is caused by the fact that the cache replacement policies may remove usefull content which otherwise would be strategically placed by il2p. The caching scheme (second peak) appears to perform slightly better than the integrated scheme (first peak). A possible explanation is that by letting the entire storage capacity to be used by cache replacement, we allow the Web caching scheme to adapt effectively to the user traffic patterns. Moreover, the fact that the Stanford's Web site contains mostly small objects, leads to low performance penalty during a cache miss. However, caching is not the choice, since the CDN is converted into a proxy server including all the scalability problems a proxy server imposes. Another important observation is that all the cache replacement algorithms follow the same behavior, with SIZE to be slightly better. SIZE's superiority can be explained by the fact that more room for new objects is available leading to better cache size utilization.

In terms of hit ratio, depicted in Fig. 5.7, the same two peaks can be detected. Pure Web caching and the integrated schemes offer comparable results, while the

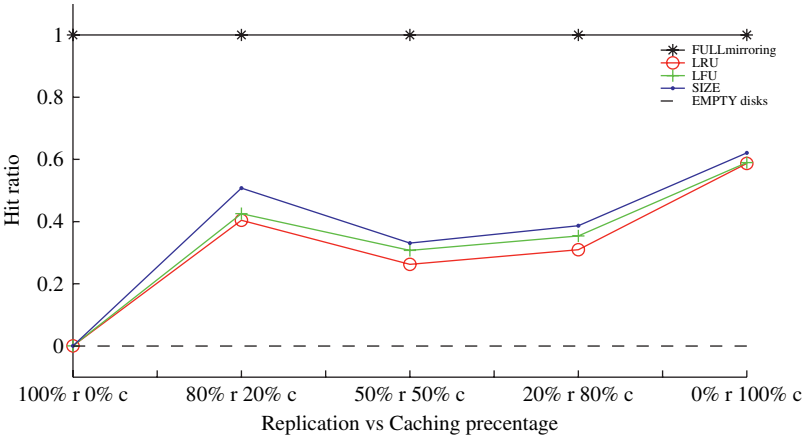



**Fig. 5.7** Hit ratio

static only replication does not keep up with. Fixed replica placement using the entire storage capacity suffers from low hit ratio since redundant content is outsourced and the placement is not optimal. The optimal placement cannot be achieved due to the changing users' requests pattern and the fact that it is a NP-complete problem. This reason also indicates why pure Web caching demonstrates slight performance superiority over the integrated scheme. The same behavior also exists in Fig. 5.8, illustrating the byte hit ratio.

A more accurate representation of the requests' satisfaction time distribution is presented in Fig. 5.9 for the first performance peak ($r = 80\%, c = 20\%$). Ignoring the ideal case of full mirroring, we observe that the integrated scheme (SIZE/il2p) outperforms (is the ceiling of all distributions) by achieving lower response times

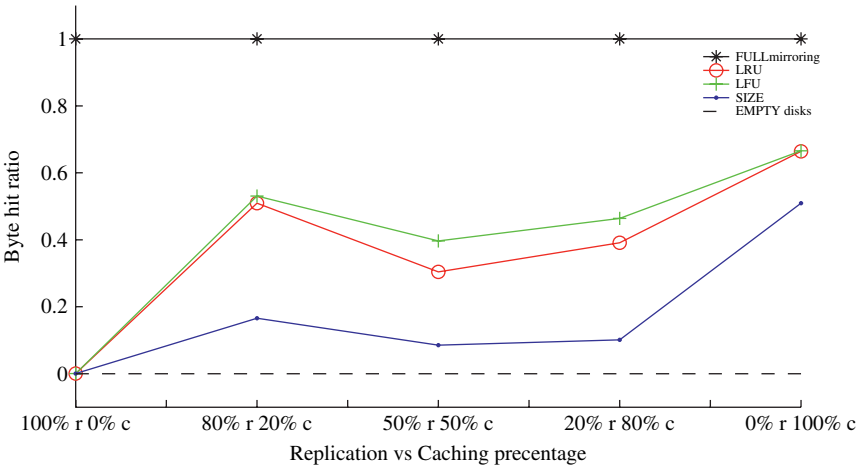



**Fig. 5.8** Byte hit ratio

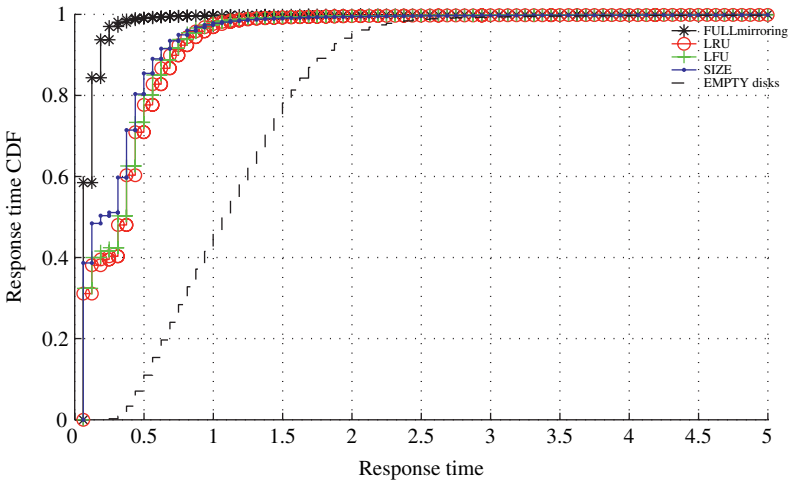



**Fig. 5.9** Response time CDF

than the other schemes. Here below we outline our observations by summarizing the results:

- There exists at least one performance peak belonging to the integrated caching and replication scheme.
- The integrated scheme may demonstrate reduced performance because of an inefficient replica placement.
- In special caches the caching only scheme perform as good as the integrated scheme, but it is not recommended since it inherits all the disadvantages of the proxy servers.

The conclusion is that there is a realistic room for performance improvement by implementing Web caching characteristics in a CDN.

## 5.5 Visionary Thoughts for Practitioners

CDNsim is a free and open tool available both for research purposes and commercial use. Two main target groups can be identified where CDNsim could be of great use; the CDN developers and the software practitioners interested in CDNs. In this section, we discuss several visionary thoughts on how CDNsim can be used by these groups.

Dealing with the first target group, CDN providers are interested in maximizing the benefit of their network infrastructure. To achieve this, the CDN developers design proprietary algorithms that manage the content effectively. The natural derivative of such activity is the creation of a new product. In the context of CDNs, the product is a new content delivery service, like streaming video, large files delivery,

etc. Although each service[4] may differ from the others in terms of functionality, a common set of periods in the life time of every service can be identified:

- *Before service release:* This period includes the development process of the service before its release to the users. CDNsim could be of use at the early development stages. It can be used to design and implement prototypes giving shape to the initial product ideas. Once the prototyping is done, it can be used to perform an *in vitro* evaluation of the performance and behavior under various network configurations and traffic patterns. CDNsim could significantly reduce the infrastructure investments during the stages of testing and prototyping until a certain level of maturity is reached. Then, evaluation is performed at the real CDN infrastructure. A real world example of the concept of prototyping and testing that could potentially be performed by CDNsim is the recent High Definition Video streaming by Akamai.[5]
- *After service release:* The service, by the time of its release to the wider public, should have passed a set of testing suites. Additionally, there is a set of documented conclusions about its behavior and performance. However, as the product is being used under untested circumstances, the behavior may divert from the initial conclusions. CDNsim may be used to reproduce a problematic or unexpected situation aiding the analysts to explain *why* an observed behavior is reached. Therefore, CDNsim could be used for continuous evaluation without disrupting the deployment of the service. Since the environment where a service runs is not static, CDNsim might act as a preventing mechanism of unwanted situations before they happen. For instance, the necessity of behavior prediction and disaster prevention is apparent before a worldwide broadcast of soccer world championship by Limelight Networks.[6]
- *Service evolution in time:* Eventually a service will reach a certain level of maturity, stability, and correctness. However, the service's "habitat" (network configurations, typical user populations, current technologies) is constantly evolving. A representative example is the increment of fast internet connections and the fact that IPv6 [11] will become a necessity since the available IP addresses are reducing. CDNsim could be used to perform a *what-if* analysis. How the service scales with larger user populations? Can the service and the existing infrastructure keep up with much faster connections currently not available? These questions could be addressed by CDNsim by setting up the respective network configurations. Failing to predict the long term evolution could result in loss of clients by not investing on upgraded infrastructure in time.

Dealing with the second target group, software practitioners are encouraged to extend the existing architecture to support the latest trend of algorithms. A visionary evolution of CDNsim could be a testbed that incorporates a complete suite of caching algorithms used for performance comparison and testing. Moreover,

[4] Using the term *service* we refer to a content delivery service in general.

[5] http://www.akamai.com/

[6] http://www.limelightnet.com/

CDNsim is able to run in parallel environments. The high performance computing researchers could find a testbed for implementing parallel algorithms in the context of CDNs. Therefore, some ideas concern the design and implementation of caching algorithms that take advantage of the new multi-core processors and the appliance of new more efficient data structures. Further research directions are outlined in the following section.

## 5.6 Future Research Directions

CDNsim might offer new perspectives for future research directions in the area of content delivery. Some indicative applications where CDNsim would be used as a simulation testbed could be the following:

- **Content Delivery Practices**: Several issues are involved in CDNs since there are different decisions related to where to locate surrogate servers, which content to outsource, and which practice to use for (selected content) outsourcing. It is obvious that each decision for these issues results in different costs and constrains for CDN providers. In this framework, CDNsim can be used to evaluate a wide range of policies as well as to explore the benefits of caching in a CDN infrastructure.
- **Pricing of CDNs Services**: Pricing of CDNs' services is a challenging problem faced by managers in CDN providers. Deployment of new services, such as *Edgesuite*, are accompanied with open questions regarding pricing and service adoption. Chapter 8 addresses some pricing issues and presents some pricing models the context of CDNs. CDNsim can be used in order to validate them.
- **Mobile CDNs**: Content delivery on the mobile wireless Web is a topic of emerging interest and importance in the academic and industrial communities. Considering the recent advances in mobile content networking (e.g. WAP, IPv6 etc.), the infrastructure of mobile CDNs may play a leading role in terms of exploiting the emerging technological advances in the wireless Web. Chapter 14 presents mobile CDNs in details. CDNsim may be used as a testbed in order to address new research pathways in the area of mobile CDNs.
- **Peering of CDNs**: Peering of CDNs is gaining popularity among researchers of the scientific community. Several approaches are being conducted for finding ways for peering CDNs. However, several critical issues (i.e. When to peer? How to peer? etc.) should be addressed. Chapter 16 discusses some of these issues in detail. CDNsim may be used to simulate the peering CDNs framework under realistic traffic, workload, and replication conditions. It can also be utilized to evaluate the best practices and new techniques for load measurement, request redirection and content replication in the proposed framework for peering CDNs.
- **Security in CDNs**: The rapid growth of business transactions conducted on the Internet has drawn much attention to the problem of data security in CDNs [35].

In this context, secure content delivery protocols should be proposed in order to maintain content integrity (the delivered content which is modified by unauthorized entities should not be accepted) and confidentiality (the delivered contents cannot be viewed by unauthorized entities, including unauthorized proxies, and other users besides the requester) in CDNs. The high extensibility of CDNsim allows researchers to adapt the proposed protocols (e.g. iDeliver [35]) under its infrastructure.

- **P2P and Grid Technologies in CDNs:** Since CDNs are complex large-scale distributed systems, their development may be supported by the new emerging technologies of P2P and Grid. The successful exploitation and integration of these paradigms and technologies under a CDN infrastructure would provide an efficient way to cope with the aforementioned issues and would contribute significantly to the development of more efficient CDNs. The CDNsim architecture can easily enhance the aforementioned emerging technologies.

## 5.7 Conclusions

The Web has evolved rapidly from a simple information-sharing mechanism offering only static text and images to a rich assortment of dynamic and interactive services, such as video/audio conferencing, e-commerce, and distance learning. However, the explosive growth of the Web has imposed a heavy demand on networking resources and Web content providers. Users often experience long and unpredictable delays when retrieving Web pages from remote sites. CDN infrastructure seems to address the issues of capacity and performance on the Web in an efficient way. More and more Web content providers rely their content to be distributed by CDNs. The key to satisfy these growing demands lies in managing the content which is replicated in CDNs. Specifically, the need of various Web data caching techniques and mechanisms on CDNs has become obligatory towards improving information delivery over the Web.

In this chapter, we have summarized the emerging caching techniques which can be applied on CDN simulated frameworks. We study how to integrate caching policies on CDN's infrastructure. We also provide a comprehensive survey of the cache consistency mechanisms that can be applied on CDNs. Furthermore, we present the caching techniques which have been applied under CDNs for delivering dynamic content. Finally, we study these techniques under an analytic simulation tool for CDNs, the CDNsim.

To sum up, CDNs are still in an early stage of development and their future evolution remains an open issue. It is essential to understand the existing practices involved in a CDN framework in order to propose or predict the evolutionary steps. In this regard, caching-related practices seem to offer an effective roadmap for the further evolution of CDNs.

## References


1. Aioffi, W. M., Mateus, G. R., Almeida, J. M., Loureiro, A. A. F.: Dynamic content distribution for mobile enterprise networks. IEEE Journal on Selected Areas on Communication **23**(10) (2005)
2. Bakiras, S., Loukopoulos, T.: Increasing the performance of cdns using replication and caching: A hybrid approach. In: IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), p. 92.2. IEEE Computer Society, Washington, DC, USA (2005)
3. Bartolini, N., Presti, F. L., Petrioli, C.: Optimal dynamic replica placement in content delivery networks. In: 11th IEEE International Conference on Networks (ICON 2003), pp. 125–130. Sydney, Australia (2003)
4. Bent, L., Rabinovich, M., Voelker, G. M., Xiao, Z.: Characterization of a large web site population with implications for content delivery. World Wide Web **9**(4), 505–536 (2006)
5. Challenger, J., Dantzig, P., Iyengar, A., Witting, K.: A fragment-based approach for efficiently creating dynamic web content. ACM Transactions on Internet Technology. **5**(2), 359–389 (2005)
6. Chen, Y., Katz, R. H., Kubiatowicz, J.: Dynamic replica placement for scalable content delivery. In: IPTPS, pp. 306–318. Cambridge, USA (2002)
7. Chen, Y., Qiu, L., Chen, W., Nguyen, L., Katz, R. H.: Efficient and adaptive web replication using content clustering. IEEE Journal on Selected Areas in Communications **21**(6) (2003)
8. Duvvuri, V., Shenoy, P., Tewari, R.: Adaptive leases: A strong consistency mechanism for the world wide web. IEEE Transactions on Knowledge and Data Engineering **15**(5), 1266–1276 (2003)
9. Freedman, M. J., Freudenthal, E., Mazieres, D.: Democratizing content publication with coral. In: 1st USENIX/ACM Symposium, vol. 2004
10. Gray, C., Cheriton, D.: Leases: an efficient fault-tolerant mechanism for distributed file cache consistency. In: SOSP '89: Proceedings of the twelfth ACM symposium on Operating systems principles, pp. 202–210. ACM, New York, NY, USA (1989). http://doi.acm.org/10.1145/74850.74870
11. Huston, G.: Ipv4: How long do we have? The Internet Protocol Journal **6**(4) (2003)
12. Kangasharju, J., Roberts, J. W., Ross, K. W.: Object replication strategies in content distribution networks. Computer Communications **25**(4), 376–383 (2002)
13. Kulkarni, P., Shenoy, P. J., Gong, W.: Scalable techniques for memory-efficient cdn simulations. In: WWW, pp. 609–618 (2003)
14. Li, D., Cao, P., Dahlin, M.: Wcip:web cache invalidation protocol. IETF Internet Draft (2000)
15. Liu, C., Cao, P.: Maintaining strong cache consistency in the world-wide web. In: ICDCS '97: Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS '97), p. 12. IEEE Computer Society, Washington, DC, USA (1997)
16. Mikhailov, M., Wills, C. E.: Evaluating a new approach to strong web cache consistency with snapshots of collected content. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, pp. 599–608. ACM, New York, NY, USA (2003)
17. Ninan, A., Kulkarni, P., Shenoy, P., Ramamritham, K., Tewari, R.: Cooperative leases: scalable consistency maintenance in content distribution networks. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, pp. 1–12. ACM Press, New York, NY, USA (2002)
18. Pai, V. S., Wang, L., Park, K., Pang, R., Peterson, L.: Codeen. In: Second Workshop on Hot Topics in Net-working (HotNets-II) (2003)
19. Pallis, G., Stamos, K., Vakali, A., Katsaros, D., Sidiropoulos, A.: Replication based on objects load under a content distribution network. In: ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06). IEEE Computer Society, Atlanta, USA (2006)
20. Pallis, G., Thomos, C., Stamos, K., Vakali, A., Andreadis, G.: Content classification for caching under cdns. In: Innovation on Information Technology. IEEE Computer Society, Dubai, United Arab Emirates (2007)

21. Pallis, G., Vakali, A., Stamos, K., Sidiropoulos, A., Katsaros, D., Manolopoulos, Y.: A latency-based object placement approach in content distribution networks. In: Third Latin American Web Congress (LA-Web 2005), pp. 140–147. Buenos Aires, Argentina (2005)
22. Pierre, G., van Steen, M.: Globule: a collaborative content delivery network. IEEE Communications Magazine **44**(8), 127–133 (2006)
23. Podlipnig, S., Böszörmenyi, L.: A survey of web cache replacement strategies. ACM Computing Surveys **35**(4), 374–398 (2003)
24. Presti, F. L., Petrioli, C., Vicari, C.: Dynamic replica placement in content delivery networks. In: 13th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005), pp. 357–360. Atlanta, GA, USA (2005)
25. Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley (2002)
26. Rabinovich, M., Xiao, Z., Douglis, F., Kalmanek, C. R.: Moving edge-side includes to the real edge – the clients. In: USENIX Symposium on Internet Technologies and Systems. Seattle, Washington, USA (2003)
27. Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F.: Automatic fragment detection in dynamic web pages and its impact on caching. IEEE Transactions on Knowledge and Data Engineering **17**(6), 859–874 (2005)
28. Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: Analysis of caching and replication strategies for web applications. IEEE Internet Computing **11**(1), 60–66 (2007)
29. Stamos, K., Pallis, G., Thomos, C., Vakali, A.: A similarity based approach for integrated web caching and content replication in cdns. In: Tenth International Database Engineering and Applications Symposium (IDEAS 2006), pp. 239–242. Delhi, India (2006)
30. Stamos, K., Pallis, G., Vakali, A.: Integrating caching techniques on a content distribution network. In: Advances in Databases and Information Systems, 10th East European Conference, ADBIS 2006, pp. 200–215. Thessaloniki, Greece (2006)
31. Tewari, R., Niranjan, T., Ramamurthy, S.: Wcdp: Web content distribution protocol. IETF Internet Draft (2002)
32. Tse, S. S. H.: Approximate algorithms for document placement in distributed web servers. IEEE Transactions on Parallel and Distributed Systems **16**(6), 489–496 (2005)
33. Vakali, A., Pallis, G.: Content delivery networks: Status and trends. IEEE Internet Computing **7**(6), 68–74 (2003)
34. Wang, L., Pai, V. S., Peterson, L. L.: The effectiveness of request redirection on cdn robustness. In: 5th Symposium on Operating System Design and Implementation (OSDI 2002)
35. Yao, D., Koglin, Y., Bertino, E., Tamassia, R.: Decentralized authorization and data security in web content delivery. In: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing, pp. 1654–1661. ACM, New York, NY, USA (2007)
36. Yin, J., Alvisi, L., Dahlin, M., Iyengar, A.: Engineering web cache consistency. ACM Transactions on Internet Technology **2**(3), 224–259 (2002)
37. Zhuo, L., Wang, C. L., Lau, F. C. M.: Load balancing in distributed web server systems with partial document replication. In: 31st International Conference on Parallel Processing (ICPP), p. 305. Vancouver, Canada (2002)

# Chapter 6
# Request Redirection for Dynamic Content

Supranamaya Ranjan

## 6.1 Introduction

The increasing reliance on the WWW as a ubiquitous medium becomes ever more apparent whenever there is a disruption in the availability of a certain Web service. Furthermore, due to the much higher access network bandwidth today than a decade ago, clients of Web services have much higher expectations with the service quality and hence are less tolerant to degradation in throughput or access times. The disruptions and degradations in a Web service can be accounted for by two overload conditions. The first condition, *Time-of-Day* effect is the diurnal variation in traffic observed at most Web sites since most Web users access Web sites during day time than during night. This diurnal variation in traffic can sometimes cause 2–20 times more load at a Web site during peak usage times, than non-peak usage times [12, 13]. As a result, while planning the amount of resources to be provisioned for serving a Web site, operators face the challenge of either provisioning on the basis of peak usage or the mean usage patterns, both of which have their own sets of disadvantages. While provisioning for peak usage could imply a better response time, most of the resources would remain under-utilized during non-peak hours. Moreover, it is sometimes very difficult to predict the peak usage and hence equally difficult to provision for the same. Meanwhile, provisioning for mean usage implies better overall resource utilization, however during peak hours, the Web users may not experience the best response times. The second common reason for disruptions is *Flash Crowd* effect, where there is a sudden surge in users at a Web site at an unexpected time or of unexpectedly high magnitude. One of the most well known examples is the 1999 Victoria Secret's Webcast which became an instant hit and attracted 1.5 million users, much more than what the Web site was provisioned for, which brought the Web site down and a result, none of the users were able to view the Webcast at all.

In order to guarantee a pleasant browsing experience to users even during such overload conditions, Web content providers are increasingly off-loading the task of content placement and distribution to Content Delivery Networks (CDNs) such

Supranamaya Ranjan
Narus Inc., 500 Logue Ave, Mountain View, CA, USA 94043, e-mail: soups.ranjan@gmail.com

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

as Akamai [2], Limelight Networks [3] or Mirror Image [4]. The primary objective of CDNs is to reduce the end-to-end response time between the clients and the data they are accessing. In this regards, CDNs take two approaches, that of *content placement* which involves placing the content closer to the users and *redirection* which involves forwarding user requests to the best server. Content placement involves identifying the geographical "hot spots" i.e. areas that originate the most requests [18, 32] so that content can be replicated in these areas. Redirection involves selection of the the "best" server that can serve the request [20, 22, 26, 28, 35, 37]. This chapter deals with request redirection techniques to select the best server, where best could be defined by the closest server or the least-loaded and hence most available server or a combination of both. Typically, CDNs replicate a Web site's content on their own mirror servers. Alternatively, large Web content providers host their content on multiple clusters world-wide, where each cluster comprises of a set of servers that can process a user's request in-site. In such cases, CDNs only provide redirection techniques by which users of the service can be forwarded to the best server in the best cluster.

The definition of best server to serve a request varies depending on the type of content. Typically, static content which do not change over time such as images or pdf files are network-intensive and hence the best server is the one closest to the user. There are several definitions that could be ascribed to the closest server. The closest server could be one which is the least number of hops away. Network hops could be defined in terms of the number of routers present in the route between the client and server as found via traceroute. Alternatively, network hops could be defined by the number of Autonomous Systems (ASs) present in the shorted route between the client and server. The closest server could also be defined as one to reach which the network bandwidth of the bottleneck link (i.e. the network link in the route with the least bandwidth) is maximized. Finally, closest could also be defined as the server to reach which the expected network latency is minimized, where the network latency would then be a function of the number of hops as well as the network bandwidth of the bottleneck link.

However, with the latest trend towards personalizing users' browsing experience, content providers are using an increasing amount of *dynamic content* in their Web sites. Such content is generated dynamically depending on the user's location, past history or the nature of the request itself e.g. latest stock quote or latest auction bid. In such cases, a typical Web page would comprise both static and dynamic fragments. However, dynamic content places different demands on the resources involved as compared to static content. Since processing dynamic content involves either forking a new process (e.g. CGI scripts) or accessing database tables (PHP scripts), it is more compute intensive than static content. Moreover, because of overprovisioning in the Internet core [21], delays across network backbones are increasingly dominated by speed-of-light delays, with minimal router queuing delays. This implies that the server selection mechanisms as designed for static content may not be optimal for dynamic content. While for static content, forwarding a request to the closest server makes sense, for dynamic content, the optimal server selection mechanism must take both the network latencies and server loads into account. While

under certain conditions it may make sense to forward a dynamic content request to the closest server, under different conditions it may be better to forward it to the server furthest away, if it has the lowest sum total of network latency and expected server processing time.

Request redirection policies can be classified as either client-side or server-side redirection on the basis of the point in a request's execution where the redirection decision is made. A *client-side* redirection policy is one in which a client is redirected to a server by a client-side proxy. As used in Akamai, these client proxies which are also known as edge servers [19] form an overlay network and use traceroute and pings to estimate link latencies among themselves as well as the content provider sites. When a client requests a name-server mapping for a particular site, then it is returned three routes: *direct, best two-hop and second best two-hop* to servers hosting the content. The client then starts download on all three routes simultaneously and continues the download on the fastest one among them. In summary, Akamai first selects three servers on the basis of network-proximity and then accounts for the server-load as well by racing all the three routes. Besides this DNS based redirection technique, CDNs also use URL rewriting where URLs for content that can be generated away from the origin sites is rewritten so that it can be resolved via DNS to a CDN client-proxy that can serve the content. The actual syntax for this URL rewriting varies across CDNs. For instance, Akamai modifies the URL `www.foo.com/bar.gif` to `a1.akamai.net/www.foo.com/bar.gif`.

In contrast, a *server-side* redirection [34] is one in which a request could be first sent to an initial cluster via existing client-side mechanisms, after which the cluster redirector redirects the request away to the the "best server", which could be in either the local or the remote cluster. In this chapter, we mainly focus on server-side redirection algorithms for dynamic content such that requests may be redirected away from an overloaded Web cluster to a remote replica. However, the reader will find that much of the underlying principles as discussed for dynamic content could be suitably extended to other content types such as static content or streaming media as well. In this regard, the chapter introduces a per-request *W*ide-*A*rea *R*e*D*irection policy, henceforth referred to as WARD. The key objective of WARD is to minimize end-to-end client delays by determining at a Web cluster whether the *total* networking plus server processing delay is minimized by servicing the request remotely (via redirection) or locally. In particular, client requests could be first routed to an initial Web cluster via a client-side redirection mechanism such as simple DNS round-robin to more sophisticated server selection schemes [37]). Upon arrival at the initial cluster, a *request dispatcher* uses a measurement-based delay-minimization algorithm to determine whether to forward the request to a remote or local server. Thus, in contrast to other approaches (see Sect. 6.2), the redirector integrates networking and server processing delays and thereby minimizes the total delay.

In this chapter, we also develop a simple analytical model to characterize the effects of wide area request redirection on end-to-end delay. The analytical model allows us to perform a systematic performance evaluation of the benefits afforded by the per-request wide-area redirection policy. An example finding by this model is that for dynamic content applications, a server selection mechanism *must* obtain

fine-grained server load information owing to a much lower tolerance to errors in server loads compared to network latencies. This shows that for dynamic content applications, a server-side redirection policy can achieve a better performance than client-side redirection policies which can not obtain server load information at the same granularity and similar overheads as the server-side policies.

Finally, in this chapter we describe the design of a proof-of-concept testbed on which we compare WARD against other strawman policies such as those that redirect requests by only taking network latencies or server loads into account or those that redirect requests via round-robin policies. The testbed emulates a CDN that comprises of two geographically distant Web clusters connected by a wide-area link whose characteristics of round trip times and congestion are emulated by using Nistnet [6]. Each Web cluster is multi-tiered with a Web server tier, a request dispatcher tier, and a database tier. The application hosted on the Web clusters is an online bookstore that is modeled after the TPC-W benchmark [9]. As one of the main results, we use the testbed to show that for an e-commerce site with 300 concurrent clients, wide area redirection reduces the mean response time by 54%, from 5 s to 2.3 s.

The remainder of this chapter is organized as follows. In Sect. 6.2, we provide a background on current redirection techniques and in Sect. 6.3 we introduce the multi-tiered architecture as prevalent at current-day Web clusters. Further, in Sect. 6.4, we describe the system architecture of clusters for wide area request redirection. In Sect. 6.5, we develop a queuing model to study the architecture and in Sect. 6.6 we present numerical studies of the fraction of requests dispatched remotely and the expected response times under varying system scenarios. Next, we describe our testbed implementation and measurements in Sect. 6.7. In Sect. 6.8, we provide visionary thoughts for practitioners. Finally, we discuss future research directions in Sect. 6.9 and conclude in Sect. 6.10.

## 6.2 Related Work

Approaches to minimize Web access times can be separated into different groups: resource vs. request management and, for the latter, client-side vs server-side redirection.

One approach to minimizing Web access times is to ensure that enough resources are available at clusters. *Server migration* assigns servers that are unused or lightly loaded *within* an cluster to hosted applications that are suffering from high usage [33]. Server migration involves transfer of the application state from an existing server to a new server and hence migration times are on the order of 10 min. Therefore, server migration is a means to avoid bottlenecks over a long period of time (minutes or hours), e.g. following time-of-day patterns. Redirection is not only able to address long-term bottlenecks (at the additional redirection costs), but it is able to address short-term bottlenecks, e.g. due to flash-crowds, as well. *Server sharing*, as applied to content distribution by e.g. Villela et al. [36], is similar to server

migration, except that a fraction of the resources are assigned. Both server migration and sharing are orthogonal approaches to request redirection, and we advocate a combination of the mechanisms.

A significant body of research has focused on *client-side* mechanisms such as request redirection in CDNs [25, 37], server selection techniques [17, 20], caching [27], mirroring, and mirror placement [18, 23]. Such techniques are based on the premise that the network is the primary bottleneck. However, serving dynamic content shifts the bottleneck onto the cluster resources and typically the server's CPU. Thus, while such schemes can be applied to finding the best initial cluster, WARD's cluster-driven redirection is essential to jointly incorporate server and network latencies.

A combination of client-side and server-side redirection is also possible and beneficial if the bottleneck is not clearly identified or varying over time. Such a combined architecture is presented by Cardellini et al. [16]. Their server-side redirection mechanism may redirect entire Web requests using HTTP-redirection if the CPU utilization exceeds a certain threshold. They conclude that server-side redirection should be used selectively. In contrast, we see server-side redirection as a fundamental mechanism for current and future clusters. Our redirection mechanism is not threshold-based, but is able to optimize cluster response times for all CPU utilization values. Moreover, they design policies which consider network proximity and server load in isolation while our redirection policy integrates the two. Finally, rather than equating redirection to HTTP-redirect, we consider the dispatcher as a basic building block within the cluster architecture, which can redirect requests at any tier.

In contrast to approaches in references [17, 18, 20, 23, 25, 27, 37], that are designed for static content, other approaches such as those in references [15] or Akamai's EdgeSuite [1] address server selection for dynamic content via caching of dynamic fragments. Caching can occur at either the client-side with expiration times set using cookies or at the server-side (on a reverse proxy server) with cached pages being expired on receiving database update queries. However, these solutions either result in stale data being served to the clients or add to the complexity of site development and management. Nevertheless, caching is complementary to the solution adopted by WARD. In cases where a request is not resolved from the cache, the request can be forwarded to a server (local or remote) that can process it the earliest.

The approach as taken by contemporary CDNs such as Akamai for handling dynamic content is that they assume most of it is cacheable and hence the algorithm for server selection for dynamic content is a direct extension of that for static. In particular, Akamai uses an *EdgeSuite* mechanism for serving dynamic content, by interpreting each Webpage to be composed of various fragments, which could be of either static, dynamic-cacheable, or dynamic-uncacheable types. Thus, an *EdgeSuite* enabled redirector could select the closest server for serving the static and dynamic-cacheable fragments while sending the dynamic-uncacheable fragments to the origin-server. The redirector then assembles the responses for all the fragments and returns it to the user. However, it is estimated that only about 60% of dynamic fragments are cacheable [30]. Hence, in Websites with a significant portion of

uncacheable dynamic content such as e-commerce or auction sites, the user perceived download times could still be severely impaced by the time taken to service the uncacheable fragments at the origin site. Thus, content delivery policies for such sites would benefit from the server-side redirection approach as discussed in this chapter.

## 6.3 Background

In this section, we first introduce the multi-tiered architecture of Web clusters and then provide a brief background on queueing theory.

### *6.3.1 Cluster Architecture*

To illustrate the multi-tiered architecture at a Web-cluster as depicted in Fig. 6.1, let us consider the requests of an e-commerce session. First, a client request arrives at the initial cluster, where the selection of the initial cluster is via client-side redirection policies such as DNS round robin or more sophisticated policies such as [17, 20]. On arriving at the initial cluster, a dispatcher dispatches the request to a server on the Web-tier using either round robin or sophisticated policies [14]. If the request is for a static page, it is served by the Web server which also forms the response and returns it to the user. If the request is for dynamically generated content such as those involving purchase processing or shopping cart, then it is forwarded to a server in the application-tier. The application server then resolves all the dynamic fragments embedded within the client request by generating relevant database queries. The decision of which database server must handle a database query is made by another dispatcher. Finally, the application server collects all the responses to the database queries and returns the page to the Web server, which then forwards it back to the client.

### *6.3.2 Queueing Theory*

The analytical model that we will later develop in Sect. 6.5 represents a server as a M/G/1 queue [29]. A M/G/1 queue has exponentially distributed interarrival time for incoming requests and an arbitrary distribution for service times (‘M’, ‘G’ represent the Markovian and General distributions respectively while ‘1’ represents the presence of 1 queue). A M/G/1 queue is more general than a M/M/1 queue, which has exponentially distributed request interarrival times as well as service times. However, the increase in generality of a M/G/1 queue comes with a price. A M/G/1 queue does not have a general, closed form distribution for the number of jobs in

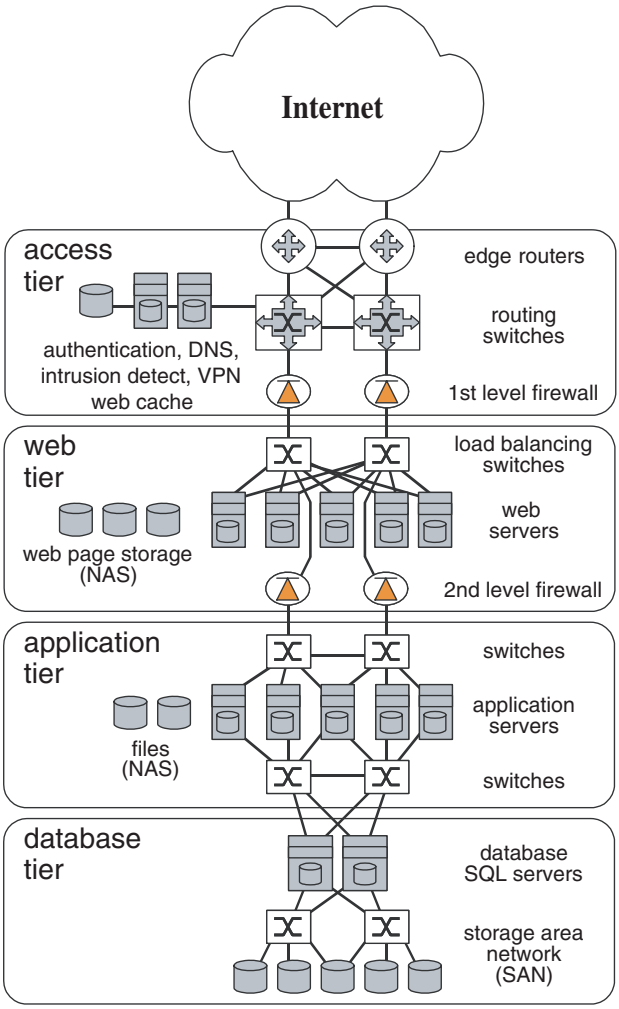



**Fig. 6.1** Cluster multi-tier architecture

the queue in steady state. It does, however, allow a general solution for the average number of jobs in the queue and the average time spent in the queue.

While we use a M/G/1 queue to represent a server, we would like to point that a more general and perhaps more accurate representation of a server would be a G/G/1 queue. Assuming an exponential request interarrival time implies that a request arriving at the Web site is independent of the past arrivals. However, requests arriving at a Web site do have some correlation amongst them, e.g. requests belonging to the same session are definitely correlated with each other. Moreover, requests at a back-end tier such as the database tier can be expected to be even more correlated since one Web request can spawn multiple correlated database queries. However, for the

purpose of simplification we assume a M/G/1 queue which provides a closed form expression for the average waiting time in a queue and hence allows us to derive the fraction of requests that can be redirected.

## 6.4 Redirection Architecture and Algorithm

In this section, we describe WARD, a *W*ide *A*rea *R*edirection architecture for *D*ynamic content, and finally present a measurement-based redirection algorithm.

### *6.4.1 WARD*

In a CDN, services and applications are replicated across all clusters, connected using custom high-bandwidth links. In practice, a cluster operator has multiple clusters at a number of geographically dispersed sites. A client request arrives at the initial cluster, where selection of the initial cluster can be static or dynamic via DNS round robin or via more sophisticated policies that take into account the content availability and network latencies as measured periodically by proxies [17, 20]. A dispatcher as illustrated in Fig. 6.2 can potentially redirect the request to a remote cluster according to the algorithm presented below. The objective of the redirection algorithm is to redirect requests only if the savings in the request's processing time at the remote cluster overwhelm the network latency incurred to traverse the backbone in both the forward and reverse path. In this way, end-to-end client delays can be reduced while requiring changes only to the dispatcher, and leaving other elements unchanged.

WARD therefore provides a foundation for spatial multiplexing of cluster resources. Namely, as a particular cluster becomes a hot-spot due to flash crowds [24, 31] or time-of-day effects [33], load can be transparently redirected to other clusters while ensuring a latency benefit to clients. For example, client access patterns have been observed to follow *time-of-day* patterns where server utilization varies with a diurnal frequency. We can exploit this effect such that no cluster has to provision for the peak demand. Thus, when the workload to one cluster is peaking, the workload at an cluster several time zones away will be much lower, enabling a significant performance improvement by allowing redirection among clusters.

### *6.4.2 Redirection Algorithm*

The objective of the redirection algorithm is to minimize the total time to service a request. Namely, if a request arrives at cluster $k$, then the objective is to dispatch the request to cluster $j$ satisfying

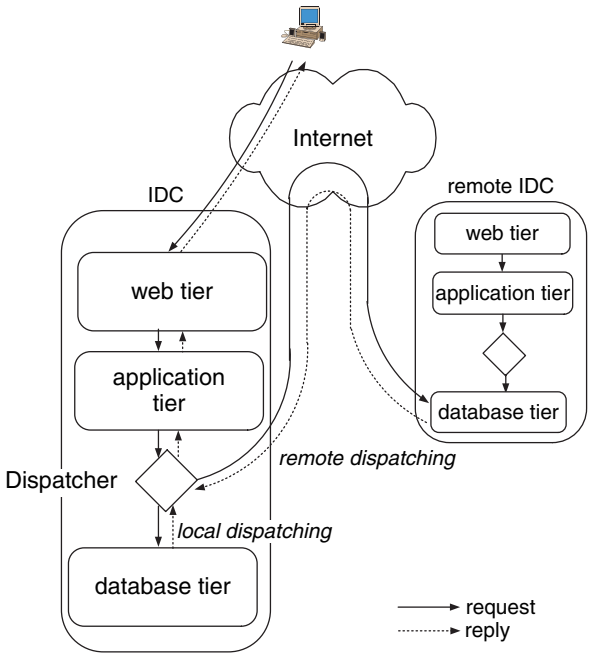



**Fig. 6.2** Wide-area redirection via redirector in front of the database tier

$$\text{argmin}_j\,(2\,\Delta_{kj} + T_j) \tag{6.1}$$

where $\Delta_{kj}$ denotes the network delay between cluster $k$ and $j$ and $T_j$ is the request's service time at cluster $j$.

In practice, the actual service time at each remote cluster $T_j$ cannot be known in advance. Yet, it can be estimated from the average load at cluster $j$ as well as the request type. Thus, we employ a measurement-based algorithm in which the average $T_j$ is estimated from $\rho_j$, the mean load at cluster $j$, as well as the request type. In WARD, this is achieved by measuring a mean delay vs. load profile for each request type. Clusters periodically exchange load information to refine their estimates of each others' processing delays. In contrast, $\Delta_{kj}$ remains relatively static among clusters due to their high-speed interconnection links. Thus, on request arrival, the dispatcher uses the measured load at cluster $j$ on the delay vs. load profile corresponding to this request's type to estimate the total service time on cluster $j$.

We consider a second policy which does not make a decision on a per request basis but rather computes a fraction of requests to be remotely dispatched. In particular, we show in the next section, that under certain simplifications there is an optimal ratio of requests that should be remotely dispatched in order to minimize the delay of all requests. Once this ratio is known, the dispatcher can simply redirect a request remotely with the computed probability. We refer to these two redirection policies as *per-request* redirection (or, equivalently *per-query* redirection) and *probabilistic* redirection.

## 6.5 Performance Model

In this section, we develop a performance model for wide area redirection. For a given workload, mean and variance of service time, and network latency, we derive an expression for the delay-minimizing fraction of requests that a dispatcher should redirect to remote clusters. Moreover, we compute the average total response time including service and waiting-times and end-to-end network latency. We then perform a systematic performance analysis to estimate the optimal dispatching ratios $\alpha^*$ and to predict the expected average request response time under varying parameters, such as the server load, the end-to-end network latency and the average request service time.

Figure 6.3 illustrates the system model for WARD. We model request arrivals at cluster $i$ as a Poisson process with rate $\lambda_i$ and consider a single bottleneck tier modeled by a general service time distribution having mean $\bar{x}_i$ and variance $\sigma_i^2$.

We consider a redirection algorithm in which a request is redirected from cluster $j$ to cluster $i$ with probability $\alpha_{ji}$, i.e. we consider probabilistic redirection. We also denote $E[T_i]$ as the expected total delay for servicing a request at cluster $i$, and denote $\Delta_{ji}$ as the one-way end-to-end network latency for a request sent from cluster $j$ to cluster $i$.

For the general case of a system of $n$ cluster replicas, let us denote $\mathbf{A} = \{\alpha_{11}, \ldots, \alpha_{ji}, \ldots, \alpha_{nn}\}$ as a matrix of request dispatching fractions, $\mathbf{E}[\mathbf{T}] = \{T_1, \ldots, T_n\}$ as the vector of all total delays at an cluster bottleneck tier and $\mathbf{D} = \{2\Delta_{11}, \ldots, \Delta_{ji} + \Delta_{ij}, \ldots, 2\Delta_{nn}\}$ as a matrix of round-trip times from cluster $i$ to cluster $j$ and back. Furthermore, we denote $\mathbf{L} = \{\lambda_1, \ldots, \lambda_n\}$ as a vector of request arrival rates at the cluster dispatchers, $\overline{\mathbf{X}} = \{\bar{x}_1, \ldots, \bar{x}_n\}$ as the average service time, $\mathbf{C} = \{c_1, \ldots, c_n\}$ as the vector of squared coefficient of variation for the service times, with $c_i = \sigma_i^2/\bar{x}_i^2$.

**Lemma 1.** *The mean service time for the redirection policy using a dispatching fraction* **A** *is given by:*

$$\mathbf{E}[\mathbf{T}] = \mathbf{A} \cdot \overline{\mathbf{X}} + \frac{(\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}}^2(1 + \mathbf{C}^2)}{2(1 - (\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}})} + \mathbf{A} \cdot \mathbf{D} \tag{6.2}$$

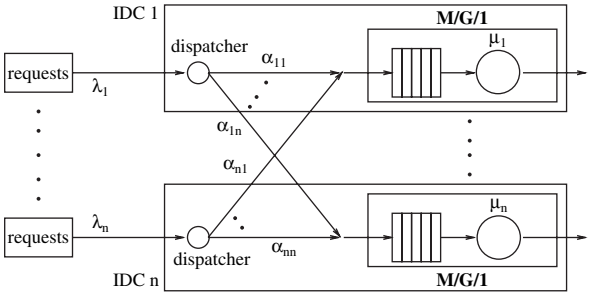



**Fig. 6.3** Redirection system model

We provide a proof for the Lemma 1 at the end of the chapter in the Appendix. The proof is based on the well known closed form expression for the average waiting time at a M/G/1 queue. Next, from (6.2), we can compute the optimal dispatching ratios that minimize the service times over all requests. In particular, let us $\mathbf{A} = \{\alpha^*_{11}, \ldots, \alpha^*_{nn}\}$ denote the matrix of optimal request dispatching ratios.

**Proposition 1.** *The optimal dispatching ratios* $\mathbf{A}^*$ *are given by:*

$$\frac{\partial}{\partial \alpha}(\mathbf{A} \cdot \overline{\mathbf{X}} + \frac{(\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}}^2(1+\mathbf{C}^2)}{2(1-(\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}})} + \mathbf{A} \cdot \mathbf{D}) = 0 \tag{6.3}$$

with $\mathbf{E}[\mathbf{T}]$ defined in (6.2).

To solve (6.3) for all $\alpha_{ji}$, we use the following constraints to reduce the number of unknowns. First, we clearly have $\sum_j \alpha^*_{ji} = 1$. Second, $\lambda_i \geq \lambda_j \implies \alpha^*_{ji} = 0$ i.e. when considering 2 clusters with different $\lambda$, under steady-state conditions, no requests will be dispatched from the cluster with a smaller arrival rate to the cluster with a higher arrival rate.

The optimal dispatching ratios $\mathbf{A}^*$ can be used to predict the average request service time for a system of cluster replicas.

**Proposition 2.** *The expected request service time under optimal dispatching ratios is given by:*

$$\mathbf{E}[\mathbf{T}^*] = \mathbf{A}^* \cdot \overline{\mathbf{X}} + \frac{(\mathbf{A}^* \cdot \mathbf{L})\overline{\mathbf{X}}^2(1+\mathbf{C}^2)}{2(1-(\mathbf{A}^* \cdot \mathbf{L})\overline{\mathbf{X}})} + \mathbf{A}^* \cdot \mathbf{D} \tag{6.4}$$

*Proof:* (6.4) follows from Lemma 1 and by using the optimal dispatching ratios from (6.3).

## 6.6 Numerical Results

Next, we show that wide area redirection is able to optimize inter-cluster performance characterized by total access delays perceived by clients. Then, we experimentally establish the higher tolerance of WARD to measurement errors in network latencies than to errors in server load measurements. This further verifies our hypothesis that redirection mechanisms must obtain finer-granularity server load measurements, which WARD is better-suited at given that it is implemented on dispatchers that are co-located with local servers and are connected via high-bandwidth links to the remote servers.

Using the system model developed in Sect. 6.5, we consider a system of 2 clusters with replicas having the same average request service time $\overline{x}$. Furthermore, we assume a symmetric network with wide area latency between the two clusters: $\Delta = \Delta_{12} = \Delta_{21}$. Finally, we set $\lambda_2 = 0$, which satisfies $\lambda_1 > \lambda_2 \implies \alpha_{21} = 0$, and denote $\lambda := \lambda_1$ and $\alpha^* := \alpha^*_{11}$ for simplicity.

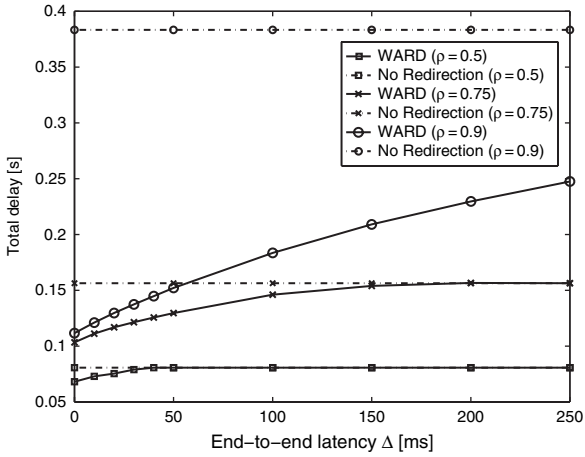



**Fig. 6.4** Comparison of total delay with and without wide area redirection

The dispatching ratio is computed based on (6.2):

$$E[T] = \alpha\bar{x} + \frac{\alpha\lambda\bar{x}^2(1+c^2)}{2(1-\alpha\lambda\bar{x})} + (1-\alpha)\bar{x} + \frac{(1-\alpha)\lambda\bar{x}^2(1+c^2)}{2(1-(1-\alpha)\lambda\bar{x})} + 2(1-\alpha)\Delta \quad (6.5)$$

Equation (6.5) is solved according to Proposition 1 to obtain the optimal dispatching ratio $\alpha^*$. Henceforth, we refer to the term $1-\alpha^*$ as the *remote redirection* ratio, i.e. the fraction of requests dispatched remotely. Then, according to Proposition 2, the expected total delay of the cluster system is given by substituting $\alpha$ by the optimal ratio $\alpha^*$ in (6.5).

If not otherwise stated, we use the following default values: $\bar{x} = 42.9$ ms, $\sigma = 40.1$ ms, where these values were obtained from our testbed and $\Delta = 36$ ms, which corresponds to a speed-of-light latency for two clusters separated by 6 time-zones at 45° latitude.[1] We will use $\rho = \lambda\bar{x}$ to denote the total load on *all* clusters. For clusters without redirection, $\rho$ corresponds to the server load on the bottleneck tier, whereas WARD can split this load among the local and remote clusters. To obtain a given value of $\rho$, the arrival rate $\lambda$ will be scaled, with $\bar{x}$ remaining fixed.

### *6.6.1 The Case for Wide-Area Redirection*

First, we provide evidence that wide area redirection is able to decrease the user-perceived total delay. We calculate the total delay of WARD using (6.3) and (6.4) and compare it to the total delay of a cluster that does not implement redirection.

[1] Given the circumference of earth at 45° latitude as 28335 km and the speed-of-light through optical-fiber as 205 km/s, the one-way latency across 1 time-zone can be calculated as: $28335/(205*24) \approx 6$ ms.

Figure 6.4 shows the total delay as a function of the end-to-end latency and different system loads $\rho$.

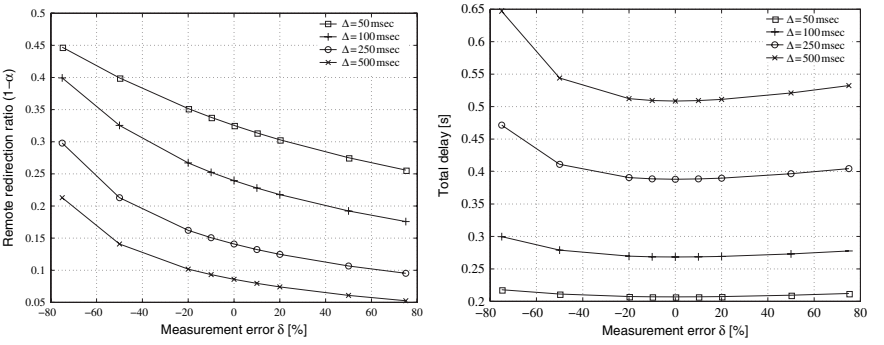
For low loads ($\rho = 0.5$), improvements are achieved only when the end-to-end latency $\Delta \leq 25$ ms. For higher latencies, the redirection cost exceeds the processing time so that all requests are serviced locally. However, a significant improvement is achievable for higher loads. For a moderate load of $\rho = 0.75$ and $\Delta < 50$ ms, the total delay is reduced from 0.16 s to 0.13 s using WARD, an improvement of 18%. For a heavily loaded system with $\rho = 0.9$ and when $\Delta < 50$ ms, the total delay is reduced from 0.38 s without redirection to 0.15 s using WARD, an improvement of 60%. Moreover, for loads $\rho > 0.9$, still higher improvements are predicted by the model.

### *6.6.2 Susceptibility to Measurement Errors*

Next, we establish the fact that the performance benefits out of a wide-area redirection policy can be exploited only when the server utilization values are available at a very fine granularity. In contrast, network latencies can be quite coarse grained without any performance penalties out of making the wrong decision. For establishing this claim, we study the performance impact due to measurement errors in network latency $\Delta$ and server load $\rho$. We quantify the performance impact in terms of *error tolerance* defined as the percentage error $\pm\varepsilon$ that increases the total delay by at most 2%.

First, we study the impact of network latency errors as follows. Let $\Delta$ denote the true inter-cluster network latency and $\widehat{\Delta} = \Delta + \delta$ the measured value, and $\widehat{\mathbf{D}}$ the corresponding round-trip time matrix. The dispatcher calculates the dispatching ratios replacing $\mathbf{D}$ by $\widehat{D}$ in (6.2).

For the calculation of average total delay, (6.4) is used with the true latency values $\mathbf{D}$. The effects of measurement error in network latency on the remote redirection ratio $(1-\alpha)$ and the resulting average request response time are shown in Fig. 6.5.




(a) Remote redirection ratio (b) Total delay

**Fig. 6.5** WARD performance under network measurement errors. A value of 0 on the x-axis corresponds to perfect end-to-end latency information. The server load is set to $\rho = 0.95$

Each curve denotes a different (true) latency $\Delta$, and the x-axis denotes the error $\delta$, in percent of $\Delta$.

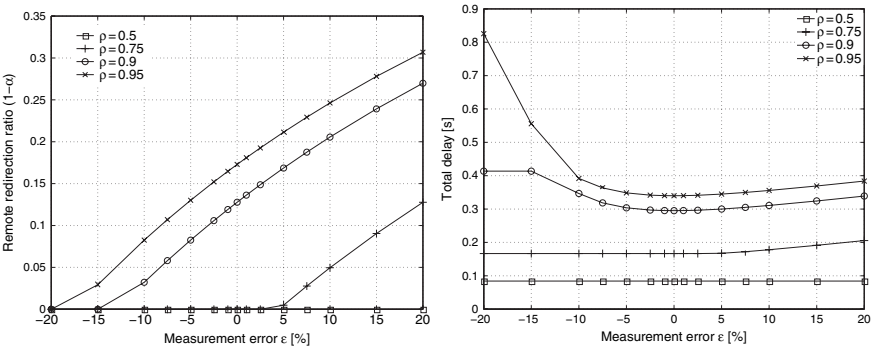
Figure 6.5(a) shows that the redirection ratio changes more for negative $\delta$ than for the corresponding positive $\delta$. The reason is that the redirection ratio does not grow linearly with the end-to-end latency. As a consequence of the asymmetry, the total delay increases more for negative $\delta$, as shown in Fig. 6.5(b). Note, however, that the response times are not highly sensitive to latency measurement errors and the error tolerance is quite high at $\pm 20\%$.

Likewise, we consider a scenario when the dispatcher has inaccurate server load measurements, e.g. due to delays in receiving the measurements. In this scenario, the measured load at the dispatcher is given by $\widehat{\rho} = \widehat{\lambda}\overline{x}$, with $\widehat{\lambda} = \lambda + \varepsilon$ (where $\varepsilon$ is in percent of the correct load $\rho$) and the corresponding measured arrival rate by $\widehat{\mathbf{L}}$. The network latency is set to $\Delta = 500$ ms.

First, consider the case of measurement error $\varepsilon > 0$, when the dispatcher assumes the server load to be higher than what it is and hence it redirects more requests than the optimal. Figure 6.6(a) shows that the remote redirection ratio increases with increasing measurement errors. These extra redirections incur additional network latencies and hence the total delay also increases linearly in Fig. 6.6(b). In particular, for $\rho \geq 0.9$, the error tolerance is $+1.5\%$. Next, consider negative $\varepsilon$, when the dispatcher assumes the local server load to be less than the actual value and hence redirects pessimistically. As a result, the load on the local server incurs greater processing times at the local cluster. As expected, Fig. 6.6(b) shows that at high server loads $\rho \geq 0.9$, the total delay is much more sensitive for negative $\varepsilon$ with an error tolerance of only $-0.5\%$.

Thus, comparing the impact of latency and server measurement errors, the error tolerance for latency is high at $\pm 20\%$ while that for server load is an order of magnitude lower at $+1.5, -0.5\%$. We thus conclude that greater accuracy is needed in server load measurements than network latency.

Since server-side redirection mechanisms can obtain more fine-grained server load information at lower overheads, this verifies their superiority in efficiently load-




(a) Remote redirection ratio (b) Total delay

**Fig. 6.6** WARD performance under server load measurement errors

balancing requests for dynamic content applications than client-side mechanisms: First, client-side redirection policies when implemented at clients or DNS servers may not have access to high-bandwidth links to the servers and; Second, client-side redirection as implemented at proxies near clients (e.g. Akamai) may have high-bandwidth access links, however, their overhead for obtaining the server load information is much higher, given the much larger number of client-side dispatchers than server-side ones. Consider the following: let the number of clusters is $n$ and thus in WARD, there are $n$ server-side dispatchers. If the total number of servers across all clusters in the tier implementing WARD is $M$, where $M > n$, then the complexity of information exchanged in WARD is $O(nM)$. In contrast, consider a client-side redirection mechanism with one dispatcher per client-side proxy for a total of N client-side proxies or dispatchers which yields a complexity of $O(NM)$. Given that the number of client-side proxies is typically much larger than the number of cluster replicas, the complexity is much less for server-side redirection mechanisms. We highlight this via an example: assuming that a client-side redirection scheme were to install one proxy per Autonomous System in the world, then N can be expected to be around 39,000. In contrast, typical cluster grids such as Google consist of about 60 clusters where $60 \ll 39,000$.

## 6.7 Testbed Implementation and Experiments

In this section, we describe a CDN prototype implementation that we use to experimentally compare wide area redirection via WARD against other redirection policies. Our results provide a proof-of-concept demonstration of wide area cluster-driven redirection, experiments into the testbed's key performance factors, and validate the performance model.

The testbed, depicted in Fig. 6.1, consists of a cluster of Intel Pentium IV 2.0 GHz processor machines running Linux 2.4.18-14, with 512 MB SDRAM, and a 30 GB ATA-66 disk drive. One machine is configured as a router and runs Nistnet [6], an IP-layer network emulation package. The router separates the remaining machines into 3 domains for the client and 2 clusters. This setup allows variation of the network conditions (delay and bandwidth) between the client and the clusters as well as between clusters. We developed a 3-tier system architecture as depicted in Fig. 6.2. At the Web tier, we use an Apache Web server [8] and dynamic content is coded using PHP scripts [7] at the application tier. Access to the 4 GB database is provided by a MySQL server [5].

### *6.7.1 Database Dispatcher*

A key aspect of the cluster architecture is the dispatcher which makes the decision whether to service a request at the local or a remote cluster. On our testbed, the

*database* tier becomes the bottleneck first, due to the substantial processing demands of complex database queries. We therefore implemented the dispatcher with remote redirection capabilities in front of the database. In our implementation, we provide the Web and application tiers with sufficient resources such that the database tier is indeed the bottleneck.

While the objective of the database dispatcher is to minimize the response time of the queries, its dispatching capabilities are restricted by consistency constraints. The dispatcher addresses the consistency issues by two means: maintaining an identical *total ordering* of writes at all database servers, and a *read-one write-all* dispatching strategy. To maintain an identical *total ordering* of writes at all database servers, each write query is assigned a unique sequence number, and the database servers process the writes in the order of the sequence numbers. In the read-one write-all dispatching strategy, write queries are sent to all database servers, and the response is returned as soon as one of the database servers has processed the write. Hence, database consistency is maintained *asynchronously*. A read query is sent to one of the database servers where the corresponding conflicting writes have been processed using a scheduling strategy described as *conflict-aware* [11]. This *asynchronously*-maintained consistency along with *conflict-aware* scheduling has been shown to scale to higher throughputs compared to the synchronous consistency algorithms.

However, though conflict-aware scheduling limits the server set available for wide area dispatching for read queries, two factors outweigh this limitation. Firstly, read queries have a larger service time than write queries and secondly, e-commerce workloads have a significantly higher percentage of read queries [10].

WARD allows a general framework to implement remote redirection at any tier. Though we implemented it in front of the database tier, we could equivalently do it in front of the Web/application tier, in which case we would redirect an entire request instead of database queries. Request redirection would incur less of network overhead, but it would constrain the queries to the local database servers, whereby the advantages due to remote redirection of queries would not be realized. We do not explore request redirection versus query redirection in this chapter and instead focus on the performance gains of remote redirection in general.

### *6.7.2 Redirection Algorithms*

Under the consistency constraints, the database dispatcher directs read queries to the database server with the least expected response time as follows. First, from the list of clusters, all those which have unprocessed conflicting writes are excluded using conflict-aware scheduling. From the remaining clusters, the dispatcher selects an cluster by using either the *per-query* or, the *probabilistic* redirection policy. In the per-query redirection policy, the dispatcher calculates the expected response time by using measured loads of database tiers to determine if the latency overhead incurred by remote dispatching is outweighed by the savings in server processing time. In the probabilistic policy, the dispatcher uses the optimal redirection ratio

computed by the model and dispatches queries with that probability. We implement the probabilistic policy such that given a number of clients, it is configured for the redirection ratio predicted by the model and hence it doesn't use the online server load measurements.

### 6.7.3 TPC-W Workload

For our experimental workload, we utilize the TPC-W benchmark [9] to represent an e-commerce workload characterizing an online bookstore site. We use the implementation developed by Amza et al. for conflict-aware scheduling for dynamic content applications [11].

The workload for TPC-W is generated by a client emulator which generates the requests specified in the browsing mix of TPC-W, which consists of 95% read queries and 5% writes. The client emulator opens a set of $n$ user sessions which last for 15 min. Each session opens a persistent HTTP connection to the Web server and sends a sequence of requests to the cluster. Between two requests, the user waits for a configurable parameter termed *think time* before the next request is generated. The mean think time, which we set to 7 s, together with the number of users, defines the request arrival rate at the cluster. Note here that a PHP script can contain multiple database queries. The extraction and serialization of the queries is done by the application tier. Due to the fact that each request consists of several embedded queries, their arrival distribution and rate at the database are different from those generated by the client emulator.

### 6.7.4 Experiments

The input parameters in our experimental study are the inter-cluster link latency which we vary through the Nistnet module and the number of clients which arrive at each cluster. The parameters we measure are *request response time* as perceived by the clients, *query response time* as perceived by the database dispatcher and *remote redirection ratio* as achieved by the database dispatcher. Request response time is defined as the time elapsed between the generation of a request and the return of the last byte of the response to the client. Query response time is defined as the time period between the sending of a query by the dispatcher to the database and the reception of the response by the dispatcher. We measure the mean and 90th percentile request (query) response time for all requests (queries) generated during the entire duration of an experiment. Remote redirection ratio is defined as the fraction of the number of queries sent by the dispatcher to a remote database server.

We first present the offline technique to configure our *per-query* redirection policy with the *query response time characteristics*. Second, we quantify the performance benefits of the WARD architecture by exploring the trade-off between the

load on the local database server and wide area link latency. Third, we compare performance gains predicted by the analytical model of Sect. 6.5 with those obtained via testbed measurements.

#### 6.7.4.1 Offline Measurement of Query Response Time Characteristics

In these experiments, we measure the response time as a function of CPU load, a key input to the per-query redirection policy. We use one cluster with access to one local database server. The execution time for a query depends on the number and type of other queries executing at the same time on the database server, which can be abstracted as the workload entering the system. Hence, we vary the CPU load on the database server by increasing the number of clients. In each case, we measure the mean execution time for each of the 30 read-only MySQL queries. The resulting delay-load curve as illustrated in Fig. 6.7 is then used in the *per-query* redirection policy.

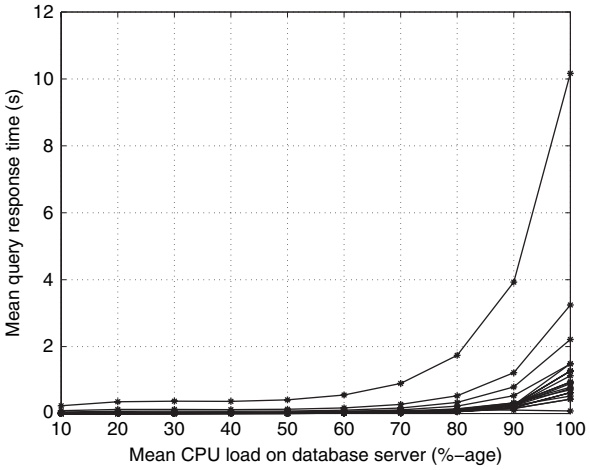



**Fig. 6.7** Mean database query response times vs mean database CPU load for the 30 read-only MySQL queries in the browsing mix of the TPC-W benchmark

#### 6.7.4.2 WARD Performance Gains

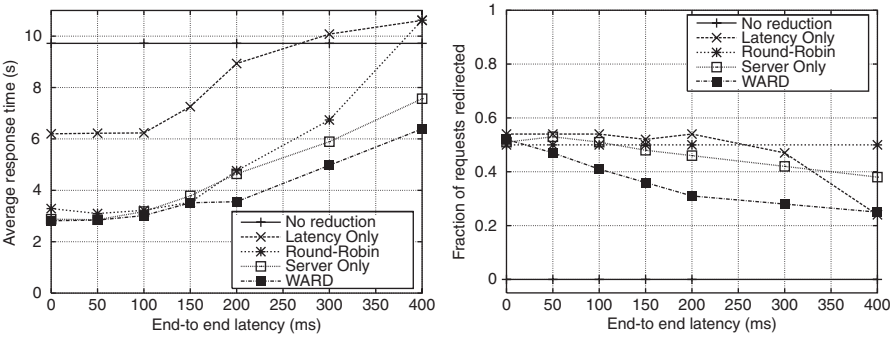
Using our experimental test bed, we first validate the efficacy of WARD in improving the cluster performance owing to its ability to perform a per-query redirection while taking both server loads and network latencies into account. In this experiment, we use a set up in which the local cluster's Web tier is over-provisioned with servers such that it never becomes the bottleneck. Both the local and remote clusters have 1 server each at their database tier. Requests only arrive at the local cluster and

after being processed at the Web tier, their queries may either be processed at the local database tier or redirected to the remote database tier.

We compare the performance of WARD against the following 4 different strawman algorithms: (1) *No Redirection* where all queries are processed locally; (2) *Latency Only* where queries are forwarded to the server with the least round-trip time as measured in the last measurement interval; (3) *Server Only* where queries are forwarded to the server with the least CPU load and hence which can be expected to process the query the fastest; (4) *Round Robin* where queries are forwarded in a round-robin fashion between the local and remote database servers. We refer to the last three algorithms that involve redirection of queries collectively as the redirection strawman algorithms.

Figure 6.8 shows the performance achieved using a trace generated using the browsing mix of TPC-W consisting of 100 client sessions with mean inter-request arrival time per session being 4 s. First, Fig. 6.8(a) shows that the performance of all algorithms that redirect queries away from the local server is better than the No Redirection algorithm. This highlights the fact that for this workload the local database server is heavily loaded and hence it is of benefit to redirect a part of its load to the remote server. However, the performance of WARD is much better compared to all the strawman algorithms, thereby proving its superiority. Infact, as seen from Fig. 6.8(b), WARD achieves a better performance while redirecting the least fraction of queries to the remote server. This is on account of the better redirection decision made by WARD on a per-query basis by accounting for both the server loads and latencies.

Second, the Latency Only algorithm achieves the worst performance amongst the redirection strawman algorithms. This behavior further validates our hypothesis that for a dynamic content Web site such as ours, the server loads are more important for forwarding decisions than the latencies. However, even the Server Only algorithm's performance becomes worse compared to WARD with increasing inter-cluster latencies. This is again an expected behavior since the Server Only algorithm doesn't




(a) Response time (a) Redirection ratio

**Fig. 6.8** Performance of WARD against strawman algorithms of No Redirection, Latency Only, Server Only and Round Robin

take the latencies into account in its redirection decision and hence is unable to make the correct redirection decision per-query. In contrast, WARD is designed to take both the server loads and latencies into account in its redirection decision.

Third, the performance of all the redirection algorithms (WARD as well as the strawman redirection algorithms) degrades with increasing inter-cluster latencies since the latency overhead of redirection is higher per-query. Eventually, the cost of redirecting even a single query can be expected to outweigh the benefit of having a remote server with lower CPU load. Hence, the performance of the Latency Only (Round Robin) algorithm becomes worse than not redirecting at all once the end-to-end latencies are as high as 270 (375) ms. In contrast, WARD can support having the remote cluster the furthest away than all the strawman algorithms.

#### 6.7.4.3 Model Validation and Redirection Policies

We validate the analytical model developed in Sect. 6.5 by comparing against the WARD redirection policies on our testbed. Since the bottleneck tier is the database tier, we compare the redirection ratios and response time for processing queries at this tier under the model as well as on our testbed. For the model, we use (6.5) from Sect. 6.6 with $\overline{x} = 42.9$ ms and $\sigma = 40.1$ ms, as measured on an unloaded database server in our testbed.

First, Fig. 6.9 compares the mean query response time of the model and the implementation on a single cluster as a function of the server load $\rho$. Observe that the model matches the measured query response time for $\rho < 0.7$ within $\pm 10\%$. Beyond this load, the model deviates from the implementation because: (1) our M/G/1 model makes the simplifying assumption that the arrival process of queries at the database tier is independent which may not hold true given the correlation across queries

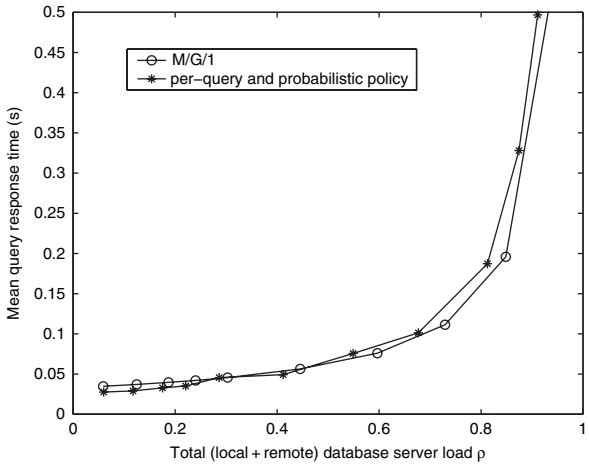



**Fig. 6.9** WARD analytical model for 1-cluster.

generated for the same Web request; (2) our M/G/1 model doesn't take read-write conflicts into account due to which queries may take longer to process than what the model predicts and; (3) at high loads there are more queries and thereby greater number of conflicts.

Next, we compare the model with the two implemented redirection policies: (1) *probabilistic*, and (2) *per-query*. The per-query policy receives the CPU load measurements every 5 s and we set the inter-cluster latency to be 25 ms in all the experiments.

Figure 6.10(a) and (b) compare the remote redirection ratio and query response time as a function of the system load. The redirection ratios of the model and the probabilistic policy are close because this policy bases itself upon the optimal values predicted by the model. On the other hand, the per-query policy begins redirecting earlier and redirects more queries until $\rho < 0.5$ compared to both the model and probabilistic policy. The reason for this behavior is that heavy queries are more sensitive to load as shown in Fig. 6.8(a), and hence it is of increasing value to redirect them at comparatively lower system loads. Hence, the per-query policy performs better and exhibits a lower mean response time for $\rho < 0.5$ in Fig. 6.10(b). When $\rho > 0.5$, the probabilistic policy redirects more queries than the per-query policy and hence yields lower response times. We attribute this difference to the fact that the the measurement interval of 5 s is too coarse grained to capture the small oscillations in CPU load. A better response time can be expected for smaller measurement intervals, but would require that an optimal tradeoff be established between measurement accuracy and measurement overhead.

Thus, we derive two important conclusions from this experiment. First, that despite its simplifying assumptions, the M/G/1 model does match the implementation closely and hence the conclusions derived by using the model in Sect. 6.6 should be expected to hold true in real world implementations as well. Second, the WARD per-query redirection algorithm performs better than the probabilistic algorithm under low load scenarios and its performance under high load scenarios can be improved by reducing the measurement interval.

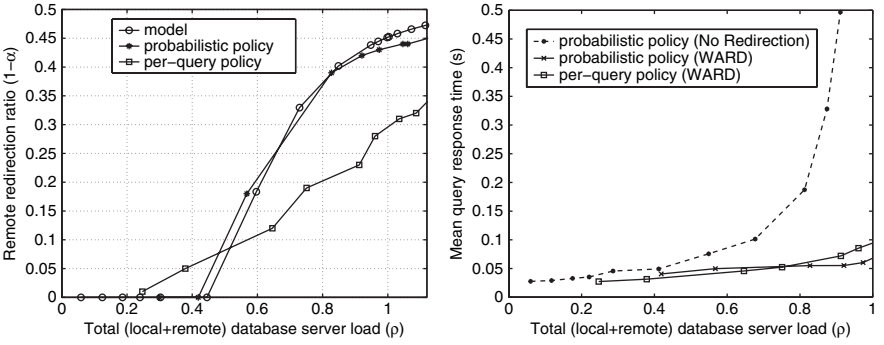



**Fig. 6.10** WARD analytical model validation

## 6.8 Visionary Thoughts for Practitioners

Our goal through this chapter is to provide the following visionary thoughts to practitioners. First, we highlight the importance of request redirection in multiplexing the CDN resources besides providing a reasonable response time to Web users. In this regards, consider a Web application that is accessed by users around the world. Web applications have been documented to experience a time-of-day traffic pattern where the traffic is higher during office hours than night time. Given the goal of client-side redirection mechanisms which typically redirect a user to the "closest" cluster server, we can thus expect the traffic at a particular cluster to be higher during its daylight hours. Now if a CDN was used to host an application across two clusters in Europe and North America, then owing to their time-difference significant resource multiplexing savings can be achieved. A wide-area redirection mechanism can then redirect requests away from the overloaded American cluster to the temporarily underloaded European cluster if during the American day time, the traffic exceeds normal levels. This can be exploited to achieve resource savings since neither the American nor the European cluster needs to overprovision their cluster resources and can instead achieve a similar performance by multiplexing their resources via a wide-area redirection mechanism. The second message for practitioners is the importance of providing a per-request redirection or server selection mechanism which incorporates both server loads and network latencies. In this regards, in Sect. 6.6.2 we highlight the ability of server-side redirection mechanisms to collect fine-grained server load measurements at lower overheads. This highlights an important finding that a server-side redirection mechanism can achieve better performance than client-side redirection mechanisms (such as DNS redirection in Akamai) at lower measurement overheads.

## 6.9 Future Research Directions

In this chapter, we discussed the performance savings that can be achieved by a request redirection mechanism and designed a request redirection algorithm for dynamic content. The example implementation for the redirection algorithm is on the database tier which is identified as the bottleneck tier in our implementation. One of the interesting future research directions would be to explore experimentally the savings that can be achieved by implementing redirection at other cluster tiers such as Web tier or application tier. We provide an example to motivate the same. Recall that to serve a client request, the application tier initiates several queries to the database tier. Redirection as implemented at the database tier may end up dispatching all the database queries (belonging to a request) to database servers at a remote cluster. In such a scenario, each query would incur network round-trip latencies which could have been avoided if the entire request had been dispatched to a Web server in the remote cluster.

## 6.10 Conclusions

In this chapter, we presented request redirection techniques that can be used by CDNs to both reduce the response times perceived by clients and to statistically multiplex resources across all the clusters. In particular, we developed a proof-of-concept request redirection mechanism for dynamic content, namely WARD in order to highlight the design principles involved. The objective of WARD is to minimize the end-to-end latency of dynamic content requests by jointly considering network and server delays. We developed an analytical model and proof-of-concept implementation that demonstrated significant reductions in average request response times. For example, for our implementation of a CDN hosting an e-commerce site and serving 300 concurrent users, WARD can reduce the average response time by 54% from 5 s to 2.3 s. Moreover, the model predicts that the performance improvements will further increase when the complexity of dynamic content processing in Web requests increases. WARD is especially suited to prevent increased response times due to short-term bottlenecks, e.g. caused by flash crowds. If the latency costs of redirection are not excessively high, WARD can also be used to exploit long-time-scale trends such as time-of-day driven workloads, and thereby avoid expensive over-provisioning of Web clusters. Finally, WARD is a server-side redirection and hence an orthogonal solution to content replication, client-side redirection and server migration policies and can therefore be seamlessly integrated with such approaches.

## Appendix

We provide a proof for the Lemma 1 here.
*Proof:* The total service time is composed of 3 durations: (i) the network latency of transferring the request to and from the remote cluster (ii) the queuing time at the cluster and (iii) the service time at the cluster.

For symmetry reasons, in the following equations, we attribute the "costs" to the receiving cluster $i$. First, we assume that the network latency between the dispatcher and a local cluster $\Delta_{ii} = 0$ and hence, network latency is incurred only by requests dispatched to a remote cluster:

$$\alpha_{ji}(\Delta_{ji} + \Delta_{ij}) \tag{6.6}$$

Second, consider the mean waiting time for a request in an cluster queue before being serviced. In general, the waiting time for for an M/G/1 queue is:

$$\frac{\rho\bar{x}(1+c^2)}{2(1-\rho)} \tag{6.7}$$

with $\rho = \lambda\bar{x}$.

For any cluster $i$, the arrival rate $\lambda$ is the sum of the requests that are dispatched from all clusters $j$ to cluster $i$, i.e. $\lambda_i = \sum_j \alpha_{ji}\lambda_j$. With this $\lambda$, (6.7) can be rewritten for a single cluster $i$ as:

$$\frac{(\sum_j \alpha_{ji}\lambda_j)\overline{x}_i^2(1+c_i^2)}{2(1-(\sum_j \alpha_{ji}\lambda_j)\overline{x}_i)} \tag{6.8}$$

Finally, the service time for a request at cluster $i$ is given by

$$\alpha_{ji}\overline{x}_i \tag{6.9}$$

The addition of these 3 terms for a set of clusters yields (6.2).

**Acknowledgements** Some of the materials presented in this chapter appeared in a preliminary format at IEEE INFOCOM'04 [34]. The author is grateful to Prof. Edward Knightly and Dr. Roger Karrer for their insightful comments and advise that contributed to the majority of the concepts explained in this chapter.

## References

1. Akamai Whitepaper: Turbo-charging Dynamic Web Sites with Akamai EdgeSuite. `http://www.akamai.com/dl/whitepapers`, 2000
2. Akamai. `http://www.akamai.com`, 2007
3. Limelight Networks. `http://www.limelightnetworks.com`, 2007
4. Mirror Image Internet. `http://www.mirror-image.com`, 2007
5. MySQL Database Server. `http://www.mysql.com`, 2007
6. NISTNET: Network Emulation Package. `http://snad.ncsl.nist.gov/itg/nistnet/`, 2007
7. PHP Scripting Language. `http://www.php.net`, 2007
8. The Apache Software Foundation. `http://www.apache.org`, 2007
9. TPC-W: Transaction Processing Council. `http://www.tpc.org`, 2007
10. Amza, C., Cecchet, E., Chanda, A., Cox, A., Elnikety, S., Gil, R., Marguerite, J., Rajamani, K., Zwaenepoel, W. Specification and implementation of dynamic content benchmarks. In: IEEE Workshop on Workload Characterization (WWC-5), Austin, TX (2002)
11. Amza, C., Cox, A., Zwaenepoel, W. Conflict-aware scheduling for dynamic content applications. In: USENIX Symposium on Internet Technologies and Systems, Seattle,WA (2003)
12. Arlitt, M., Williamson, C. Internet web servers: Workload characterization and performance implications. IEEE/ACM Trans on Networking 5(5) (1997)
13. Arlitt, M., Krishnamurthy, D., Rolia, J. Characterizing the scalability of a large web-based shopping system. ACM Trans on Internet Technology 1(1) (2001)
14. Aron, M., Sanders, D., Druschel, P., Zwaenepoel, W. Scalable content-aware request distribution in cluster-based network servers. In: USENIX ATC (2000)
15. Bouchenak, S., Mittal, S., Zwaenepoel, W. Using code transformation for consistent and transparent caching of dynamic web content. Tech. Rep. 200383, EPFL, Lausanne (2003)
16. Cardellini, V., Colajanni, M., Yu, PS. Geographic load balancing for scalable distributed web systems. In: MASCOTS, San Francisco, CA (2000)
17. Carter, R., Crovella, M. Server selection using dynamic path characterization in wide-area networks. In: IEEE INFOCOM, Kobe, Japan (1997)
18. Chen, Y., Katz, R., Kubiatowicz, J. Dynamic replica placement for scalable content delivery. In: International Workshop on Peer to Peer Systems, Cambridge, MA (2002)

19. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Welhl, B. Globally distributed content delivery. IEEE Internet Computing (2002)
20. Fei, Z., Bhattacharjee, S., Zegura, E., Ammar, M. A novel server selection technique for improving the response time of a replicated service. In: IEEE INFOCOM, San Francisco, CA (1998)
21. Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., Diot, C. Packet-level Traffic Measurement from the Sprint IP Backbone. IEEE Network Magazine (2003)
22. Guyton, J., Schwartz, M. Locating nearby copies of replicated internet servers. In: ACM SIGCOMM, Cambridge, MA (1995)
23. Jamin, S., Jin, C., Kurc, A., Raz, D., Shavitt, Y. Constrained mirror placement on the internet. In: IEEE, INFOCOM., Anchorage, AK (2001)
24. Jung, J., Krishnamurthy, B., Rabinovich, M. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: International World Wide Web Conference (2002)
25. Kangasharju, J., Ross, K., Roberts, J. Performance evaluation of redirection schemes in content distribution networks. Computer Communications 24(2) (2001)
26. Karger, D., Lehman, E., Leighton, T., Levine, M., Lewin, D., Panigrahy, R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In: ACM Symposium on Theory of Computing (1997)
27. Karger, D., Sherman, A., Berkhemier, A., Bogstad, B., Dhanidina, R., Iwamoto K., Kim, B., Matkins, L., Yerushalmi, Y. Web caching with consistent hashing. In: World Wide Web Conference (1999)
28. Karlsson, M., Mahalingam, M. Do we need replica placement algorithms in content delivery networks. In: 7th International Workshop on Web Content Caching and Distribution (WCW) (2002)
29. Kleinrock, L. "Queueing Systems, Volume II: Computer Applications". Wiley (1976)
30. Myers, A., Chuang, J., Hengartner, U., Xie, Y., Zhuang, W., Zhang, H. A secure, publisher-centric web caching infrastructure. In: Proc. of IEEE INFOCOM (2001)
31. Padmanabhan, VN., Sripanidkulchai, K. The case for cooperative networking. In: Intl. Workshop on Peer-to-Peer Systems, Cambridge, MA (2002)
32. Qiu, L., Padmanabhan, V., Voelker, G. On the placement of web server replicas. In: IEEE INFOCOM, Anchorage, AK (2001)
33. Ranjan, S., Rolia, J., Fu, H., Knightly, E. Qos-driven server migration for internet data centers. In: Intl. Workshop on Quality-of-Service, Miami, FL (2002)
34. Ranjan, S., Karrer, R., Knightly, E. Wide area redirection of dynamic content in internet data centers. In: IEEE INFOCOM, HongKong (2004)
35. Shaikh, A., Tewari, R., Agrawal, M. On the effectiveness of dns-based server selection. In: IEEE INFOCOM, Anchorage, AK (2001)
36. Villela, D., Rubenstein, D. Performance analysis of server sharing collectives for content distribution. In: Intl. Workshop on Quality-of-Service (2003)
37. Wang, L., Pai, V., Peterson, L. The effectiveness of request redirection on cdn robustness. In: OSDI, Boston, MA (2002)

# Part II
# CDN Modeling and Performance

# Chapter 7
# Economics-Informed Design of CDNs

Nicolas Christin, John Chuang and Jens Grossklags

## 7.1 Introduction

Over the past decade, both the contents available on the Internet, and its means of distribution have undergone a drastic change. In the mid-nineties, the development of electronic commerce and web services had fueled the dominance of HTTP traffic, which resulted in easily distinguishable "clients" and "servers." However, since then two important events radically transformed the Internet landscape.

First, the Napster service [1], launched in 1999, ushered the era of "Peer-to-Peer (P2P)" computing. P2P computing markedly differs from the client-server paradigm that preceded it (and now co-exists with it), in that any host at the edge of the network can act as an information provider. Second, digital production tools became accessible to the masses, through the availability of high-resolution cameras, video-recorders, or high-quality sound cards with sampling capabilities. This in turn led to a proliferation of user-generated contents, which intensified even further as information dissemination platforms, e.g. weblogs, became easier to use.

Shortly stated, end-users have gained the tools and resources to now act as active content contributors. However, at the same time (and, paradoxically, maybe because of this empowerment), novel phenomena, such as occurrences of the "tragedy of the commons" [23] have started to be observed on the network.

The tragedy of the commons is best explained by its canonical example. Consider a village pasture that can accommodate a fixed number of sheep, and is a "common" good shared among all herdsmen. Each herdsman can see a significant advantage in slightly increasing their own herd – more sheep mean more wool, and in turn more

Nicolas Christin
Carnegie Mellon University, INI and CyLab Japan, 1-3-3-17 Higashikawasaki-cho, Chuo-ku, Kobe, 650-0044, Japan, e-mail: nicolasc@cmu.edu

John Chuang
School of Information, The University of California at Berkeley, 102 South Hall, Berkeley, CA 94720, USA, e-mail: chuang@ischool.berkeley.edu

Jens Grossklags
School of Information, The University of California at Berkeley, 102 South Hall, Berkeley, CA 94720, USA, e-mail: jensg@ischool.berkeley.edu

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

income; and, if the pasture is sufficiently big, surely no one would mind an additional one or two sheep grazing on the pasture. The problem is that all herdsmen may come to the exact same conclusion, and there is now a very large increase in the number of sheep grazing on the pasture. In the worst-case (the "tragedy"), all grass is quickly depleted, sheep die of starvation, and all herdsmen lose all of their resources. The tragedy of the commons is an illustration of a *misalignment* between each individual's own *incentives* and a situation desirable from a societal point of view.

Occurrences of the tragedy of the commons have been observed in modern information networks such as P2P systems, which present strong cases of incentive misalignment. In a now widely cited study [2], Adar and Huberman identified that as many as 70% users on the Gnutella file-sharing network were not sharing any data, and were instead "free-riding", i.e. benefiting from the service without contributing anything in return. At the time of the study, the Gnutella network was essentially subsisting thanks to the 1% of users who were serving as much as 50% of all requests. In other words, free-riding had led the Gnutella network to become, in practice, very much akin to a centralized network. Such empirical evidence arguably ran counter to the founding principles behind P2P networks—supposed to be enablers facilitating collaboration and cooperation among all participants.

Related studies provide additional evidence of the high levels of free-riding in P2P networks [45], or unfairness in ad-hoc networks [25]. Such empirical research substantiates that users are, by and large, "rational", and in fact, an overwhelming majority are acting primarily with their own interests in mind. That is, they are interested in maximizing the benefits they exert from the network, while minimizing the costs associated with participation ("selfish" behavior).

How does this perceived user rationality, or selfishness, impact the practitioner interested in developing content delivery networks (CDN)? After all, are not most content delivery networks owned by a single entity, who can keep a tight control over all of its participants, and should therefore be able to solve any incentive misalignment problem?

In fact, studying of participants' incentives is relevant to several levels of interest to a CDN designer. First, a content provider may want to use end-users as helpers for the content distribution mechanism, in order to both increase the quality of service available to users, while reducing the infrastructure costs [26]. This could be a strategy particularly interesting to providers of large-scale, bandwidth intensive contents, such as video on demand. As YouTube's rising popularity [9] exemplifies, demand for such kind of services is poised to become considerable in the near future, and may require to rethink the design of the delivery infrastructure.

Second, different CDNs operated by different service providers may need to be interconnected. Given that different service providers are usually rival entities, studying their respective incentives can help devise meaningful, enforceable, service level agreements.

Third, even assuming obedient participants, e.g. as is the case when the whole CDN infrastructure is controlled and owned by a single entity, it is important to be able to characterize the amount of resources each participant is required to contribute for the system to operate properly. The analysis allows the network

architect to pinpoint potential "hot spots", that is, parts of the network that have to invest a disproportionate amount compared to their available resources. As hot spots are likely to be the primary points of failure when the network operates under heavy load, their identification is of utmost importance.

Fourth, understanding the cost and benefits associated with participation to a network helps gain valuable insights into viable customer pricing schemes, as discussed in some of the other chapters.

All of these arguments point to the direction that one needs to achieve a fundamental understanding of both the incentives at play in the design of a structured CDN, as well as the resources to be contributed by participating entities.

This type of research, while relatively new to the networking literature, has been explored for many years in applied mathematics and economics. In particular, game theory, the study of strategic interactions among non-cooperative entities, has been traditionally used for understanding and modeling competitive markets; for instance, traders competing on the stock market. Recent work has shown that, game theory applies equally well, if not better, to provide a formal background for network design and evaluation in presence of (potentially) selfish participants [46].

This chapter builds on game theory research to offer the following contributions. First, we propose a novel cost model for agents participating in a content-delivery overlay network. Using the proposed cost model, we then provide an analysis of the root causes of incentive misalignment in CDNs. We further investigate some of the network structures proposed in the literature for P2P CDNs, to determine if they are incentive compatible. We also revisit some of the traditional game-theoretic assumptions to study their applicability to CDN design.

The remainder of this chapter is structured as follows. In Sect. 7.2, we provide a short tutorial on the game-theoretic concepts we use in our analysis, and review some of the research related to our discussion. We then turn to a presentation of our cost model in Sect. 7.3. With the model defined, we proceed to analyze incentives in link establishment in CDNs in Sect. 7.4. We then relate our incentive analysis to recently proposed infrastructures in Sect. 7.5, and complement our formal analysis with numerical simulations in Sect. 7.6. We discuss how relaxing some of the assumptions made in the analysis may further impact results in Sect. 7.7. We identify future research directions in Sect. 7.8, and provide a summary and brief conclusions in Sect. 7.9.

## 7.2 Background and Related Work

Based on the empirical observations of user rationality, as discussed above, system architects have become increasingly interested in considering network participants as selfish [37] or competing [46] entities. Such concepts have made their way into deployed P2P systems: for instance, in an effort to discourage free-riding, some popular P2P systems such as KaZaA or BitTorrent [16] rely on simple incentive mechanisms. More generally, as summarized in [18, 37], a number of recent research efforts have been applying concepts from game theory and mechanism design

to networked systems in an effort to align the incentives of each (self-interested) user with the goal of maximizing the overall system performance. In this section, we first review some basic game theoretic concepts, before turning to a more comprehensive discussion of the related work.

### *7.2.1 Game-Theoretic Background*

A cornerstone of game theory is the notion of competitive equilibrium, which is used to predict user behavior and infer the outcome of a competitive game. As discussed in [37], the concept of *Nash equilibrium* is predominantly used in system research to characterize user behavior. An interesting feature of Nash equilibrium modeling is that it is useful to identify tensions between individual incentives and other motivations [24].

Assuming that each user obtains a utility dependent on the strategy she adopts (and given a set of strategies played by other users), a Nash equilibrium is defined as a set of strategies from which no user willing to maximize her own utility has any incentive to deviate [35].

Formally, we consider strategic interactions (called *games*) of the following simple form: the individual decision-makers (also called *players*) of a game simultaneously choose actions that are derived from their available strategies. The players will receive payoffs that depend on the combination of the actions chosen by each player. In short, a player decides upon an action. Based on her actions and that of other players, she gets a reward, or a penalty. In economics, such rewards and penalties are denoted by the general term "utility," and are sometimes expressed in a monetary form. In system design, such as CDN design, the utility can also denote monetary amounts, but is not restricted to them. The utility may instead characterize a user satisfaction index, function of (for instance) a reduced latency, increased throughput, or an overall better service.

More precisely, consider a set $V = \{1,...,N\}$ of players. Denote as $Z_u$ the set of *pure* (i.e. deterministic) strategies available to player $u$, $Z_{-u}$ the set of strategies available to all players other than $u$, and denote as $\zeta_u$ an arbitrary member of $u$'s strategy set. $C_u$ represents player $u$'s payoff (or *utility*) function: $C_u(\zeta_u, \zeta_{-u})$ is the payoff to player $u$ given her strategy $(\zeta_u)$ and the other players' strategies (summarized as $\zeta_{-u}$). A $N$-player game can then be described as $G = \{V; Z_u, Z_{-u}; C_u, C_{-u}\}$.

Players are in a Nash equilibrium if a change in strategies by any one of them would lead that player to obtain a lower utility than if she remained with her current strategy [35]. Formally, we can define a Nash equilibrium as follows:

**Definition 1.** A vector of pure strategies $\zeta^* = (\zeta_1^*, ..., \zeta_N^*) \in Z$ comprises a (pure) Nash equilibrium of a game $G$ if, for all $u \in V$ and for all $\zeta_u \in Z_u$, $C_u(\zeta_u, \zeta_{-u}^*) - C_u(\zeta_u^*, \zeta_{-u}^*) \leq 0$.

We can extend the notion of Nash equilibrium to probabilistic strategies. There are indeed games where players can figure that the best strategies to be played are

non-deterministic. Consider the case of a goal-keeper facing a penalty kick in a soccer game. Clearly, a strategy of always diving to the right is unappealing as opponents will, over time, be able to predict it. Instead, most goal-keepers slightly randomize their strategies, for instance, diving to the right with only a 70% probability if the kicker shoots with his left foot.

Note that such a probabilistic strategy is different from playing completely at random. In computer systems, probabilistic strategies may be useful to account for exogenous conditions that can probabilistically impact user behavior, such as the congestion on the network, the amount of noise on a wireless channel, and so forth.

A probability distribution over pure strategies is called a *mixed* strategy $\sigma_u$. Accordingly, the set of mixed strategies for each player, $\Sigma_u$, contains the set of pure strategies, $Z_u$, as degenerate cases. Each player's randomization is statistically independent of those of the other players. An $N$-player mixed strategy game can then be described as $G = \{V; \Sigma_u, \Sigma_{-u}; C_u, C_{-u}\}$.

The notion of Nash equilibrium can be extended to mixed strategies as follows.

**Definition 2.** A vector of mixed strategies $\sigma^* = (\sigma_1^*, ..., \sigma_N^*) \in \Sigma$ comprises a mixed-strategy Nash equilibrium of a game G if, for all $u \in V$ and for all $\sigma_u \in \Sigma_u$, $C_u(\sigma_u, \sigma_{-u}^*) - C_u(\sigma_u^*, \sigma_{-u}^*) \leq 0$.

The third definition that we need to present to facilitate the exposition in the rest of this chapter is that of "social optimum." While Nash strategies essentially characterize best-responses from each player to all other players' strategies, the social optimum describes the situation that is the best for all players, *taken as an aggregate*, that is, to society. Essentially, the social optimum is what a benevolent dictator would impose to ensure that the average utility over all players is the highest.

Going back to the example of sheep grazing on a shared pasture, the Nash strategy for each player (i.e. herdsman) is to let all of his/her sheep use the pasture. However, as we have discussed, all players converging to the same strategy leads to disaster. On the other hand, a socially optimal strategy would be to forcibly limit the number of sheep each herdsman can let graze on the pasture, so that all sheep can be well-fed.

Formally, the social optimum can be defined as follows:

**Definition 3.** A vector of pure strategies $\zeta^* = (\zeta_1^*, ..., \zeta_N^*) \in S$ defines a social optimum of a game $G$ if and only if for all $\zeta \in Z$, $\sum_u C_u(\zeta^*) - \sum_u C_u(\zeta) \leq 0$.

That is, the social optimum is the strategy (or strategies) that maximize the sum of all players' utilities.

### 7.2.2 Related Applications of Game-Theory to Networking Problems

Research on applications of game-theory to algorithmic and networking problems has recently enjoyed considerable interest and exposure. The reader can refer to [36] for a comprehensive treatment of recent game-theoretic contributions to algorithmic

problems. Rather than providing a comprehensive literature survey on the topic, we discuss here a relatively modest list of publications, representative of some of the major trends in the field, and which can be of direct relevance to CDN designers. Namely, we focus on research related to (1) network routing, (2) caching and (3) network formation.

**Traffic routing.** Using a specific case study, Braess' paradox [6] states that increasing a network capacity by adding a new route can sometimes adversely affect performance of the whole network. A number of papers have subsequently attempted to characterize incentives in routing data. For instance, [43] generalizes Braess' paradox by providing bounds to explore how bad the situation could get in networks where users can freely choose how to route their traffic.

Other research on game-theory applied to routing, e.g. [8, 19], has looked at how undesirable "hidden actions" can be avoided in multi-hop networks. The problem is that, in most multi-hop networks, end nodes can usually not easily monitor the individual behavior of intermediate nodes in charge of routing messages, and can instead only determine if end-to-end transmission was successful. Therefore, in the absence of incentives for intermediary nodes to properly behave, the risk is great that nodes in charge of routing traffic arbitrarily discard messages and/or downgrade their priority. Feldman et al. [19] show the importance of good contract design to overcome hidden action.

**Caching and replication.** A number of papers (e.g. [14, 20, 38]) have explored incentives in caching and replication of data over distributed networks. At a broad level, incentives issues in caching fall into two major categories. Given an existing CDN or overlay network, one may want to find out where data should be stored to satisfy performance objectives without creating incentives for some caches to defect from the network. Additionally, different CDN, operated by rival entities, may need to be interconnected – an issue directly leading to the question of incentive-compatible contract design.

**Network formation.** Finally, a large body of literature, e.g. [10, 12, 15, 17, 27], analyzes incentives in network formation. That is, given an existing network of rational and potentially competing entities, these papers describe how an additional rational participant may elect to join the network. This problem is easy to view as a graph problem, expressing the existing network as a set of nodes, linked by a set of edges. The questions posed by network formation can then be phrased as follows. What are the new links (edges) an incoming node will build to nodes in the existing network? Will some edges be deleted when the new node joins the network? Link creation can be modeled as an incentive problem: a node will only create or remove a link to another node if it increases its own utility by doing so.

Cost models for network formation were initially proposed to study the formation of social and economic networks [27], for instance, to determine how likely were academics to co-author papers with other academics. These socio-economic models served as the foundation for more recent studies [14, 17], which characterize peer-to-peer network formation as a non-cooperative game, where nodes have an incentive to participate in the network, but want to minimize the price

they pay for doing so. These initial works focused on distance to other nodes and degree of connectivity as the key parameters in the computation of each node's utility function. Subsequent works extended the research by considering the load imposed on each node in addition to the distance to other nodes and degree of connectivity [10, 11]. In the rest of this chapter, we elaborate on such cost models.

## 7.3 Modeling Costs and Benefits in CDNs

The centerpiece of this chapter is a cost model, which aims at assessing the amount of resources a given node must contribute to participate in a CDN or overlay network, and the benefits it exerts from participation. We express the benefits of participating in the CDN in terms of a cost reduction. We point out that the model we describe here is actually not specific to CDNs. It can indeed apply to any network where nodes request and serve items, or serve requests between other nodes. In addition to CDNs, this includes peer-to-peer file-sharing systems [2, 16], ad-hoc networks [39], distributed lookup services [42, 48], or application-layer multicast overlays [5, 13, 30], to name a few examples. Table 7.1 provides a summary of notations used in the model.

Formally, we define a (CDN, overlay, ...) network by a quadruplet $(V,E,K,F)$, where $V$ is the set of nodes in the network, $E$ is the set of directed edges, $K$ is the set of items in the network, and $F : K \to V$ is the function that assigns items to nodes.

Each node $u \in V$ is assigned a unique identifier (integer or string of symbols), which, for the sake of simplicity, we will also denote by $u$. We define by $K_u = \{k \in K : F(k) = u\}$ the set of items stored at node $u \in V$. We have $K = \bigcup_u K_u$, and we assume, without loss of generality, that the sets $K_u$ are disjoint. Indeed, if an item is stored on several nodes (replication), the replicas can be viewed as different items with the exact same probability of being requested.

**Table 7.1** Summary of notations

| **Contents Properties** | |
|---|---|
| $K_u$ | Set of items held by node $u$ |
| $F$ | Mapping function of contents to nodes |
| $X$ | Node source of a request |
| $Y$ | Item requested |

| **Network Properties** | |
|---|---|
| $V$ | Set of nodes in the network |
| $E$ | Set of edges in the network |
| $K$ | Set of content items in the network |
| $N$ | Number of nodes |
| $D$ | Dimension |
| $\Delta$ | Base |

| **Network Metrics** | |
|---|---|
| $t_{u,v}$ | Hop count between nodes $u$ and $v$ |
| $\chi_{v,w}(u)$ | Test if $u$ is on the path from $v$ to $w$ |

| **Cost Metrics and Functions** | |
|---|---|
| $L_u$ | Latency cost node $u$ |
| $R_u$ | Routing cost experienced by node $u$ |
| $S_u$ | Service cost experienced by node $u$ |
| $M_u$ | Maintenance cost experienced by node $u$ |
| $l_{u,k}$ | Nominal cost for node $u$ to request item $k$ |
| $r_{u,k}$ | Nominal cost for node $u$ to forward item $k$ |
| $s_{u,k}$ | Nominal cost for node $u$ to serve item $k$ |
| $m_{u,v}$ | Nominal cost for node $u$ to maintain information about node $v$ |
| $C_u$ | Total cost experienced by node $u$ |
| $C$ | Total cost experienced by the network |

We characterize each request with two independent random variables, $X \in V$ and $Y \in K$, which denote the node $X$ making the request, and the item $Y$ being requested, respectively.

Consider a given node $u \in V$. Every time an item $k \in K$ is requested in the entire network, node $u$ is in one of four situations:

**1. Idle.** Node $u$ does not hold or request $k$, and is not on the routing path of the request. Node $u$ is not subject to any cost.

**2. Issuing the request.** Node $u$ requests item $k$. In our model, we express the benefits of participating in a peer-to-peer network in terms of latency reduction, similar to related proposals, e.g. [17]. In particular, we assume that the farther the node $v$ holding $k$ is from $u$ (in a topological sense), the costlier the request is. If there is no path between nodes $u$ and $v$, the request cannot be carried out, which yields an infinite cost. More precisely, we model the cost incurred by node $u$ for requesting $k$ as $l_{u,k}t_{u,v}$, where $t_{u,v}$ is the number of hops between nodes $u$ and $v$, and $l_{u,k}$ is a (positive) proportional factor. We define the *latency cost* experienced by node $u$, $L_u$, as the sum of the individual costs $l_{u,k}t_{u,v}$ multiplied by the probability $k \in K_v$ is requested, that is

$$L_u = \sum_{v \in V} \sum_{k \in K_v} l_{u,k} t_{u,v} \Pr[Y = k] \,, \tag{7.1}$$

with $t_{u,v} = \infty$ if there is no path from node $u$ to node $v$, and $t_{u,u} = 0$ for any $u$. With this definition, to avoid infinite costs, each node has an incentive to create links such that all other nodes holding items of interest can be reached. An alternative is to store or cache locally all items of interest so that the cost of all requests reduces to $l_{u,k}t_{u,u} = 0$.

As a concrete example of the latency cost, consider the Domain Name Service (DNS) [33]. DNS can be viewed as an overlay network using a tree topology, where the leaf nodes are the DNS clients, and all other nodes are DNS servers. Consider that a client $u$ wants to access a DNS record $k$ so unusual that the query has to be redirected all the way to a DNS root server $v$. Here, we might have a relatively high value for the number of hops between $u$ and $v$, say $t_{u,v} = 5$. After the query is resolved, $u$'s primary DNS server, $u'$, will have a copy of $k$, thereby reducing the latency for a request from $u$ for $k$ from $t_{u,v} = 5$ to $t_{u,u'} = 1$. The notion of latency is captured in (7.1) as observed by $u$ in terms of a weighted average over all possible queries $u$ can make. The weights $l_{u,k}$ are introduced to express the relative value of one record compared to another. In our DNS example, if, from node $u$'s perspective, the ability to resolve $k =$ `www.google.com` is considered 100 times more valuable than the ability to resolve $k' =$ `dogmatix.sims.berkeley.edu`, we should have $l_{u,k} = 100 \cdot l_{u,k'}$.

**3. Serving the request.** Node $u$ holds item $k$, and pays a price $s_{u,k}$ for serving the request. For instance, in a filesharing system, the node uses some of its upload capacity to serve the file. We define the *service cost* $S_u$ incurred by $u$, as the expected value of $s_{u,k}$ over all possible requests. That is,

$$S_u = \sum_{k \in K_u} s_{u,k} \Pr[Y = k] \,.$$

Going back to our earlier DNS example, copying the record $k$ to the server $u'$ implies that $u'$ has to use some resources to store the copy of the record $k$, which our cost model characterizes by an increase in the service cost $S_{u'}$. In the DNS example, for a given DNS server, the cost of serving a DNS record $k$ is the same for all $k$, so that we have for all $k$, $s_{u',k} = s_{u'}$, which corresponds to the cost of storing one record.

**4. Forwarding the request.** Node $u$ does not hold or request $k$, but has to forward the request for $k$, thereby paying a price $r_{u,k}$. The overall *routing cost* $R_u$ suffered by node $u$ is the average over all possible items $k$, of the values of $r_{u,k}$ such that $u$ is on the path of the request. That is, for $(u,v,w) \in V^3$, we consider the binary function

$$\chi_{v,w}(u) = \begin{cases} 1 \text{ if } u \text{ is on the path from } v \text{ to } w, \text{ excluding } v \text{ and } w, \\ 0 \text{ otherwise,} \end{cases}$$

and express $R_u$ as

$$R_u = \sum_{v \in V} \sum_{w \in V} \sum_{k \in K_w} r_{u,k} \Pr[X = v] \Pr[Y = k] \chi_{v,w}(u) \; . \tag{7.2}$$

In our DNS example, the routing cost denotes the resources used by a server which receives a query for $k$, cannot resolve it, and has to redirect the query to a DNS server higher up in the tree, averaged over all possible queries.

In addition, each node keeps some state information so that the protocol governing the network operates correctly. In most overlay network protocols, each node $u$ has to maintain a neighborhood table and to exchange messages with all of its neighbors, that is, the nodes $v$ for which an edge $(u,v)$ exists. Denoting by $\mathcal{N}(u)$ the set of neighbors of $u$, we characterize a *maintenance cost* $M_u$, as

$$M_u = \sum_{v \in \mathcal{N}(u)} m_{u,v} \; ,$$

where $m_{u,v} \geq 0$ denotes the cost incurred by node $u$ for maintaining a link with its neighbor $v \in \mathcal{N}(u)$.

Returning to the DNS example, the maintenance cost characterizes the resources used by the DNS server $u$ to maintain information about all the other servers $u$ might contact (or refer to) when a query cannot be answered locally.

Last, we define the *total cost* $C_u$ imposed on node $u$ as

$$C_u = L_u + S_u + R_u + M_u \; .$$

We can use $C_u$ to compute the total cost of the network, $C = \sum_{u \in V} C_u$. Note that the expression of $C_u$ only makes sense if $S_u$, $R_u$, $M_u$, and $L_u$ are all expressed using the same unit. Thus, the coefficients $s_{u,k}$, $r_{u,k}$, $l_{u,k}$ and $m_{u,v}$ have to be selected appropriately. For instance, if $l_{u,k}$ is given in monetary units per hop per item, then $m_{u,v}$ has to be expressed in monetary units.

## 7.4 Social Optima and Nash Equilibria

In this section, we investigate the network structures that constitute a social optimum or a Nash equilibrium in the cost model defined above. Formally, we define a network structure as a "geometry" [22], that is, as a collection of nodes and edges, or topology, associated with a route selection mechanism. Unless otherwise noted, we assume shortest path routing, and distinguish between different topologies. We discuss a few simplifications useful to facilitate our analysis, before characterizing some possible social optima. In this discussion, we mostly focus on results, and insights. In particular, we do not detail most of the technical proofs, and instead refer the interested reader to [11] for a more complete treatment.

**Assumptions.** For the remainder of this section, we consider a network of $N > 0$ nodes, where, for all $u \in V$ and $k \in K$, $l_{u,k} = l$, $s_{u,k} = s$, $r_{u,k} = r$, and for all $u \in V$ and $v \in V$, $m_{u,v} = m$. In other words, we assume that the costs associated with incurring a one-hop latency, serving one request, routing one request, or maintaining one link, are the same on all nodes, irrespective of the item requested or served. While very crude in general, this simplification is relatively accurate in the case of a network of homogeneous nodes, containing fixed-sized items, and homogeneous links. This is particularly true of indexing mechanisms based on distributed hash tables (see for instance, [40, 42, 44, 48, 49]).

We suppose that the network is in a steady-state regime, i.e. nodes do not join or leave the network, so that the values $l$, $s$, $r$ and $m$ are constants. We also suppose that requests are uniformly distributed over the set of nodes, that is, for any node $u$, $\Pr[X = u] = 1/N$.

For our analysis, we make a further simplification by choosing the mapping function $F$ such that all nodes have an equal probability of serving a request. In other words, $\sum_{k \in K_u} \Pr[Y = k] = 1/N$, which implies

$$S_u = \frac{s}{N} \ ,$$

regardless of the geometry used. This assumption will be removed when we proceed to numerical simulations in Sect. 7.6. Moreover, if we use $E[x]$ to denote the *expected value* of a variable $x$, Eqs. (7.1) and (7.2) reduce to

$$L_u = lE[t_{u,v}] \ ,$$

and

$$R_u = rE[\chi_{v,w}(u)] \ ,$$

respectively. Also, because each node $u$ has $\deg(u)$ neighbors, we immediately obtain

$$M_u = m \deg(u) \ .$$

Last, we assume that no node is acting maliciously.

### 7.4.1 Full Mesh

In our investigation of possible social optima, let us first consider a full mesh, that is, a network where any pair of nodes is connected by a bidirectional edge, i.e. $t_{u,v} = 1$ for any $v \neq u$. Nodes in a full mesh never any route any traffic, as there is always a direct connection between two nodes, and $\deg(u) = N - 1$. Thus, for all $u$, $R_u = 0$, $L_u = l(N-1)/N$, and $M_u = m(N-1)$. With $S_u = s/N$, we get $C_u = s/N + l(N-1)/N + m(N-1)$, and, summing over $u$,

$$C(\text{full mesh}) = s + l(N-1) + mN(N-1) \; . \tag{7.3}$$

Let us remove a link from the full mesh, for instance the link $0 \to 1$. The maintenance cost at node 0, $M_0$, decreases by $m$. However, to access the items held at node 1, node 0 now has to send a request through another node. The actual mechanism that informs node 0 of which node to contact to send a request to node 1 is irrelevant to this discussion: One can for instance assume without loss of generality that nodes periodically advertise their list of neighbors. Assume that node 0 contacts node 2. As a result, $L_0$ increases by $l/N$, and the routing cost at node 2, $R_2$, increases by $r/N^2$.

Hence, removing the link $0 \to 1$ causes a change in the total cost $\Delta C = -m + l/N + r/N^2$. If $\Delta C \geq 0$, removing a link causes an increase of the total cost, and the full mesh is the social optimum. In particular, we have shown:

**Proposition 1.** *The full mesh is the social optimum if the maintenance cost is "small enough," that is, if*

$$m \leq \frac{l}{N} + \frac{r}{N^2} \; . \tag{7.4}$$

Note that, as $N \to \infty$, the condition in Eq. (7.4) tends to $m = 0$. In fact, we can also express $\Delta C \geq 0$ as a condition on $N$ that reduces to $N \leq \lfloor l/m + r/l \rfloor$ when $m \ll l^2/r$, using a first-order Taylor series expansion.

We can here again draw a parallel with DNS to illustrate condition (7.4). As long as the number of Internet hosts remained reasonably small, each host used a large `HOSTS.TXT` file to directly resolve hostnames into IP addresses, effectively creating a full mesh for the naming overlay: Each node knew about all the other nodes.[1] The DNS protocol was only introduced when the number of hosts on the Internet grew large enough to render maintaining all information in a single, distributed file impractical.

### 7.4.2 Star Network

Suppose now that Eq. (7.4) does not hold, and consider a star network. Let $u = 0$ denote the center of the star, which routes all traffic between peripheral nodes. That

[1] Note that we are here only concerned with name resolution. Updating and disseminating the `HOSTS.TXT` file is a separate issue, and was actually done in a centralized manner [33].

is, $\chi_{v,w}(0) = 1$ for any $v \neq w$ ($v, w > 0$). One can easily show that $R_0 = r(N-1)(N-2)/N^2$, $L_0 = l(N-1)/N$ and $M_0 = m(N-1)$, so that the cost $C_0$ incurred by the center of the star is

$$C_0 = m(N-1) + \frac{s}{N} + l\frac{N-1}{N} + r\frac{(N-1)(N-2)}{N^2} \; . \tag{7.5}$$

Peripheral nodes do not route any traffic, i.e. $R_u = 0$ for all $u > 0$, and are located at a distance of one from the center of the star, and at a distance of two from the $(N-2)$ other nodes, giving $L_u = l(2N-3)/N$. Further, $\deg(u) = 1$ for all peripheral nodes. Hence, $M_u = m$, and the individual cost imposed on nodes $u > 0$ is

$$C_u = m + \frac{s}{N} + l\frac{2N-3}{N} \; . \tag{7.6}$$

**Proposition 2.** *$C_0 = C_u$ can only hold when $N$ is a constant, or when $l = r = m = 0$.*

*Proof:* Assume that $C_0 - C_u = 0$. Because $N \neq 0$, $C_0 - C_u = 0$ is equivalent to $N^2(C_0 - C_u) = 0$. Using the expressions for $C_0$ and $C_u$ given in Eqs. (7.5) and (7.6), rewriting the condition $N^2(C_0 - C_i) = 0$ as a polynomial in $N$, and factoring by $(N-2)$, we obtain

$$(N-2)(mN^2 - (l-r)N - r) = 0 \; .$$

A polynomial in $N$ is constantly equal to zero if and only if all of the polynomial coefficients are equal to zero, which, here, imposes $l = r = m = 0$.
Since the difference $C_0 - C_u$ quantifies the (dis)incentive to be a priori in the center of the star, Proposition 2 tells us that there is a (dis)incentive to be in the center of the star in a vast majority of cases. In practice, star-like networks denote highly centralized architectures, where resources are concentrated in one "hub." The hub, usually held by the content provider, gets appropriately compensated (e.g., financially) for its central position as unique server.

Next, we compute the total cost of the star, and determine under which condition it is a social optimum. Summing Eqs. (7.5) and (7.6), we obtain

$$C(\text{star}) = 2m(N-1) + s + 2l\frac{(N-1)^2}{N} + r\frac{(N-1)(N-2)}{N^2} \; . \tag{7.7}$$

**Proposition 3.** *For any number of nodes $N \geq 3$, the star is a social optimum, if (i) Eq. (7.4) does not hold and (ii) all links are bidirectional, i.e. for any $u \in V$ and $v \in V$, if $(u \to v) \in E$ then $(v \to u) \in E$.*

The proof, available in [11] essentially consists in repeatedly removing links from a full mesh, until the network is about to be disconnected, thereby showing that the star configuration actually maximizes the aggregate utility of all players.

Let us make two remarks regarding Proposition 3. First, Proposition 3 does not guarantee that the star is a unique social optimum. In fact, in the limit case where

$m = l/N + r/N^2$, adding any number of "shortcuts" between peripheral nodes of a star still results in a social optimum. Second, the assumption that the links are bidirectional is crucial for Proposition 3 to hold for any $N$. For instance, if we allow for unidirectional links, it can be shown that, if $m$ is large enough and $N$ remains small, the unidirectional ring $0 \to 1 \to \ldots \to N \to 1$ has a lower cost than the star network.

However, while finding the social optimum when unidirectional links are allowed is an open problem, we conjecture that the star network still plays a predominant role, and that geometries such as the unidirectional ring may only appear under very stringent conditions. More concisely, the above analysis tells us that, when the number of links to maintain becomes too high to make a full mesh an attractive solution, a centralized network is generally optimal from the point of view of resource consumption.

### *7.4.3 Nash Equilibria*

Assume now that each node can choose which links it maintains, but does not have any control over the items it holds, and honors all routing requests. In other words, each node is selfish when it comes to link establishment, but is obedient once links are established. When each node $u$ is (perfectly) rational, i.e. tries to minimize its individual cost $C_u$ given the behavior of all other nodes, the resulting topology constitutes a Nash equilibrium, per our definition in Sect. 7.2.

Even though the existence or uniqueness of a Nash equilibrium is in general not guaranteed, the following results yield some insight on the possible equilibria that may occur in our proposed cost model.

**Proposition 4.** *If $m < l/N$, the full mesh is a unique (pure) Nash equilibrium.*

**Proposition 5.** *If $m \geq l/N$, the star network is a pure Nash equilibrium.*

These results can be proven by showing that, under the condition of Proposition 4, removing links from a full mesh decreases the utility of at least one participant. Conversely, under the condition of Proposition 5, adding a link to the star network results in lowering the utility of at least one node [11].

Propositions 4 and 5, tell us that, if maintaining links is cheap, or if the network is small, the only Nash equilibrium is the full mesh. If maintaining links is more expensive, or if the network is large, a star network is a possible Nash equilibrium; we cannot guarantee unicity of the equilibrium, however. For instance, in the limit case $m = l/N$, any network created by adding an arbitrary number of links between peripheral nodes of a star constitutes a Nash equilibrium.

We note that these results are a generalization of the results obtained by the authors of [27], who used a different cost model, which does not take the routing cost into account.

### 7.4.4 Interpretation

We summarize our findings in Fig. 7.1, where we discriminate between social optima and Nash equilibria according to the value of the unit maintenance cost $m$.

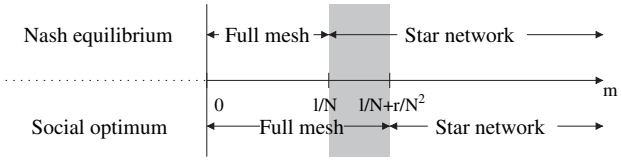



**Fig. 7.1** Social optimum and Nash equilibrium. Incentives of individual nodes are not aligned with the social optimum in the interval $[l/N, l/N + r/N^2]$

For $m < l/N$, the full mesh is both a Nash equilibrium and a social optimum; for $m > l/N + r/N^2$, the star network is both a Nash equilibrium and a social optimum. In both cases, the incentives of each node are aligned with the most efficient overall usage of the resources in the network. This is represented by the white areas in the figure.

The most interesting region in Fig. 7.1 is perhaps the gray area, in which individual incentives and overall resource usage are conflicting. This area corresponds to the parameter range $l/N < m < l/N + r/N^2$, whose size solely depends on $r$. Stated differently, under the assumption that all nodes have an identical probability of serving a request, *the social optimum may significantly deviate from a Nash equilibrium as soon as nodes value the resources they use to forward traffic on behalf of other nodes*.

As a corollary, a network where "forwarding comes for free" (i.e. $r = 0$), e.g. where bandwidth and computational power are extremely cheap, is ideal from the system designer's perspective, because individual incentives should produce a socially optimal solution. Unfortunately, in most networks, the price paid for forwarding data cannot be neglected, which suggests that our cost model is better suited at capturing possible disincentives than previous models solely based on node degree (i.e. maintenance costs) and hop count (i.e. latency costs).

For a CDN designer, this result leads to an important insight. Either the CDN must be designed to avoid as much as possible having nodes forwarding significant amount of data. Or, if that is not possible, forwarding nodes likely have to be compensated as a function of the routing load they must support.

## 7.5 Analyzing Existing Structures

In the discussion in the previous section, we have ignored robustness against attacks, fault-tolerance, or potential performance bottlenecks. All these factors pose practical challenges in a centralized approach, as does providing an incentive to occupy

(or relinquish) the central position of a star. Using a full mesh avoids most of these concerns, but, as we have seen, is only a solution for a modest number of nodes.

Many research efforts have been directed at designing network geometries that provide reasonable performance, while addressing the aforementioned robustness concerns. In this section, we use the cost model proposed above to evaluate a few of the routing geometries that have been recently proposed for overlay networks in the networking literature. We focus on de Bruijn graphs, $D$-dimensional tori, PRR trees, and Chord rings. We present the various costs experienced by a node in each geometry; here again, we focus on the results and intuitions behind them, deferring the more technical steps to reports published elsewhere [11]. We later compare our results with those obtained in our study of the social optima and Nash equilibria.

### 7.5.1 De Bruijn Graphs

Due to very desirable properties, such as short average routing distance and high resiliency to node failures, De Bruijn graphs are used in a number of overlay network maintenance algorithms [28, 31, 34]. In a de Bruijn graph, any node $u$ is represented by an identifier string $(u_1,\ldots,u_D)$ of $D$ symbols taken from an alphabet of size $\Delta$. The node represented by $(u_1,\ldots,u_D)$ links to each node represented by $(u_2,\ldots,u_D,x)$ for all possible values of $x$ in the alphabet. The resulting directed graph has a fixed out-degree $\Delta$, and a diameter $D$.

Denote by $V'$ the set of nodes such that the identifier of each node in $V'$ is of the form $(h,h,\ldots,h)$. Nodes in $V'$ link to themselves, so that $M_u = m(\Delta-1)$ for $u \in V'$. For nodes $u \notin V'$, the maintenance cost $M_u$ is $M_u = m\Delta$. The next two lemmas allow us to show that the routing cost at each node also depends on the position of the node in the graph.

**Lemma 1.** *With shortest-path routing, nodes $u \in V'$ do not route any traffic, and $R_u = 0$.*

This lemma can be proven by contradiction [11].

**Lemma 2.** *The number of routes $\rho_u$ passing through a given node u, or node loading, is bounded by $\rho_u \leq \rho_{\max}$ with*

$$\rho_{\max} = \frac{(D-1)(\Delta^{D+2}-(\Delta-1)^2)-D\Delta^{D+1}+\Delta^2}{(\Delta-1)^2}\,.$$

*The bound is tight, since it can be reached when $\Delta \geq D$ for the node $(0,1,2,\ldots,D-1)$.*

The proof is similar in spirit to the proof used in [47] to bound the maximum number of routes passing through a given edge, and a complete derivation can be found in [11]. Let us sketch the strategy here. First, notice that, determining an upper bound on the number of paths of length $k$ that pass through a given node $u$ is equivalent to computing the maximum number of strings of length $D+k$ that include

node $u$'s identifier as a substring. We then sum over $k$ for $k \in [1,D]$, and obtain an intermediary bound, which we improve by removing all strings of length $2D$ that denote a cycle, as a cycle cannot characterize a shortest path in a de Bruijn graph.

From Lemmas 1 and 2, we infer that, in a de Bruijn graph, for any $u$, $v$ and $w$, $0 \leq \Pr[\chi_{v,w}(u) = 1] \leq \rho_{\max}/N^2$. Because $\chi_{v,w}(u)$ is a binary function, $\Pr[\chi_{v,w}(u) = 1] = E[\chi_{v,w}]$, and we finally obtain $0 \leq R_i \leq R_{\max}$ with

$$R_{\max} = \frac{r\rho_{\max}}{N^2} .$$

We next present upper and lower bounds on the latency cost, $L_{\max}$ and $L_{\min}$. It can be shown [11] that nodes $u \in V'$ are subject to $L_u = L_{\max}$, where

$$L_{\max} = l\frac{D\Delta^{D+1} - (D+1)\Delta^D + 1}{N(\Delta - 1)} .$$

We can lower bound $L_u$ by

$$L_{\min} = \frac{l}{N}\left(D\Delta^D + \frac{D}{\Delta - 1} - \frac{\Delta(\Delta^D - 1)}{(\Delta - 1)^2}\right),$$

and we observe that $L_u = L_{\min}$ for the node $(0, 1, \ldots, D-1)$ when $\Delta \geq D$.

Note that, the expressions for both $L_{\min}$ and $L_{\max}$ can be further simplified for $N = \Delta^D$, that is, when the identifier space is fully populated.

### 7.5.2 *D-dimensional Tori*

We next consider $D$-dimensional tori, where each node is represented by $D$ Cartesian coordinates, and has $2D$ neighbors, for a maintenance cost of $M_u = 2mD$ for any $u$. This type of routing geometry is for instance used in CAN [42].

Routing at each node is implemented by greedy forwarding to the neighbor with the shortest Euclidean distance to the destination. We assume here that each node is in charge of an equal portion of the $D$-dimensional space. This constraint could also be expressed using the slightly stronger assumption that $N^{1/D}$ is an integer, and that all possible sets of Cartesian coordinates $(u_1, \ldots, u_D)$ (where each $u_i$ maps to an integer in $[0, N^{1/D} - 1]$) map to a node. In other words, we assume the identifier space $(u_1, \ldots, u_D)$ is fully populated.

From [31], we know that the average length of a routing path is $(D/4)N^{1/D}$ hops for $N$ even, and $(D/4)N^{1/D} + D/4 - o(1)$ for $N$ odd. Because we assume that the $D$-dimensional torus is equally partitioned, by symmetry, we conclude that for all $u$,

$$L_u = l\frac{DN^{1/D}}{4} ,$$

using the same approximation as in [42] that the average length of a routing path is almost equal $(D/4)N^{1/D}$ hops even for $N$ odd.

To determine the routing cost $R_u$, we compute the node loading as a function $\rho_{u,D}$ of the dimension $D$. With our assumption that the $D$-torus is equally partitioned, $\rho_{u,D}$ is the same for all $u$ by symmetry.

**Lemma 3.** *In a D-torus completely populated with N nodes, the node loading at any node u is given by*

$$\rho_{u,D} = 1 + N^{\frac{D-1}{D}} \left( -N^{\frac{1}{D}} + D \left( N^{\frac{1}{D}} - 1 + \left( \left\lfloor \frac{N^{\frac{1}{D}}}{2} \right\rfloor - 1 \right) \left( \left\lceil \frac{N^{\frac{1}{D}}}{2} \right\rceil - 1 \right) \right) \right). \tag{7.8}$$

This lemma can be proven by induction on the dimension $D$ [11]. First, notice that for $D = 1$, the node loading $\rho_{u,1}$ at each node $u$, is equal to the sum of the number of routes passing through *each* node when the source is held fixed. For instance, for $N = 7$, we have for any $u$, $\rho_{u,1} = 0+1+2+2+1+0 = 6$. We get two different expressions for $N$ even and $N$ odd, which can be summarized as the general condition $\rho_{u,1} = \left(\lfloor \frac{N}{2} \rfloor - 1\right)\left(\lceil \frac{N}{2} \rceil - 1\right)$.

The key observation to compute the number of routes $\rho_{u,D}$ passing through each node $u$ for $D > 1$, is that there are several equivalent shortest paths along the Cartesian coordinates, because the coordinates of two consecutive nodes in a path cannot differ in more than one dimension. Consider for instance, for $D = 2$, going from node (0,0) to node (1,1): both $\mathscr{P}_1 = (0,0) \to (1,0) \to (1,1)$ and $\mathscr{P}_2 = (0,0) \to (0,1) \to (1,1)$ are equivalent shortest paths. Therefore, we can always pick the path that corrects coordinates successively, starting with the first coordinate, i.e. $\mathscr{P}_1$ in the above example.

Denote the source of the route as node $v$, the destination of the route as node $w$, and the coordinates of $u$, $v$, and $w$ by $(u_1, \ldots, u_D)$, $(v_1, \ldots, w_D)$, and $(w_1, \ldots, w_D)$. Only three possibilities for $u$ are allowed by the routing scheme that corrects coordinates one at a time: 1) node $u$ has the same $D$-th coordinate as both the source $v$ and the destination $w$ (i.e. $u_D = v_D = w_D$), 2) nodes $u$, $v$ and $w$ all differ in their $D$-th coordinate, i.e. $u_D \neq v_D \neq w_D$, and 3) node $u$ has the same $D$-th coordinate as node $v$, and a $D$-th coordinate different from that of the destination $v$ ($u_D = v_D$, $u_D \neq w_D$). By computing the node loadings for each case and summing, we obtain the value for $\rho_{u,D}$ given in (7.8).

For all $u$, $R_u$ immediately follows from $\rho_{u,D}$ with

$$R_u = r \frac{\rho_{u,D}}{N^2} .$$

### 7.5.3 PRR Trees

We next consider the variant of PRR trees [40] used in Pastry [44] or Tapestry [49]. Nodes are represented by a string $(u_1, \ldots, u_D)$ of $D$ digits in base $\Delta$. Each node is

connected to $D(\Delta-1)$ distinct neighbors of the form $(u_1,\ldots,u_{i-1},x,y_{i+1},\ldots,y_D)$, for $i=1\ldots D$, and $x\neq u_i\in\{0,\ldots,\Delta-1\}$. The resulting maintenance cost is $M_u=mD(\Delta-1)$.

Among the different possibilities for the remaining coordinates $y_{i+1},\ldots,y_D$, the protocols generally select a node that is nearby according to a proximity metric. We here assume that the spatial distribution of the nodes is uniform, and that the identifier space is fully populated, which enables us to pick $y_{i+1}=u_{i+1},\ldots,y_D=u_D$. Thus, two nodes $u$ and $v$ at a distance of $n$ hops differ in $n$ digits.

There are $\binom{D}{n}$ ways of choosing which digits are different, and each such digit can take any of $(\Delta-1)$ values. So, for a given node $u$, there are $\binom{D}{n}(\Delta-1)^n$ nodes that are at distance $n$ from $u$. Multiplying by the total number of nodes $N=\Delta^D$, and dividing by the total number of paths $N^2$, we infer that, for all $u$, $v$, and $w$, we have

$$\Pr[t_{u,v}=n]=\frac{\binom{D}{n}(\Delta-1)^n}{N}\ . \tag{7.9}$$

Now, for any $u$ and $v$ such that $t_{u,v}=n$, because routes are unique, there are exactly $(n-1)$ different nodes on the path between $u$ and $v$. So, the probability that a node $w$ picked at random is on the path from $u$ to $v$ is

$$\Pr[\chi_{u,v}(w)=1|t_{u,v}=n]=\frac{n-1}{N}\ . \tag{7.10}$$

We apply the total probability theorem to (7.9) and (7.10), express the right-hand side as a function of the derivative of a series, and use the binomial theorem to obtain that the expression for $\Pr[\chi_{u,v}(w)=1]$, which we multiply by $r$ to obtain the routing cost,

$$R_u=r\frac{\Delta^{D-1}(D(\Delta-1)-\Delta)+1}{N^2}\ . \tag{7.11}$$

To compute the access cost $L_u$, we use the relationship

$$L_u=lE[t_{u,v}]=l\sum_{n=1}^{D}k\Pr[t_{u,v}=n]=l\frac{D\Delta^{D-1}(\Delta-1)}{N}\ , \tag{7.12}$$

using the expression for $\Pr[t_{u,v}=n]$ given in (7.9), and relying, here again, on the binomial theorem [11]. Note that, for $N=\Delta^D$, (7.12) reduces to $L_u=lD(\Delta-1)/\Delta$.

### *7.5.4 Chord Rings*

In a Chord ring [48], nodes are represented using a binary string (i.e. $\Delta=2$). When the ring is fully populated, each node $u$ is connected to a set of $D$ neighbors, with identifiers $((u+2^p)\mod 2^D)$ for $p=0\ldots D-1$. An analysis similar to that carried out for PRR trees yields $R_u$ and $L_u$ as in Eqs. (7.11) and (7.12) for $\Delta=2$. Simulations confirm this result [48].

### 7.5.5 Discussion

The analytical results we have discussed in this section can serve a number of purposes. First, they confirm that all of the routing geometries considered here have the same asymptotic behavior: the routing costs decrease in $\log N$, while the latency costs grow with $\log N$. Second, while these asymptotic results are well known (see for instance [22, 31, 42, 48]), the main advantage of the game-theoretic analysis discussed above is to provide closed-form equations that can be used for tuning configuration parameters such as $\Delta$ or $D$ in function of the relative importance of each cost, e.g. routing cost vs. latency cost.

Third, our analysis provides us with a baseline we can use in a comparison with (1) the social optima and/or Nash equilibria and (2) more realistic scenarios where the identifier space is sparsely populated or where some items are more popular than others. These comparisons are the object of the next section.

## 7.6 Numerical Evaluation

We present here selected Monte Carlo simulations to compare between the different analytic results we obtained for different network geometries. We also complement the analysis by investigating numerically the effect of relaxing the assumptions that all items have identical popularity, and that the identifier space is fully populated.

**Comparison with Social Optima.** Let us first illustrate numerically the analysis of Sect. 7.5. In Table 7.2, we consider five de Bruijn graphs with different values for $\Delta$ and $D$, and $X$ and $Y$ i.i.d. uniform random variables. Table 7.2 shows that while the latency costs of all nodes are comparable, the ratio between $R_{\max}$ and the second best case routing cost,[2] $R'_{\min}$, is in general significant. Thus, if $r \gg l$, there can be an incentive for the nodes with $R_u = R_{\max}$ to defect. For instance, these nodes may leave the network and immediately come back, hoping to be assigned a different identifier $u' \neq u$ with a lower cost. Additional mechanisms, such as enforcing a cost of entry to the network, may be required to prevent such defections.

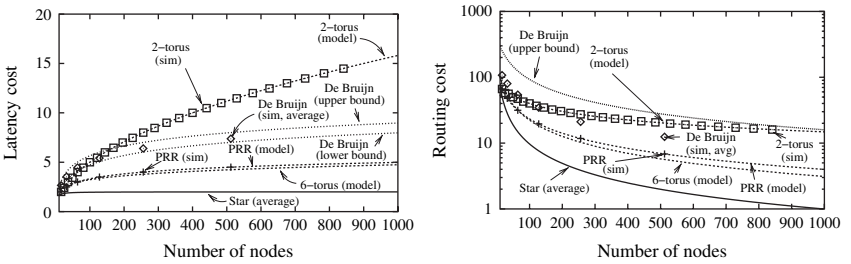
**Table 7.2** Asymmetry in costs in a de Bruijn graph ($l = 1, r = 1{,}000$)

| $(\Delta, D)$ | $L_{\min}$ | $L_{\max}$ | $\frac{L_{\max}}{L_{\min}}$ | $R'_{\min}$ | $R_{\max}$ | $\frac{R_{\max}}{R'_{\min}}$ |
|---|---|---|---|---|---|---|
| (2, 9) | 7.18 | 8.00 | 1.11 | 3.89 | 17.53 | 4.51 |
| (3, 6) | 5.26 | 5.50 | 1.04 | 2.05 | 9.05 | 4.41 |
| (4, 4) | 3.56 | 3.67 | 1.03 | 5.11 | 13.87 | 2.71 |
| (5, 4) | 3.69 | 3.75 | 1.02 | 1.98 | 5.50 | 2.78 |
| (6, 3) | 2.76 | 2.80 | 1.01 | 5.38 | 9.99 | 1.86 |

[2] That is, the minimum value for $R_u$ over all nodes but the $\Delta$ nodes in $V'$ for which $R_u = 0$.

We next simulate the costs incurred in the different geometries we discussed. We choose $\Delta = 2$, for which the results for PRR trees and Chord rings are identical. We choose $D = \{2, 6\}$ for the $D$-dimensional tori, and $D = \log_\Delta N$ for the other geometries.

We vary the number of nodes between $N = 10$ and $N = 1,000$, and, for each value of $N$ run ten differently seeded Monte Carlo simulations, consisting of 100,000 requests each, with $X$ and $Y$ i.i.d. uniform random variables. We plot the latency and routing costs averaged over all nodes and all requests in Fig. 7.2.




(a) Latency cost ($l = 1$) (b) Routing cost ($r = 1{,}000$)

**Fig. 7.2** Latency and routing costs. Curves marked "sim" present simulation results. The full mesh, for which the latency cost is constantly equal to 1, and the routing cost is constantly equal to 0, is omitted for readability purposes

The graphs confirm that the star provides a lower average cost than all the other geometries, which is consistent with our earlier finding that the star is, in many cases, a social optimum. Note however, that our cost model does not take into account factors such as scalability and resiliency, both of which are cause for serious concerns in a completely centralized architecture. Additionally, while we have shown that the star network was potentially a Nash equilibrium, we nevertheless need incentive mechanisms (e.g. monetary rewards) to compensate for the asymmetry of a star network, and to convince a node to occupy the central position in the first place.

**Asymmetry in Item Popularity.** We investigate next how relaxing the assumption that all items have identical popularity impacts the results we have obtained so far. To that effect, we run a set of simulations, where items have a popularity that follows a Zipf-like distribution inspired by measurement studies such as [7].

In this set of Monte Carlo runs, we simulate a network of size $N = 512$ nodes. We select $D = 3$ for the $D$-torus, and $\Delta = 2$ and $D = 9$ for the other geometries. We run 1,024 trials, each consisting of 100,000 requests. The source of the request $X$ is a uniform random variable, and the requested item $Y$ is determined according to a Zipf-like distribution. That is, we have

$$S_u = s\Omega/(\mathrm{Rank}(u))^\alpha \ ,$$

with $\alpha = 0.75$, $\Omega = \left(\sum_{i=1}^{N} i^\alpha\right)^{-1}$, and $\mathrm{Rank} : V \to \{1, \ldots, N\}$, a bijective function that orders the nodes $u \in V$ by decreasing probability that a given item $k$ is held by $u$.

Because $Y$ is not a uniform random variable anymore, different nodes experience different latency and routing costs. In each experiment, we collect the ratios between the highest ($L_{\max}$ and $R_{\max}$) and lowest ($L_{\min}$ and $R'_{\min}$) latency and routing costs observed over all nodes. In de Bruijn graphs, because some nodes do not route any traffic, we use again $R'_{\min} = \min_{u \in V}\{R_u > 0\}$.

In Table 7.3, we present the average ratios $L_{\max}/L_{\min}$ and $R_{\max}/R'_{\min}$, averaged over all 1,024 experiments. Numbers in parentheses denote the corresponding standard deviation. The results indicate that, for all geometries, the latency costs of all nodes are relatively similar, but, the routing costs present significant differences. We explain the higher degree of asymmetry of the de Bruijn graph by the disparities in the node loadings (see Sect. 7.5), that magnify inequalities in routing costs. As a comparison to the social optima, we point out that in a star or a full mesh, the routing and latency costs are similar regardless of the popularity of the different items.

**Table 7.3** Asymmetry in costs in a network where item popularity follows a Zipf-like distribution

| | $\frac{L_{\max}}{L_{\min}}$ | $\frac{R_{\max}}{R'_{\min}}$ |
|---|---|---|
| 3-torus | 1.2675 ($\pm$0.0442) | 5.2845 ($\pm$0.3516) |
| De Bruijn | 1.2453 ($\pm$ 0.0265) | 30.7275 ($\pm$9.5970) |
| PRR tree | 1.2591 ($\pm$0.0420) | 9.2154 ($\pm$0.6590) |

We next determine whether asymmetries in routing costs compensate asymmetries in latency costs, or, more significantly, in service costs. To that effect, we compute the correlation coefficient (denoted as $\mathrm{Corr}(x,y)$ for two variables $x$ and $y$) between $R$ and $L$, $R$ and $S$, and $L$ and $S$, computed over the 512 nodes $\times$ 1,024 experiments= 524,288 data points available for the triplet $(R,L,S)$, and present our findings in Table 7.4. For all three geometries, Table 7.4 indicates that there is almost no correlation[3] between $S$ and $R$ or $L$. In other words, the service cost $S$ incurred by a node has almost no incidence on $R$ or $L$. The correlation between $R$ and $L$ is also very weak, which indicates that different nodes may have, in the end, completely different costs.

In other words, with all three routing geometries considered, an asymmetry in the popularity of the items can cause a significant disparity in the costs incurred by different nodes. The disparity in costs itself results in some nodes being overloaded, or at least having strong incentives to leave and re-join the network to get a "better spot." This result emphasizes the importance of efficient load-balancing primitives for CDNs based on protocols relying on any of these routing geometries.

[3] The correlation coefficient actually only tests for a linear correlation. Additional tests, such as the $\eta$-test (or correlation ratio) are generally required to confirm the lack of correlation between two variables. We omit these tests here, but point out that additional data (e.g. scatter plots) confirm the lack of correlation between the variables.

**Table 7.4** Correlation between routing, latency, and service costs in a network where item popularity follows a Zipf-like distribution

| | $\mathrm{Corr}(R,L)$ | $\mathrm{Corr}(R,S)$ | $\mathrm{Corr}(L,S)$ |
|---|---|---|---|
| 3-torus | −0.3133 | −0.0166 | −0.0960 |
| De Bruijn | −0.3299 | −0.0112 | −0.0981 |
| PRR tree | −0.2278 | −0.0128 | −0.1027 |

**Sparse Population of the Identifier Space.** So far, we have assumed that the identifier space is fully populated. For instance, a PRR tree with $\Delta = 2$ and $D = 9$ would necessarily contain $N = 512$ nodes. In practice however, the identifier space is likely to be relatively sparsely populated.

We run the following simulations. For each geometry, we consider a fixed number of nodes $N = 512$. We start with a fully populated identifier space, with $\Delta = 2$ and $D = 9$ for both de Bruijn graphs and PRR trees, and gradually increase $D$ up to $D = 15$. For the $D$-torus, we use $D = 3$, so that each node $u$ is represented by a set of coordinates $(u_x, u_y, u_z)$. We allow each coordinate to take integer values between 0 and $n$. Initially, we select $n = 8$, so that each possible set of coordinates corresponds

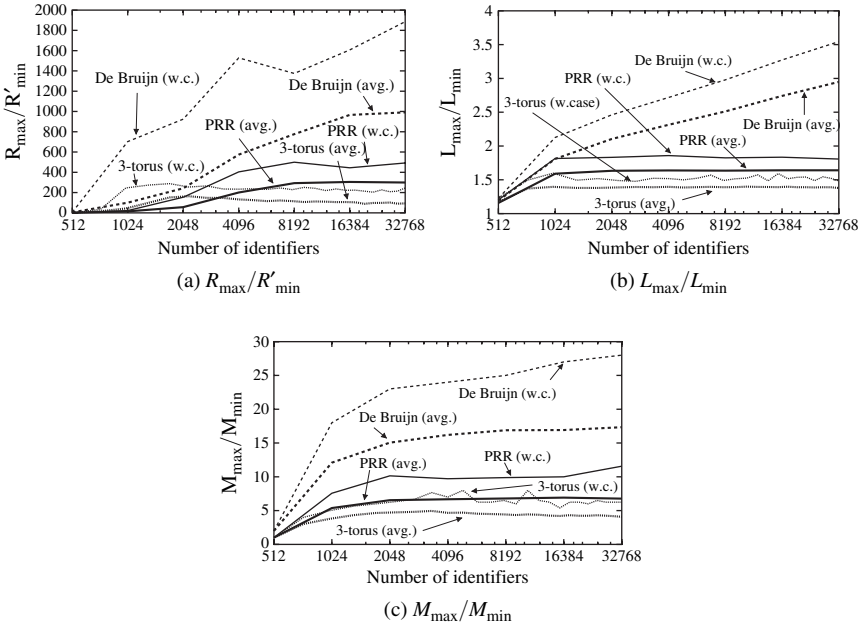



**Fig. 7.3** Ratios between maximum and minimum routing, latency, and maintenance costs experienced at a given node in function of the number of identifiers used. Curves marked "avg." indicate average results over all experiments in a given set, while curves marked "w.c." denote the maximum ratio, or worst case, observed over all experiments in a given set

to a given node (because $n^D = N$), and we then gradually increase $n$ up to $n = 32$. In other words, for all three topologies, we increase the identifier space from 512 to 32,768 identifiers. Identifiers that initially do not map to any node are selected using a uniform random variable. For each value of $D$ (resp. $n$) we run 100 experiments with different random seeds, corresponding to 100 different ways of populating the identifier space. Each experiment consists of 100,000 requests, where $X$ and $Y$ are i.i.d. uniform random variables.

We plot in Fig. 7.3, for each geometry, the average value of the ratios $R_{\max}/R'_{\min}$, $L_{\max}/L_{\min}$, and $M_{\max}/M'_{\min}$ averaged over the 100 experiments corresponding to a given number of identifiers, as well as their worst-case (i.e., maximum) value over the same 100 experiments.

We observe that the imbalance in latency costs remains relatively modest, with factors of about 3–4, in a sparsely populated identifier space. The imbalance in maintenance costs is more significant (ratios of 20–25 between the highest and lowest maintenance cost). Our main observation is that the imbalance in routing costs can become very large, with ratios between the worst-case and best-case routing costs *in the order of hundreds to thousands*. This observation emphasizes the urgent need for efficient load balancing routing algorithms, in absence of which a large number of nodes may have a strong incentive to leave the network and re-join immediately in hopes of obtaining a more favorable spot, thereby causing instability.

Furthermore, it can be shown that a sparsely populated identifier space has the effect of making the different costs correlated [11]. This confirms the intuition that the routing and latency costs of a given node are largely dependent on how well the node is connected to the rest of the network, which is expressed by the maintenance cost.

## 7.7 Visionary Thoughts for Practitioners

We have so far considered Nash equilibria to be a perfectly adequate tool for modeling rational behavior from participants in a CDN. The main advantage of the concept of Nash equilibrium resides in its simplicity. However, because Nash equilibria rely on very stringent assumptions on the capabilities and objectives of each player, they can predict counter-intuitive or unrealistic outcomes. In particular, rational players are expected to demonstrate error-free decision-making, to have perfect foresight of the game and to be unbounded in their computational abilities. Intuitively, players such as network users, which are not necessarily perfectly rational, or automated agents, which can be faulty, due to software bugs or misconfiguration, or have limited computational resources, will likely deviate from these rigid assumptions.

There is, therefore, ample motivation to formulate more generalized models of strategic behavior that include the notion of the Nash equilibrium as a special case. In particular, to relax the assumption of perfect rationality required by the concept of Nash equilibrium, some have introduced the concept of *bounded rationality*. Players that are bounded rational are not necessarily picking the best strategy available across the entire decision space, but instead are allowed to make small errors on a

number of levels, such as the evaluation of the payoffs associated with a strategy, the assessment of the best available strategy, or the execution of a specific strategy.

There are many different techniques to model bounded rationality. One way is to introduce (possibly small) amounts of noise into the decision-making process (see, for instance, [21]). Another model of equilibrium with bounded rationality, called Quantal Response Equilibrium [32], has been used to characterize equilibria in games where users make errors on the computation of the payoffs associated with a given strategy.

Perhaps the simplest relaxation to consider is that of *near rationality* [4, 41], exemplified for instance by the $\varepsilon$-equilibrium concept [41]. The $\varepsilon$-equilibrium is relaxing the conception of a fully rational player to a model where each player is satisfied to get close to (but does not necessarily achieve) her best response to the other player's strategies. No player can increase her utility by more than $\varepsilon > 0$ by choosing another strategy. Therefore, we locate an $\varepsilon$-equilibrium by identifying a strategy for each player so that her payoff is within $\varepsilon$ of the maximum possible payoff given the other players' strategies.

**Definition 4.** A vector of mixed strategies $\sigma^{\varepsilon} = (\sigma_1^{\varepsilon}, ..., \sigma_N^{\varepsilon}) \in \Sigma$ comprises a mixed-strategy $\varepsilon$-equilibrium of a game G if, for all $i \in N$, for all $\sigma_i \in \Sigma_i$, and a fixed $\varepsilon > 0$, $u_i(\sigma_i, \sigma_{-i}^{\varepsilon}) - u_i(\sigma_i^{\varepsilon}, \sigma_{-i}^{\varepsilon}) \leq \varepsilon$.

A pure-strategy $\varepsilon$-equilibrium is a vector of pure strategies, $\zeta^{\varepsilon} \in Z$, that satisfies the equivalent condition. If we allow $\varepsilon = 0$ this condition reduces to the special case of a Nash equilibrium. Thus, one can consider $\varepsilon$-equilibria as a more generalized solution concept for competitive equilibria.

**Revisiting Overlay Network Formation.** Recall that, considering only Nash equilibria, we have shown in Propositions 4 and 5, that, depending on the relative values of the different parameters $m$, $l$, and $N$, the star network or the full mesh were the most likely candidates to be Nash equilibria. However, if instead of considering Nash equilibrium, we now consider an $\varepsilon$-equilibrium, then, for any $m \in [l/N - \varepsilon, l/N + \varepsilon]$, *any* network topology constitutes an $\varepsilon$-equilibrium. This can be proven by simply including $\varepsilon$ in the proofs of Propositions 4 and 5. Additionally, if, to account for failures in link establishment due for instance to lossy channels, we allow nodes to use mixed strategies instead of being restricted to pure strategies, we conjecture that the range of possible values for $m$ such that any network is an $\varepsilon$-equilibrium is much larger than $2\varepsilon$.

**Practical Relevance.** Given the uncertainty on the equilibrium concept itself, what benefits could a practitioner take away from applying game-theory to CDN design?

First, irrespective of the equilibrium concept considered, social optimum analysis is crucial to understanding the upper bounds on the network's performance. In particular, Papadimitriou defined an interesting performance metric he termed the "price of anarchy [37]." The price of anarchy is the ratio of the aggregate utility $\sum_u C_u$ obtained in the worst-case Nash equilibrium over that of the social optimum.

The price of anarchy is a useful metric in system design, as it allows to show how far performance can degrade if players are left to their own devices.

Along the same lines, Nash equilibria, are, despite their stringent assumptions, a very useful tool as a first-order approximation of individual incentives. In particular, the power assumed of the players in a Nash equilibria can lead to characterize a worst-case scenario, which allows the designer to obtain lower bounds on the network performance. As an illustration of the Nash equilibrium being a possible worst-case equilibrium, let us consider TCP congestion control. It has been shown that, if a Nash equilibrium were reached among competing players vying for throughput in a TCP/IP network, everybody would turn off congestion control (or use UDP) [3]. Reality is markedly different, in that most users are happy with leaving their TCP parameters unchanged, which can be explained by bounded rationality arguments [12].

Third, considering near rationality instead of perfect rationality can help evaluate the accuracy of a game-theoretic model. If the model seems to lack robustness, its chances of being an accurate model of reality decrease. In the above example, we see that the parameter space that yields uncertain results grows linearly with the uncertainty $\varepsilon$, which shows the model is robust enough, and presumably reliable. On the other hand, other examples, outside of network formation, given in [12] do not exhibit the same desirable properties.

## 7.8 Future Research Directions

The present chapter has described applications of game theory to model interactions in overlay networks, such as CDNs. Economics-informed network design is however not limited to game theory, but is, more generally, the study of individual incentives in networks. As such, this research field is expected to remain vibrant for the foreseeable future.

In the near term, we expect research contributions to help develop more sophisticated models of incentives in content delivery. In fact, it is notable that more and more recent system design papers, e.g. [14, 26] among many others, rest on economic principles to architect their proposals.

More generally it seems almost inevitable that, in light of a booming market for content distribution, fierce competition among infrastructure providers, and potentially slim margins, CDN designers will have to better understand market forces and individual incentives to ensure profitability.

Among other potential directions for future research, we can cite the study of incentives to enhance CDN security. For instance, [29] shows that, by refactoring existing technology to realign diverging incentives, one can create an overlay network that offers potentially competing content providers resilience to Distributed Denial of Service (DDoS) attacks.

Such research denotes an interesting trend that turns the original problem on its head. Instead of trying to fix incentive alignment problems, future system research

may very well *exploit* differences in incentives to achieve superior system design, by having different entities serve different roles best suited to their own aspirations.

## 7.9 Conclusion

We proposed a model, based on experienced load and node connectivity, for the cost incurred by each node participating in an overlay network such as a CDN. We argue such a cost model is a useful complement to topological performance metrics [22, 31], in that it allows to predict disincentives to collaborate (nodes refusing to serve requests to reduce their cost), discover possible network instabilities (nodes leaving and re-joining in hopes of lowering their cost), identify hot spots (nodes with high routing load), and characterize the efficiency of a network as a whole.

One of the key insights is that inefficiencies may occur when nodes value the resources they use to forward traffic on behalf of other nodes. In such cases letting nodes choose which links they wish to maintain can yield a sub-optimal network with respect to overall resource usage.

Further, our analysis shows that designing very efficient load-balancing primitives is a must to avoid favoring some nodes at the expense of others, which can potentially create network instability. A possible alternative to load balancing primitives lies in incentive mechanisms that make it desirable for nodes to forward as much traffic as possible. The game-theoretic formulation proposed here lends itself to incentive-compatible design.

Finally, we believe that the framework described can be useful for a CDN designer in determining which type of topology is more appropriate for a specific context, e.g. content-delivery over ad-hoc networks, on-demand video broadcasting. The exercise indeed then becomes a parametrization effort to try to assess the different nominal costs relative to each other.

**Acknowledgements** Some of the materials presented in this chapter appeared in a preliminary form at IPTPS'04, INFOCOM'05, and PINS'04 [10, 11, 12]. This research was mostly conducted while Nicolas Christin was with the University of California at Berkeley, School of Information. This work was supported in part by the National Science Foundation through grants ANI-0085879 and ANI-0331659. The material presented in this chapter greatly benefited from discussions with Paul Laskowski.

## References

1. Napster protocol specification (2000). `http://opennap.sourceforge.net/napster.txt`
2. Adar, E., Huberman, B.: Free riding on Gnutella. First Monday **5**(10) (2000)
3. Akella, A., Seshan, S., Karp, R., Shenker, S., Papadimitriou, C.: Selfish behavior and stability of the Internet: A game-theoretic analysis of TCP. In: Proc. ACM SIGCOMM'02, pp. 117–130. Pittsburgh, PA (2002)

4. Akerlof, G., Yellen, J.: Can small deviations from rationality make significant differences to economic equilibria? American Economic Review **75**(4), 708–720 (1985)
5. Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable application layer multicast. In: Proc. ACM SIGCOMM'02, pp. 205–217. Pittsburgh, PA (2002)
6. Braess, D.: Über ein Paradoxon aus der Verkehrsplanung. Unternehmensforschung **12**, 258–268 (1969)
7. Breslau, L., Cao, P., Fan, L., Philips, G., Shenker, S.: Web caching and Zipf-like distributions: Evidence and implications. In: Proc. IEEE INFOCOM'99, pp. 126–134. New York, NY (1999)
8. Buchegger, S., Le Boudec, J.Y.: Performance analysis of the confidant protocol. In: Proc. ACM MobiHoc'02, pp. 226–236. ACM (2002)
9. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.Y., Moon, S.: I Tube, You Tube, Everybody Tubes: Analyzing the world's largest user generated content video system. In: Proc. ACM IMC'07. San Diego, CA (2007)
10. Christin, N., Chuang, J.: On the cost of participating in a peer-to-peer network. In: Proc. IPTPS'04, Lecture Notes in Computer Science, Vol. 3279, pp. 22–32. San Diego, CA (2004)
11. Christin, N., Chuang, J.: A cost-based analysis of overlay routing geometries. In: Proc. INFOCOM'05, Vol. 4, pp. 2566–2577. Miami, FL (2005)
12. Christin, N., Grossklags, J., Chuang, J.: Near rationality and competitive equilibria in networked systems. In: Proc. ACM SIGCOMM'04 Workshop on Practice and Theory of Incentives in Networked Systems (PINS), pp. 213–219. Portland, OR (2004)
13. Chu, Y.H., Rao, S., Zhang, H.: A case for endsystem multicast. In: Proc. ACM SIGMETRICS'00, pp. 1–12. Santa Clara, CA (2000)
14. Chun, B.G., Chaudhuri, K., Wee, H., Barreno, M., Papadimitriou, C., Kubiatowicz, J.: Selfish caching in distributed systems: a game-theoretic analysis. In: Proc. ACM PODC'04, pp. 21–30. Saint John's, NL, CA (2004)
15. Chun, B.G., Fonseca, R., Stoica, I., Kubiatowicz, J.: Characterizing selfishly constructed overlay networks. In: Proc. IEEE INFOCOM'04, Vol. 2, pp. 1329–1339. Hong Kong (2004)
16. Cohen, B.: Incentives build robustness in BitTorrent. In: Proc. 1st Workshop on the Economics of Peer-to-Peer Systems. Berkeley, CA (2003)
17. Fabrikant, A., Luthra, A., Maneva, E., Papadimitriou, C., Shenker, S.: On a network creation game. In: Proc. ACM PODC'03, pp. 347–351. Boston, MA (2003)
18. Feigenbaum, J., Shenker, S.: Distributed algorithmic mechanism design: Recent results and future directions. In: Proc. DIAL-M'02, pp. 1–13. Atlanta, GA (2002)
19. Feldman, M., Chuang, J., Stoica, I., Shenker, S.: Hidden-action in multi-hop routing. In: Proc. ACM EC'05, pp. 117–126. ACM (2005)
20. Goemans, M., Li, L., Mirrokni, V., Thottan, M.: Market sharing games applied to content distribution in ad-hoc networks. In: Proc. ACM MobiHoc '04, pp. 55–66. ACM, Roppongi Hills, Tokyo, Japan (2004)
21. Goeree, J., Holt, C.: A model of noisy introspection. Games and Economic Behavior **46**(2), 365–382 (2004)
22. Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of DHT routing geometry on resilience and proximity. In: Proc. ACM SIGCOMM'03, pp. 381–394. Karlsruhe, Germany (2003)
23. Hardin, G.: The tragedy of the commons. Science **162**(3859), 1243–1248 (1968)
24. Holt, C., Roth, A.: The Nash equilibrium: a perspective. Proc. National Academy of Sciences **101**(12), 3999–4002 (2004)
25. Hsieh, H.Y., Sivakumar, R.: Performance comparison of cellular and multi-hop wireless networks: A quantitative study. In: Proc. ACM SIGMETRICS'01, pp. 113–122. Cambridge, MA (2001)
26. Huang, C., Li, J., Ross, K.: Can internet video-on-demand be profitable? In: Proc. ACM SIGCOMM'07, pp. 133–144. Kyoto, Japan (2007)
27. Jackson, M., Wolinsky, A.: A strategic model for social and economic networks. Journal of Economic Theory **71**(1), 44–74 (1996)

28. Kaashoek, M.F., Karger, D.: Koorde: A simple degree-optimal distributed hash table. In: Proc. IPTPS'03, pp. 323–336. Berkeley, CA (2003)
29. Khor, S.H., Christin, N., Wong, T., Nakao, A.: Power to the people: Securing the Internet one edge at a time. In: Proc. ACM SIGCOMM'07 Workshop on Large-Scale Attack Defense (LSAD), pp. 89–96. Kyoto, Japan (2007)
30. Liebeherr, J., Nahas, M., Si, W.: Application-layer multicast with Delaunay triangulations. IEEE Journal of Selected Areas in Communications **20**(8), 1472–1488 (2002)
31. Loguinov, D., Kumar, A., Rai, V., Ganesh, S.: Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In: Proc. ACM SIGCOMM'03, pp. 395–406. Karlsruhe, Germany (2003)
32. McKelvey, R., Palfrey, T.: Quantal response equilibria for normal form games. Games and Economic Behavior **10**(1), 6–38 (1995)
33. Mockapetris, P., Dunlap, K.: Development of the domain name system. In: Proc. ACM SIGCOMM'88, pp. 123–133. Stanford, California (1988)
34. Naor, M., Wieder, U.: Novel architectures for P2P applications: the continuous-discrete approach. In: Proc. ACM SPAA'03, pp. 50–59. San Diego, CA (2003)
35. Nash, J.: Non-cooperative games. Annals of Mathematics **54**(2), 286–295 (1951)
36. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.): Algorithmic Game Theory. Cambridge University Press (2007)
37. Papadimitriou, C.: Algorithms, games and the Internet. In: Proc. ACM STOC'01, pp. 749–753. Heraklion, Crete, Greece (2001)
38. Pathan, A.M., Buyya, R.: Economy-based content replication for peering content delivery networks. In: Proc. CCGRID, pp. 887–892. IEEE Computer Society (2007)
39. Perkins, C. (ed.): Ad hoc networking. Addison-Wesley, Boston, MA (2000)
40. Plaxton, C.G., Rajamaran, R., Richa, A.: Accessing nearby copies of replicated objects in a distributed environment. Theory of Computing Systems **32**(3), 241–280 (1999)
41. Radner, R.: Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. Journal of Economic Theory **22**, 136–154 (1980)
42. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proc. ACM SIGCOMM'01, pp. 161–172. San Diego, CA (2001)
43. Roughgarden, T., Tardos, É.: How bad is selfish routing? Journal of the ACM **49**(2), 236–259 (2002)
44. Rowston, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large scale peer-to-peer systems. In: Proc. IFIP/ACM Middleware'01, pp. 329–350. Heidelberg, Germany (2001)
45. Saroiu, S., Gummadi, K., Gribble, S.: A measurement study of peer-to-peer file sharing systems. In: Proc. SPIE/ACM MMCN'02, pp. 156–170. San Jose, CA (2002)
46. Shenker, S.: Making greed work in networks: A game-theoretic analysis of switch service disciplines. IEEE/ACM Transactions on Networking **3**(6), 819–831 (1995)
47. Sivarajan, K., Ramaswami, R.: Lightwave networks based on de Bruijn graphs. IEEE/ACM Transactions on Networking **2**(1), 70–79 (1994)
48. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup protocol for Internet applications. IEEE/ACM Transactions on Networking **11**(1), 17–32 (2003)
49. Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A., Kubiatowicz, J.: Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications **22**(1), 41–53 (2004)

# Chapter 8
# CDN Pricing

Kartik Hosanagar

## 8.1 Introduction

Content Delivery Networks (CDNs) are a very important part of the content delivery supply chain. The supply chain consists of content providers that create content, backbone, and access networks that help transport the content; and CDN providers that store and deliver the content to end users from the edges of the network. By co-locating their servers close to the edge of the network, CDNs are uniquely positioned within this value chain and are thus used by a large number of the major content providers on the Internet. With increasing broadband adoption and a marked shift towards multimedia content delivery, CDNs will likely play an increasingly important role in media delivery over the Internet.

CDNs provide significant economic benefits to content providers. By delivering content from the edges of the network, they help in speeding content delivery. Further, they help achieve economies of scale in infrastructure costs by aggregating traffic across multiple customer sites. As a result, a CDN's marginal cost of delivering content is much lower than that incurred by a small content provider that chooses to deliver content on its own. Aggregation of traffic also reduces the impact of variability in demand for content. Since the traffic of different content providers are unlikely to surge at the same time, a sudden surge in demand for one content provider is handled by increasing the fraction of the CDN's infrastructure being consumed by that content provider. Finally, since there are several candidate servers from which the CDN can serve content, no single point will be a bottleneck. This helps improve the availability of content, especially during flash crowds or Denial of Service (DoS) attacks.

At the same time, CDNs incur costs in maintaining the infrastructure and in delivering the content. Due to the costs incurred by the CDN and the significant value realized by content providers, a market mechanism is needed to ensure that content providers can continue to derive value and CDN providers simultaneously have

Kartik Hosanagar
Operations and Information Management, The Wharton School, University of Pennsylvania,
e-mail: kartikh@wharton.upenn.edu

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

incentives to deploy and manage the infrastructure. The simplest such mechanism is to price the service. The issue of CDN pricing is the focus of this chapter.

CDN pricing has received considerable attention in the business domain. The pricing policies significantly impact the margins of the CDN providers and the surplus of subscribing content providers. Some of the issues that have generated interest include the impact of bursty traffic on pricing and profitability [7], the use of volume discounts and overages in current pricing plans [18] and the impact of competition and pricing wars on the CDN market [15].

The remainder of the chapter is organized as follows. We begin by providing an overview of common pricing models employed in the industry. We then discuss academic work on the economics of the content delivery market including prior work on CDN pricing. Section 8.3 provides a model to capture content providers' value from CDN services and uses that to discuss pricing policies. We then present a discussion for industry practitioners. Finally, we discuss opportunities for future research and conclude this chapter.

## 8.2 Common Pricing Models in the Industry

A number of different pricing models are used by CDNs. All of them have a usage component. We briefly discuss the two most popular pricing models below.

### *8.2.1 Pricing Based on Aggregate Usage*

The simplest pricing structure involves a content provider committing to a certain level of usage (e.g. 50 TBs/month). Based on the commitment level, the CDN determines a price per GB delivered. These pricing plans usually offer a volume discount (e.g. $0.5/GB for a traffic commitment of 40–50TBs and $0.15/GB for commitment of over 100TBs). Some CDNs impose a penalty in case usage exceeds the monthly commitment in an attempt to induce the content provider to provide more accurate monthly commitment levels which in turn can help the CDN in better capacity planning. However, the general pricing structure involves volume discounts for content providers with significantly high traffic volume.

### *8.2.2 Percentile-Based Pricing*

Another popular alternative involves pricing based on the 95th percentile of traffic. In this policy, the CDN periodically samples the bandwidth usage of a subscribing content provider. It then computes the 95th percentile of usage at the end of the month and charges a price per Mbps based on the 95th percentile of usage.

Most CDNs offer both pricing plans and content providers can select the preferred plan. Later in this chapter, we discuss the profit implications of these pricing schemes.

## 8.3 Background and Related Work

There are two streams of work that are closely related to CDN pricing. The first relates to work on pricing in networks with congestion. The second stream relates to work on the economics of content delivery including a few research efforts on CDN pricing.

### *8.3.1 Congestion Pricing in Networks*

Work on congestion pricing in networks usually focuses on the interaction between pricing and Quality of Service (QoS) by studying the trade-off between congestion cost and capacity cost. When capacity cannot be easily increased and QoS requirements of applications are stringent, pricing plays a key role in achieving desired QoS. Specifically, an increase in the price encourages users to shape their traffic and control the demand for network services which in turn reduces congestion in the network. Mendelson [16] and Mendelson and Whang [17] present models for pricing computer services in the presence of delay costs. Other relevant work includes pricing of congestible resources on the Internet [5, 14] and QoS pricing in prioritized transmission of data packets based on QoS schemes such as Diffserv and Intserv [2, 6].

The literature has also studied pricing under bursty traffic. Kelly proposes that prices under bursty traffic can be based on effective bandwidth [13]. However, the effective bandwidth depends on the characteristics of the multiplexed traffic and the link resources and cannot be easily predicted. As a result, Kelly proposes an approximation that includes a charge per unit time, a charge per unit volume of traffic carried and a charge per connection [13].

In the CDN context, pricing is not driven by the goal of reducing congestion within the CDN's network. Content providers do not want to drop client requests because some traffic profile limit has been reached. For instance, flash crowds cannot be predicted in advance, and content providers subscribe to CDN services precisely to manage the traffic spikes. The CDN's very appeal lies in that it has sufficient capacity to manage peak traffic and that traffic across its subscribers are not highly correlated so that a surge in one subscriber's traffic is easily addressed by increasing the fraction of the CDN's capacity allocated to that subscriber. As a result, congestion reduction is not a key goal of pricing. Thus, the results from the research stream on congestion pricing, while relevant, do not readily transfer to the CDN domain.

### 8.3.2 Economics of Content Distribution

The history of the CDN market can be traced back to the proxy caches used by retail ISPs to store and deliver content from the edges of the network. Hosanagar et al. [8, 9] study the economics of web caching and find that adoption of traditional best-effort caching will decrease as content providers move towards dynamic content and simultaneously seek accurate business intelligence regarding website usage. The authors report that CDN services can play an important role intermediating between content providers that seek the benefits of edge delivery and retail ISPs that have the ability to install servers at the edge of the network. Viewed in this manner, CDNs allow content providers to reap the benefits of edge delivery of content without incurring the costs of best effort caching.

Kaya et al. [12] conduct an economic and operational analysis of the content delivery market. In terms of CDN pricing, Ercetin et al. [4] study optimal pricing and resource allocation for a differentiated services CDN. Hosanagar et al. [7, 10] study the optimal pricing for a monopoly CDN. They find that traditional usage-based pricing plans should entail volume discounts when subscribing content providers have similar levels of traffic burstiness but that volume discounts can prove suboptimal when traffic burstiness is highly heterogeneous. Further, the authors find that profitability from a percentile-based pricing plan can be substantially higher than traditional usage based billing. In the following section, we discuss the models and results from these research efforts on CDN pricing in greater detail.

Other than the above related work, there exists other models for edge delivery of content including Peer-to-Peer (P2P)-based content delivery. Courcobetis et al. [3] discuss market models for P2P-based content distribution as an alternative to CDN-based content delivery. Johar et al. [11] explore the impact of P2P networks on the CDN market. The authors find that P2P networks can sometimes benefit CDN firms by encouraging content providers to seek high quality CDN services as a means to compete with illegal delivery of their media in P2P networks.

## 8.4 Models for CDN Pricing

In this section, we develop the models for CDN pricing to capture the content providers' value from CDN services. We then use the models to discuss pricing policies. The models are based on those in [10].

Consider a market with a monopoly CDN. Content providers have the option of delivering content on their own (from their own servers or those of a hosting service) or outsourcing content delivery to a CDN. The differences between the content providers are specified in terms of the differences in their mean traffic level $\lambda$ and their outsourcing cost $C_o$. The mean arrival rate, $\lambda$, is a measure of the volume of traffic handled by the content provider. Prior empirical studies [9] suggest that the number of content providers with mean arrival rate $\lambda$ is given by $g(\lambda) = \beta/\lambda^{\delta}$, where $\delta \in [1, 2]$ and $\beta$ are constants. $C_o$, the cost of outsourcing content delivery

includes the cost of modifying content to facilitate delivery by the CDN, or the cost of sharing confidential data with a third party (i.e. CDN provider). We assume that the cdf and pdf of $C_o$ are given by $H(C_o) = {C_o}^W; h(C_o) = W{C_o}^{W-1}$, where $W$ is a positive constant and $C_o \in [0,1]$. The parameter $W$ allows us to vary the relative density of content providers with low outsourcing cost. When $W = 1$, $C_o$ is Uniformly distributed. $W > 1$ captures negative skews and $W < 1$ captures positive skews in the distribution. The upper bound of $C_o$ can be arbitrarily high but is merely normalized to 1 without loss of generality. Typically, the CDN knows $\lambda$ for all content providers as the traffic can be directly observed by the CDN, but does not know the outsourcing cost $C_o$. We assume that the CDN knows the distribution of outsourcing costs across content providers $(H(C_o))$.

To determine the CDN's optimal pricing policy, we proceed as follows. First, we determine a content provider's expected surplus from self provisioning and that from delivering content through the CDN. The content provider chooses the option that generates the higher expected surplus. This subscription decision is a function of the CDN's pricing policy. Based on the content provider's subscription decision, we determine the CDN's optimal pricing policy in order to maximize the CDN's expected profit. We now discuss the content provider's surplus under self-provisioning and provisioning through a CDN.

### *8.4.1 Self-Provisioning by Content Provider*

Consider a content provider (denoted as CP) with mean traffic $\lambda$ delivering content to users. Let $X$ be a random variable denoting the realized number of requests to the content provider in a given period. In any period, the distribution of $X$ is known a priori but the realized value of $X$ is unknown. The content provider can choose to deliver this content directly by investing in infrastructure to process a mean of $I$ requests per unit time. If it does so, its surplus from serving content is

$$U_{self}(X) = V(X) - C(I) - c \cdot L(I,X) \tag{8.1}$$

where $V()$ is the content provider's benefit from responding to all $X$ requests, $C()$ is the cost of maintaining the infrastructure (servers, bandwidth, software, etc.), which is concave in $I$ because of economies of scale. $L()$ is the number of lost requests which increases with $X$ but decreases with $I$, and $c$ is the cost of each lost request. $V()$ includes all sources of revenue from the content provider's Internet operations (e.g. selling products on the Internet). We model the CP's infrastructure cost as: $C(I) = a_1 \cdot I - a_2 \cdot I^2$ for $\forall I \leq a_1/2a_2$, which captures the concavity between $I$ and cost. The constraint $I \leq a_1/2a_2$ ensures that the infrastructure cost is always non-decreasing in infrastructure (note that $C'(I) < 0$ for $I > a_1/2a_2$). In this formulation, a large value for $a_1$ indicates high infrastructure costs and a large value for $a_2$ indicates significant economies of scale. $L()$, the number of lost requests, increases with $X$ but decreases with $I$.

The content provider's expected surplus from delivering content is obtained from (1) as follows:

$$U_{self} = E[U_{self}(X)] = V - C(I) - c \cdot L(I) \tag{8.2}$$

where $L(I) = E[L(I,X)]$ and $V = E[V(X)]$. The content provider chooses an infrastructure level in order to maximize the expected surplus. The content provider's decision problem is $\max_I \{U_{self}(I)\}$. We denote the optimal infrastructure level as $I^*$ and associated expected surplus as $U_{self}(I^*)$.

### 8.4.2 Provisioning Through a CDN

The other alternative available to a content provider is to deliver content through a CDN. The content provider's surplus from delivering content through the CDN is

$$U_{CDN}(X) = V(X) + \tau(N) \cdot X - C_O - P(X) \tag{8.3}$$

where $V()$, and $X$ are defined as above, $\tau()$ is the benefit per request from faster content delivery through a set of $N$ CDN servers, $C_o$ is the cost of outsourcing content delivery, and $P()$ is the usage-based price the CDN charges the content provider.

The outsourcing costs incurred by content providers may be in the form of content modification costs or the cost of sharing confidential data with the CDN. The former is the cost associated with modifying content in order to facilitate delivery by the CDN. The cost of sharing confidential data arises because the content provider may be sharing sensitive information such as customer records, credit card information or patient medical history with the CDN. The cost may be in the form of perceived risk or may be due to additional steps needed to ensure security. The cost of sharing confidential data is expected to vary across content providers because of inherent differences in the type of content handled by content providers.

The CP does not know how many requests ($X$) will be made for its content in any period, but can compute the expected surplus from using the CDN:

$$U_{CDN} = E[U_{CDN}(X)] = V + \tau(N)\lambda - C_o - E[P(X)] \tag{8.4}$$

Given any price function $P(X)$, the CP can compute the expected surplus. The CP chooses the CDN if $U_{CDN} \geq U_{self}(I^*)$. Substituting Eqs. (8.2) and (8.4) into this condition, a CP with mean traffic $\lambda$ and outsourcing cost $C_o$ subscribes to the CDN if

$$C_o \leq \tau(N)\lambda + C(I^*) + c \cdot L(I^*) - E[P(X)] \tag{8.5}$$

### 8.4.3 CDN's Profit Function

The content providers choose to either self-provision or deliver content through the CDN depending upon which option provides the highest expected surplus. Recollect

that the CDN does not know the outsourcing cost for any individual CP. So for a given choice of $P(X)$, it cannot determine whether a specific CP will subscribe. However, the CDN knows the distribution of $C_o$ across CPs. Thus it can compute the probability that a CP subscribes to the service. With $H(C_o) = {C_o}^W$, the probability that a CP with mean traffic $\lambda$ subscribes to a CDN is

$$\Pr(\text{Subscribe}|\lambda) = \left(\tau(N)\lambda + C(I^*) + c \cdot L(I^*) - E[P(X)]\right)^W \tag{8.6}$$

If $g(\lambda)$ denotes the number of CPs with mean arrival rate $\lambda$, then the expected number of these CPs subscribing to the CDN is given by

$$Subs(\lambda) = g(\lambda)\left(\tau(N)\lambda + C(I^*) + c \cdot L(I^*) - E[P(X)]\right)^W \tag{8.7}$$

Any subscribing CP pays $P(X)$ for a realized level of requests $X$. Since $X$ is not known a priori, the CDN does not know a priori its realized profit from a price function $P(X)$. However, the CDN's expected profit can be computed as follows:

$$\pi = \left\{ \int_\lambda Subs(\lambda) \left( \int_X \Pr(X|\lambda) \cdot P(X) dX \right) d\lambda \right\} - \left\{ b_1 \left( \int_\lambda \lambda \cdot Subs(\lambda) d\lambda \right) - b_2 \left( \int_\lambda \lambda \cdot Subs(\lambda) d\lambda \right)^2 \right\} \tag{8.8}$$

In the expression above, the first term represents the CDN's expected revenues. That is, $\Pr(X|\lambda)$ denotes the probability that a content provider with mean traffic $\lambda$ gets $X$ requests. Thus, $\int_X \Pr(X|\lambda) \cdot P(X) dX$ denotes the expected revenues from one content provider with mean traffic $\lambda$. The CDN's total expected revenues are obtained by summing the above expression over all content providers. The second term represents the CDN's cost, which is modeled to be quadratic over the mean volume of traffic handled by the CDN (given by $\int_\lambda \lambda \cdot Subs(\lambda) d\lambda$). This cost includes the cost of keeping content consistent across replicas, an accounting mechanism that collects and tracks information related to request routing and delivery [20], and the cost associated with content delivery. All these activities are expected to involve economies of scale and thus the costs are expected to be concave in volume. Concavity is captured here using a quadratic cost function. Note that the CP and CDN cost parameters are different (i.e. $a_1 \neq b_1$, $a_2 \neq b_2$) because the CDN cost includes other factors, such as accounting cost and cost of maintaining consistency, in addition to the content delivery cost.

The CDN's decision problem is to choose a price function $P(X)$ in order to maximize its expected profit.

### 8.4.4 Optimal Pricing for Poisson and Bursty Traffic

We now investigate the CDN's optimal pricing policy using simulations. We consider a population of 1000 content providers. The mean arrival rates for the CPs are drawn from a Pareto distribution in [1000,8000]. Given the mean arrival rates, the traffic is drawn from the distributions specified below:

(a) Poisson: In this case, all 1000 content providers have Poisson distributed traffic.
(b) Bursty Traffic: In this case, we assume that all CPs have bursty traffic. We model bursty traffic through a a Markov Modulated Poisson Process (MMPP). MMPP is commonly used to model bursty traffic to web servers [1, 19]. MMPP is a doubly stochastic Poisson process in which the arrival rate is given by an $m$-state Markov process. When the Markov chain is in state $j$, arrivals follow a Poisson process with arrival rate $\lambda_j$. We consider a simple 2-state MMPP with arrival rates $\lambda_1$ and $\lambda_2$. The limiting state probabilities of the phase process are $q = (q_1, q_2)$ The mean and variance of the number of requests in a unit time period are denoted $\bar{\lambda}$ and $\Psi$ respectively. A burst in traffic is modeled by assuming a very large value of $\lambda_2$ along with a non-zero probability of transitioning to state 2. We set $\lambda_2 = 10\lambda_1$ and ($q_1 = 0.9,\ q_2 = 0.1$) as our MMPP parameters in this section. In other words, the mean arrival rate during bursts is ten times the regular mean arrival rate; and the system, on average, bursts 10 % of the time. Different values of $\bar{\lambda}$ are simulated by varying $\lambda_1$. Further, when the mean arrival rate $\bar{\lambda}$ is increased, we adjust the state transition probabilities to maintain constant burstiness (constant value for $\sqrt{\Psi}/\bar{\lambda}$).
(c) Mixed Traffic: 500 CPs have Poisson traffic and 500 CPs have MMPP traffic.

The remaining simulation parameters are as follows. For the infrastructure cost function, $C(I) = a_1 \cdot I - a_2 \cdot I^2$, we assume that $a_1 = 3.56$ and $a_2 = 0.000043$. These values are roughly comparable to current infrastructure costs. For example, under these parameter values the cost of serving 233 requests/min is $804 per month. If we assume that the average size of the response to a request is 100 KB, this implies that the cost of serving data at 3.10 Mbps is $804 per month. This is reasonable given the cost of a fractional T3 connection and of maintaining a low-end server. Likewise, the cost of serving 6,975 requests per minute is $22,042, which is approximately the cost of a T3 connection and the associated cost of maintaining a server. These costs are also comparable to managed hosting costs at the time of writing. We assume that the cost of a lost request, $c$, is $10. This is based on an assumption that 10% of visitors purchase products/services, the average purchase is $100, and a customer leaves a website if a request does not go through. Finally, CP outsourcing cost is drawn from a U[0,30000] distribution.

Under these settings, we compute the optimal infrastructure and associated expected surplus under self-provisioning for each CP. Next, we compute the CP's expected surplus from CDN-provisioning for a given CDN price function. For the CDN price function, we restrict attention to quadratic functions specified by $P(X) = p_0 \cdot X \pm p_1 \cdot X^2$, and perform a grid search for optimal values of $p_0$ and $p_1$. Note that the above price function allows us to model both concave and convex price

functions. For each CP, we draw 1000 values of $X$ from the corresponding arrival distribution (Poisson or MMPP). Given $p_0$ and $p_1$, we can compute the price $P(X)$ corresponding to each value of $X$ and also the expected price for the CP by averaging over the 1000 values of $X$. Given these parameters, the expected surplus from using the CDN is computed. The CP subscribes to the CDN if the expected surplus is higher than under self-provisioning. The CDN's expected profit is obtained by summing expected revenues from each subscribing CP and subtracting the CDN's cost as in (8) with $b_1 = 3$ and $b_2 = 0.000047$. We compute the optimal price in 50 replications of the simulation. We present the main results below.

#### 8.4.4.1 Optimal Pricing Under Poisson Traffic Entails Volume Discounts

When content providers have Poisson traffic, the CDN's optimal price function $P^*(X)$ is $3.9X - 6.6e-05X^2$. Thus, the optimal pricing policy under Poisson traffic entails volume discounts. The concavity in the optimal price holds even as we change the parameters of the simulation. Further, the magnitude of the volume discount increases with an increase in the economies of scale in the content provider's infrastructure costs (increase in $a_2$).

#### 8.4.4.2 Optimal Prices are Higher Under Bursty Traffic

The optimal prices for the three traffic types are plotted in Fig. 8.1. It can be seen that the CDN is able to charge higher prices as traffic burstiness increases. Specifically, $Price(MMPP) > Price(Mixed) > Price(Poisson)$. This is because the CDN's value proposition to CPs in terms of avoiding lost requests is enhanced in the presence of bursty traffic. As a result, there is a marked increase in the price charged by the CDN. Interestingly, the optimal price under mixed traffic is convex and involves a volume tax rather than a volume discount. The rationale behind this is described below.

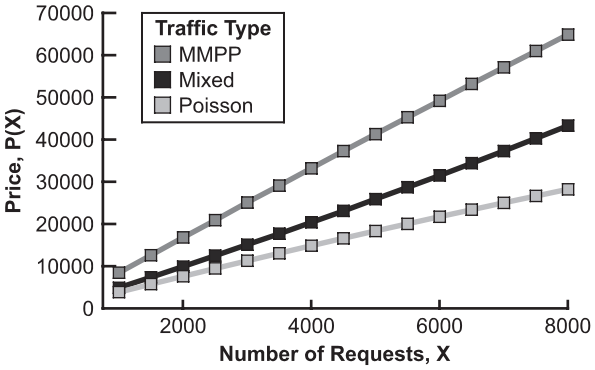



**Fig. 8.1** Optimal price functions for the three cases

#### 8.4.4.3 Traditional Usage-Based Pricing is Inefficient Under Bursty Traffic

Consider the pricing scheme with volume discounts shown in Fig. 8.2. $CP_1$ has a mean arrival rate given by $\bar{\lambda}$. Without loss of generality, assume that $CP_1$ has a deterministic arrival process. Every period, $CP_1$ receives $\bar{\lambda}$ requests (point A in Fig. 8.2) and pays an expected price $P_1$ to the CDN. $CP_2$ on the other hand has the same mean $\bar{\lambda}$ as $CP_1$ but has higher variance. With some high probability, $CP_2$ receives requests shown by point B; but for the remainder of the time it receives a high number of requests shown by point C. The expected price, $P_2$ paid by $CP_2$ is shown in the Figure and is clearly lower than $P_1$. This is an artifact of the concave price function. However, this is not desirable as the CP with higher variance derives greater surplus from the CDN, and hence the CDN should ideally charge $CP_2$ a higher expected price. For this reason, the CDN may choose a convex price function even though the concavity in infrastructure costs under self-provisioning exerts a force on the price function that tends to make it concave. Note also that such convexity arises only when the traffic burstiness profile is mixed and not when all CPs with the same mean arrival rate also have the same variance (Poisson traffic or MMPP with same burstiness across CPs).

If the CDN chooses a convex price function, CPs with high mean arrival rates are penalized. Consider a CP with a fixed deterministic arrival rate of $2\lambda$. Compared to a CP with fixed arrivals of $\lambda$, the CP pays a high premium for using the CDN. In contrast, this CP gets volume discounts when self-provisioning and may thus be tempted to deliver content on its own. Thus, a convex price function dissuades CPs with high volume and low variability traffic from subscribing to the CDN. Therefore, the shape of the optimal price function in the mixed traffic case depends on the distribution of traffic burstiness across CPs and the amount of volume discounts in CP's own infrastructure costs.

The analysis above indirectly suggests the inefficiency of the traditional usage-based pricing policy when the traffic profile is mixed. The policy will either penalize CPs with low burstiness or CPs with high volume, depending on whether a concave or convex price function is used. We thus consider an alternative policy, which entails pricing based on a certain high percentile of usage.

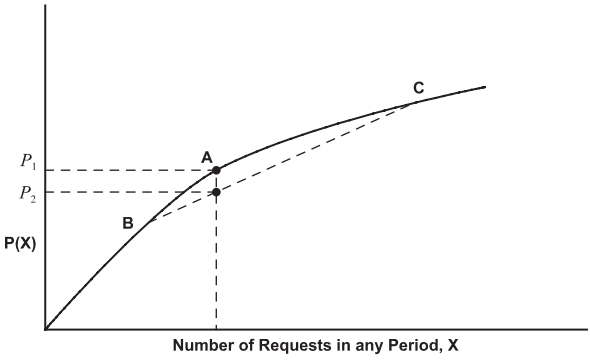



**Fig. 8.2** Expected price for a concave price function

#### 8.4.4.4 Percentile-Based Pricing is More Profitable for the CDN

We now consider pricing based on a certain high percentile of usage. Specifically, the CDN monitors the request rate, $X$, over a period of time (e.g. a month) and computes the 95th percentile of the request rate for each CP. The price to the CP is then based on the 95th percentile of his/her usage rate. Let $Z$ be the 95th percentile of request rate, $X$. As before, we restrict attention to quadratic price functions ($P(Z) = p_0 \cdot Z \pm p_1 \cdot Z^2$) to simplify computation. We numerically computed the optimal percentile-based price. In Fig. 8.3, we plot the CDN's expected profit under optimal percentile-based pricing and traditional usage-based pricing. When the traffic profile is mixed, the CDN's profit with a percentile-based pricing strategy is higher than with a traditional usage-based pricing policy. At the same time, there is no noticeable difference in profit from traditional usage-based and percentile-based pricing policies for pure Poisson and MMPP traffic. This is not surprising because once the mean request rate is fixed, the variance is also determined in both these cases,[1] and hence a mean-based pricing policy can be converted to a percentile-based policy or vice versa. With mixed traffic, percentile-based pricing permits a CDN to provide volume discounts to CPs and simultaneously charge a higher price to CPs with greater traffic burstiness.

There are some drawbacks of percentile-based billing, including complicated billing relative to traditional usage-based billing and the lack of standardization (e.g. choice of sampling times can affect the bill). This has resulted in some debate in the content delivery industry regarding the most appropriate billing policy. As a result, several CDNs such as SyncCast have adopted traditional usage-based billing because of its simplicity. However, our results suggest that when different CPs have different levels of burstiness, as expected in reality, percentile-based pricing is more profitable than traditional volume-based pricing. When traffic burstiness

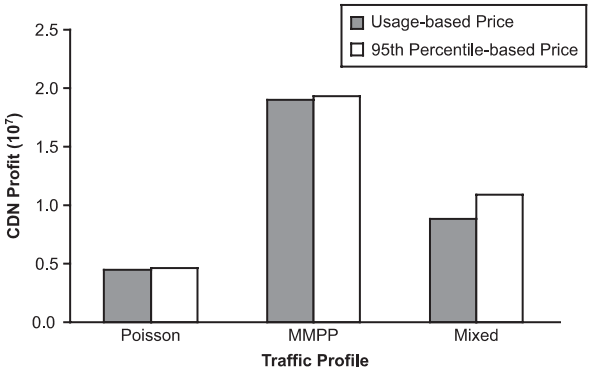



**Fig. 8.3** CDN profit with different pricing policies and traffic profiles

[1] For Poisson, the variance is equal to the mean. For our MMPP process, the variance is equal to the square of the product of burstiness (a constant) and the mean.

across subscribers is similar, CDNs can choose traditional volume-based pricing to simplify billing.

## 8.5 Visionary Thoughts for Practitioners

CDNs face a non-trivial pricing problem. On the one hand, percentile-based pricing generates considerably higher profits relative to traditional usage-based pricing schemes that do not explicitly monitor bursts or peak traffic levels. On the other hand, percentile-based pricing is perceived as a form of peak pricing and there seems to be some resistance to this form of billing among content providers and industry experts.

Despite the higher profitability of percentile pricing, we expect that content providers' preference for transparent pricing plans coupled with competitive pressure will ultimately drive a decrease in the use of percentile pricing over the next few years. If the major incumbents primarily offer percentile pricing, entrants will try to differentiate themselves by offering more transparent pricing plans. If content providers adopt the newer CDNs because of the more transparent pricing, this will in turn force the incumbents to offer more transparent plans as well.

Another possibility is that more CDNs will offer both the traditional usage-based pricing and percentile pricing and let the content providers self-select the plan they prefer. Indeed, several CDNs have started offering both billing options and let the content providers self-select the pricing policies preferred by them. However, it is not clear if this is particularly desirable for the CDNs. The content providers with highly bursty traffic will choose the traditional plans to avoid being penalized during bursts. The content providers with relatively low levels of burstiness may prefer percentile pricing plans because these plans are created to extract value from content providers with highly bursty traffic and tend to reward those with lower burstiness. Thus, the simultaneous use of both pricing plans can give rise to adverse selection where content providers choose precisely the pricing plan that the CDN would not want them to choose. Thus, CDN providers may be better off identifying a single pricing policy that is acceptable to content providers while allowing the CDNs to efficiently extract value created by them.

In summary, CDN firms need to investigate longer-term solutions. These solutions may range from educating content providers of the need to introduce percentile-based pricing to that of completely eliminating percentile-based pricing from the industry. The industry needs to be proactive in this regard.

## 8.6 Future Research Directions

This chapter highlights that there are non-trivial tradeoffs between choosing a traditional usage based pricing model and a percentile-based pricing model. On the one hand, percentile-based pricing allows the CDN to increase the price charged

to a content provider with highly bursty traffic while simultaneously providing volume discounts to a content provider with high traffic volume (and low variability in traffic). On the other hand, percentile-based pricing can be complex and may deter content providers from subscribing. As a result, many CDNs offer both pricing policies allowing content providers to self-select the plan. An interesting area of further study is the interaction between the two pricing approaches and the computation of optimal prices when both traditional and percentile based billing are offered.

In addition, much of the work on CDN pricing has focused on monopoly settings. An interesting future direction will be to investigate the impact of competition on the pricing policies chosen by the CDNs. Although Akamai has historically dominated the industry, the last couple of years have witnessed significant growth of other competitors such as Limelight Networks and Level 3. The competition among these firms seems to have a significant impact on the pricing policies of even the dominant player [15]. A study of the impact of competition on CDN pricing will help shed more light on the evolution of the industry.

Another recent trend has been the advent of P2P content delivery as an alternative to CDN-based content delivery. A number of hybrid CDN/P2P players have emerged as well. The impact of these P2P and hybrid players on the CDN market is not fully understood. A recent study by Johar et al. [11] explores the impact of P2P media delivery on the profits of a pure CDN and reveals a number of interesting insights. For example, the authors find that a P2P competitor can sometimes have a positive impact on the CDN. However, much still needs to be done to understand the interaction between P2P and CDN based content delivery.

## 8.7 Conclusions

The pricing of Content Delivery Networks is a complex problem. Subscribing content providers can be highly heterogeneous in terms of their traffic patterns and the type of content they handle. At the same time, the CDNs have to announce a single pricing policy that accounts for all these different traffic types. This has also been a source of much debate in the industry.

Some observers have questioned the simultaneous use of volume discounts and overages in the industry. The discussion in this chapter highlights that economies of scale in content delivery exert a force that drives volume discounts in CDN pricing. However, bursty traffic exerts a force that favors overages. One solution to this issue is the use of percentile-based billing. Percentile-based pricing allows a CDN to provide volume discounts to high volume content providers while charging content providers with highly bursty traffic. However, industry observers have also questioned the need for percentile-based pricing which is akin to a form of peak pricing. One answer lies in the observation that it helps generate higher profits for the CDN.

However, much remains to be understood in terms of the right pricing models for the CDN industry, the impact of competition on pricing and the economics of the CDN market. This will remain an area of considerable interest to CDN researchers and practitioners.

**Acknowledgements** Some of the materials presented in this chapter appeared in a preliminary form in HICSS'04 [7] and Wharton School Working Paper [10].

## References

1. Anderson, M., Cao, J., Kihl, M., and Nyberg, C. Performance modeling of an Apache web server with bursty arrival traffic. In *Proc. of International Conference on Internet Computing (IC)*, June 2003.
2. Cocchi, R., Shenker, S., Estrin, D., and Zhang, L. Pricing in computer networks: motivation, formulation and example. *IEEE/ACM Transactions on Networking*, vol 1, December 1993.
3. Courcoubetis, C., Antoniadis, P. Market models for P2P content distribution. In *Proc. of First International Workshop on Agents and Peer-To-Peer Computing (AP2PC)*, 2002.
4. Ercetin, O. and Tassiulas, L. Pricing strategies for differentiated services content delivery networks. *Computer Networks*, vol 49, no 6, pp 840–855, 19 December 2005.
5. Gibbens, J. and Kelly, F.P. Resource pricing and the evolution of congestion control. *Automatica* 35, 1999.
6. Gupta, A., Stahl, D.O., and Whinston, A. B. Priority pricing of integrated services networks. *Internet Economics*, eds Lee W. McKnight and Joseph P. Bailey, MIT Press, 1997.
7. Hosanagar, K., Krishnan, R., Smith, M., Chuang, J. Pricing and service adoption of content delivery networks (CDNs). In *Proc. of the Hawaii International Conference on Systems and Sciences (HICSS)*, Hawaii, January 2004.
8. Hosanagar, K., Krishnan, R., Chuang, J., and Choudhary. V. Pricing vertically differentiated web caching services. In *Proc. of the International Conference on Information Systems (ICIS)*, Barcelona, December 2002.
9. Hosanagar, K., Krishnan, R., Chuang, J., and Choudhary, V. Pricing and resource allocation in caching services with multiple levels of quality of service. *Management Science*, vol 51, no 12, 2005.
10. Hosanagar, K., Chuang, J., Krishnan, R., and Smith, M. Service adoption and pricing of content delivery network (CDN) services. *Management Science*, vol 54, no 09, 2008.
11. Johar, M., Kumar, N., and Mookerjee, V. Analyzing the Impact of Peer-to-Peer Networks on the Market for Content Provision and Distribution. University of Texas, Dallas, Working Paper, 2007.
12. Kaya, C., Dogan, K., and Mookerjee, V. An Economic and Operational Analysis of the Market for Content Distribution Services. In *Proc. of the International Conference on Information Systems*, Seattle, December 2003.
13. Kelly, F. Charging and accounting for bursty connections. *Internet Economics*, eds Lee W. McKnight and Joseph P. Bailey, MIT Press, 1997.
14. MacKie-Mason, J.K. and Varian, H.R. Pricing congestible network resources. *IEEE Journal of Selected Areas in Communications*, vol 13, no 7, pp 1141–149, September 1995.
15. Malik, O. Akamai and the CDN Price Wars. GigaOM Blog, August, 2007.
16. Mendelson, H. Pricing computer services: queuing effects. *Communications of the ACM*, vol 28, 1990.
17. Mendelson, H., and Whang, S. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations Research*, vol 38, 870–83, 1990.
18. Rayburn, D. Content delivery pricing: understanding CDN overages." Streamingmedia Blog, October 2007.
19. Scott, S. L. and Smyth, P. The Markov modulated Poisson process and Markov Poisson cascade with applications to web traffic modeling." In *Bayesian Statistics*, Oxford University Press, 2003.
20. Vakali, A. and Pallis, G. Content delivery networks: status and trends. *IEEE Internet Computing* vol 7, no 6, pp 68–74, 2003.

# Chapter 9
# Mathematical Models for Resource Management and Allocation in CDNs

Tolga Bektaş and Iradj Ouveysi

## 9.1 Introduction

To achieve a cost-effective content delivery strategy that a CDN provider seeks, the resources of a CDN, consisting primarily of the network infrastructure, the content to be distributed in the network, and the caching servers (holding a set of objects) that are to be distributed throughout the network, need to be efficiently managed and allocated. Now that the customer preferences have begun to play a key role in provisioning CDN services, the provider should also take into account some specific Quality-of-Service (QoS) considerations in planning its content delivery activities.

Mathematical modeling is a powerful and an effective tool that can be used to efficiently solve the resource allocation and management problems in a CDN. The aim of this chapter is to demonstrate how a variety of problems of this domain can be formulated in terms of mathematical models, and how the resulting models can be solved efficiently using the available techniques. For this purpose, we review the recent literature in the next section; simultaneously describe the relevant work and present the associated mathematical models. Solution techniques that we believe to be appropriate for the resolution of these models are described in Sect. 9.3, where we will also illustrate how these techniques can be applied to some of the models presented in this chapter. Section 9.4 offers some new models for a number of CDN architectures, and Sect. 9.5 presents their performance results. We offer our thoughts for practitioners in Sect. 9.6, provide directions for further research in Sect. 9.7 and state our conclusions in Sect. 9.8.

Tolga Bektaş
School of Management, University of Southampton, Highfield, Southampton SO17 1BJ, UK,
e-mail: T.Bektas@soton.ac.uk

Iradj Ouveysi
Honorary research fellow, Electrical and Electronic Engineering Department, The University of Melbourne, Vic. 3010, Australia, e-mail: iradjouveysi@yahoo.co.uk

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

## 9.2 Related Work

In this section, we review the relevant literature that offer mathematical models for resource management and allocation in CDNs, and at the same time present the related mathematical models. Before presenting the models, we define the terminology that will be used throughout the chapter. The term *content* refers to any kind of information that is available on the World Wide Web to public such as Web pages, multimedia files and text documents. *Object* refers to a specific item of the content, such as a sound file or a text document. The *content provider* issues content for the access of others, and a *CDN provider* (most often a commercial one) disseminates the content on behalf of the content provider. There may be a few exceptions where the content provider takes care of the content delivery itself, but in this chapter we shall assume that this task is outsourced to a CDN provider by the content provider. The term *client* refers to an individual (either person or corporate) who issue requests for content. The CDN providers hold either the whole or a subset of the content in *caching servers* that are deployed throughout the telecommunications network, through or by which client requests are served. For all the models that follow, we assume a given complete network $G = (V, E)$, where $V$ is the set of nodes and $E = (\{i, j\} : i, j \in V)$ is the set of links. The node set $V$ is further partitioned into three nonempty subsets $I$, $J$ and $S$, where $I$ is the set of clients, $J$ is the set of nodes where caching servers[1] are (or can be) installed, and $S$ is the set containing the origin servers ($S = \{0\}$ in case of a single origin server). All the models presented in this section use a common notation framework that is given in Table 9.1.

### *9.2.1 The Fundamental Problems*

There are three fundamental problems that arise in designing a cost-effective delivery network on which most of the more complex mathematical models proposed within the CDN domain are based on. This section presents a brief overview of these three problems.

***Caching server placement problem.*** Given an existing infrastructure, the *caching server placement problem* consists of optimally placing a given number of servers to a given number of sites, such that a cost function (overall flow of traffic, average delay the clients experience, and total delivery cost) is minimized [22]. Qiu et al. [26] offer two well-known mathematical models to the caching server placement problem, namely the uncapacitated $p$-median (e.g. see [2]) and facility location problems (e.g. see [11]). The problem of placing transparent caches is described by Krishnan et al. [19]. The objective function considered in this study is interesting in that it

[1] We have chosen to use the term *caching server* as opposed to *proxy server* to avoid confusion as the concept of proxy was originally used to perform filtering and request relay, etc., but this is not practical any more as web pages are changing fast and dynamically. It is therefore more appropriate to refer to the additional servers as caching servers, or simply caches.

**Table 9.1** Summary of the notation used in the chapter

| Sets | |
|---|---|
| $I$ | Set of clients ($I \subset V$) |
| $J$ | Set of nodes on which caching servers can be established ($J \subset V$) |
| $S$ | Set of origin servers ($S \subset V$) |
| $K$ | Set of objects |
| Parameters | |
| $b_k$ | Size of object $k \in K$ |
| $\lambda_i$ | Aggregate request rate of client $i \in I$ |
| $h_{ij}$ | Fraction of the request originating from node $i \in I$ that can be satisfied by $j \in J$ |
| $c_{ij}$ | 'Distance' between two nodes $i \in V$ and $j \in V$ (i.e. number of hops, cost) |
| $f_{ij}$ | Amount of flow between a client $i \in I$ and a caching server $j \in J$ |
| $d_{ik}$ | Request rate of client $i \in I$ for object $k \in K$ per unit time |
| $f_j$ | Cost of operating a caching server on node $j \in J$ |
| $\psi_j$ | Cost of placing an object on a caching server $j \in J$ |
| $\beta_j$ | Cost per unit of bandwidth required by caching server $j \in J$ |
| $\delta_j$ | Cost per unit of processing power required by caching server $j \in J$ |
| $C_j$ | Units of processing power available at a caching server $j \in J$ |
| $s_j$ | Storage capacity of a caching server $j \in J$ |
| $l_k$ | Amount of bandwidth consumed by object $k \in K$ |
| $pw_k$ | Amount of processing power consumed by object $k \in K$ |
| $\rho_k$ | Revenue generated by providing object $k \in K$ to the clients |
| $L_{ij}$ | Latency between two nodes $i \in V$ and $j \in V$ |
| $\Delta_d$ | Upper bound on latency (may be defined in terms of a link or an object, or both |
| $p_{jk}$ | Probability that object $k \in K$ exists at caching server $j \in K$ |
| Variables | |
| $\vartheta_{jk} \in \{0,1\}$ | 1, if request for object $k \in K$ is directed to caching server $j \in J$; 0, otherwise |
| $x_{ij} \in \{0,1\}$ | 1, if client $i \in I$ is assigned to caching server $j \in J$; 0, otherwise |
| $x_{ijk} \in \{0,1\}$ | 1, if object $k \in K$ requested by client $i \in I$ is held at caching server $j \in J$; 0, otherwise |
| $y_j \in \{0,1\}$ | 1, if a caching server is active at node $j \in J$; 0, otherwise |
| $z_{jk} \in \{0,1\}$ | 1, if object $k \in K$ is placed on a caching server $j \in J$; 0, otherwise |
| $u_k \in \{0,1\}$ | 1, if object $k \in K$ is replicated (on any caching server $j \in J$); 0, otherwise |
| $r^k_{ji} \geq 0$ | fraction of accesses for object $k \in K$ directed to server $j \in J$ requested by client $i \in I$ |

considers the case where the requested content is *not* found in a specific caching server. Thus, the cost of serving client $i$ from server $j$ is given by the following:

$$cost(i,j) = f_{ij}(h_{ij}c_{ij} + (1-h_{ij})(c_{ij} + c_{js})), \tag{9.1}$$

This cost function (9.1) is a good representation of how a CDN operates and has been used in formulating other problems (e.g. see [6, 17]).

***Request routing.*** *Routing* in a computer network refers to sending data from one or more sources to one or more destinations so as to minimize the total traffic flowing on the network. For a detailed review on the problem as well as a survey of combinatorial optimization applications, we refer the reader to [24]. *Request routing*, on the other hand, is basically the process of guiding a client's request to a

suitable caching server that is able to serve the corresponding request. The problem is formally defined as, given a request for an object, selecting a server to address the request such that a cost function is minimized. For a mathematical formulation of the problem (albeit a simplified one) the reader may refer to [14].

***Object placement.*** Previously mentioned studies assume that the content held in the origin server is entirely replicated onto the caching servers (in case of which the caching servers are usually referred to as *replica*s or *mirror*s). Unfortunately, this may not always be possible in situations where the objects are significantly large in size (i.e. multimedia files) and only a partial replication can be performed due to the limited storage capacity of the caching servers. In this case, any caching server can only hold a subset of the content. Determining which objects should be placed at each caching server under storage capacity restrictions is known as the *object placement problem*. The reader may see [18] for a mathematical model of this problem.

### *9.2.2 Integrated Problems*

In this section, we present and discuss some of the more complex issues in CDNs in which several problems mentioned above jointly arise. We start by the static data placement problem defined on a network with no origin server which consists of placing objects so as to minimize the total access cost (the cost for a client $i \in I$ to access object $k \in K$ from node $j \in J$ is $b_k d_{ik} c_{ij}$). A mathematical model for this problem, as offered by Baev et al. [4], is given below.

$$\textbf{(M1)} \qquad \text{Minimize} \qquad \sum_{i\in I}\sum_{j\in I}\sum_{k\in K} b_k d_{ik} c_{ij} x_{ijk} \tag{9.2}$$

subject to

$$\sum_{j\in I} x_{ijk} = 1 \qquad \forall i \in I, k \in K \tag{9.3}$$

$$x_{ijk} \le z_{jk} \qquad \forall i, j \in I, k \in K \tag{9.4}$$

$$\sum_{k\in K} b_k z_{jk} \le s_j \qquad \forall i \in I \tag{9.5}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in I, k \in K \tag{9.6}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j \in I, k \in K. \tag{9.7}$$

Model **M1** uses, a binary variable $z_{jk}$ that equals 1 if object $k \in K$ is held at caching server $j \in J$, and 0 otherwise; as well as a three-index binary variable $x_{ijk}$ that is equal to 1 if object $k \in K$ requested by client $i \in I$ is served by node $j \in J$ that holds a copy, and 0 otherwise. In this model, the objective function (9.2) expresses the total cost of serving requests for all nodes and objects. Note that $J = I$, i.e. each node acts both as a client and a potential caching server. Constraint (9.3) expresses

that each node's request should be forwarded to exactly one node. Constraint (9.4) indicates that an assignment to a node can only be made if that specific node is holding the requested object. Finally, constraint (9.5) relates to the limited capacity ($s_j$) of each node $j \in J$.

Laoutaris et al. [20] study the joint problem of object placement and node dimensioning, where the latter refers to determining the fraction of a given a total storage capacity to allocate to each node of the network. The overall aim is to minimize the average distance from all clients to all the requested objects. The authors assume that all the objects are unit-sized. Another study by the same authors [21] describes a model to solve the storage capacity allocation problem in CDNs, taking into account decisions pertaining to the location of the caching servers to be installed, the capacity that should be allocated to each caching server, and the objects that should be placed in each caching server. This model is defined on a tree network, and each node $i$ has a set of ancestors denoted by $a(i)$, and a set of leaves denoted by $l(i)$. The model is rewritten based on the notation introduced earlier, as opposed to that used in [21].

$$\textbf{(M2)} \qquad \text{Maximize} \qquad \sum_{i\in I}\lambda_i \sum_{k\in K} d_{ik} \sum_{v\in a(i)} (c_{is} - c_{iv}) x_{ijk}$$

subject to

$$\sum_{v\in a(i)} x_{ijk} \leq 1 \qquad \forall i \in I, k \in K \tag{9.8}$$

$$\sum_{v\in l(i)} x_{ijk} \leq M z_{jk} \qquad \forall i \in I, k \in K \tag{9.9}$$

$$\sum_{j\in J}\sum_{k\in K} z_{jk} \leq D \tag{9.10}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in I, j \in J, k \in K \tag{9.11}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \tag{9.12}$$

As one can see, this model is quite similar to **M1** presented by Baev et al. [4] but differs with respect to the objective function, which maximizes the savings that one can obtain by the placement of objects on the caching servers. Constraints (9.8) and (9.9) are related to the assignment of customers to the caching servers, where $M$ is a sufficiently big number. Constraints (9.10) ensure that the node dimensioning is performed without exceeding the available resource of storage capacity, denoted by $D = \sum_{j\in J} s_j$. Since all objects are assumed to be unit-sized by Laoutaris et al. [21], the dimension of a node is therefore equivalent to the number of objects placed on that node.

Nguyen et al. [23] consider the problem of provisioning CDNs on shared infrastructures and propose a joint provisioning and object replication model so as to minimize the total cost of storage, request serving, and start-up. We will present their model here in terms of the already defined notations but also define the following additional parameters: An object can be placed on each caching server at

a unit cost $\psi_j$ and a unit bandwidth cost $\beta_j$, while the unit processing power cost is denoted by $\delta_j$ and a total of $C_j$ units of processing power is available at each caching server. Each object $k$ consumes $l_k$ units of bandwidth and $c_k$ units of processing power. The service provider has revenue of $\rho_k$ from each object $k$ per unit time. Latency between two nodes $i$ and $j$ is denoted by $L_{ij}$ which should be limited by an upper bound $\Delta_d$. An additional binary decision variable $u_k$ denotes if an object is replicated (in any caching server) or not, and variable $r_{ji}^k$ denotes the fraction of accesses for object $k$ requested by customer $i$ that should be directed to server $j$.

$$\textbf{(M3)} \quad \text{Maximize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} r_{ji}^k \rho_k - \sum_{j\in J}\sum_{k\in K} \psi_j b_k z_{jk} - \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} r_{ji}^k(\beta_j l_k + \delta_j c_k) - \sum_{j\in J} f_j y_j \tag{9.13}$$

subject to

$$\sum_{i\in I} r_{ji}^k pw_k \le C_j y_j \qquad \forall j \in J \tag{9.14}$$

$$\sum_{j\in J} r_{ji}^k pw_k = u_k d_{ik} \qquad \forall i \in I, k \in K \tag{9.15}$$

$$r_{ji}^k (L_{ij} - \Delta_d) \le 0 \qquad \forall i \in I, j \in J, k \in K \tag{9.16}$$

$$r_{ji}^k \le d_{ik} z_{jk} \qquad \forall i \in I, j \in J, k \in K \tag{9.17}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{9.18}$$

$$u_k \in \{0,1\} \qquad \forall k \in K \tag{9.19}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \tag{9.20}$$

Model **M3** has an objective function which maximizes the profit of the service provider, calculated by subtracting from the total revenue (first component of (9.13)) the total cost related to storage, bandwidth, CPU and site establishment. Constraints (9.14) enforce the capacity restrictions for each server whereas constraints (9.15) ensure that all requests are served. Constraints (9.16) guarantee that all requests are served within the allowable latency bound. Finally, constraints (9.17) dictate that a request can be served by a caching server only when the requested object is available therein.

The joint problem of server location, object placement and request routing is studied by Bektaş et al. [9], where a new model is proposed that extends the standard facility location model to CDNs by considering multiple objects and an incorporation of a suitable, albeit nonlinear, objective function similar to (9.1). The integer programming model, as proposed by Bektaş et al. [9], is given below:

$$\textbf{(M4)} \quad \text{Minimize} \sum_{j\in J} f_j y_j + \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} (b_k d_{ik} c_{ij} z_{jk} x_{ij} + b_k d_{ik}(1 - z_{jk})(c_{jS} + c_{ij}) x_{ij}) \tag{9.21}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \tag{9.22}$$

$$x_{ij} \le y_j \qquad \forall i \in I, j \in J \tag{9.23}$$

$$\sum_{k\in K} b_k z_{jk} \le s_j y_j \qquad \forall j \in J \tag{9.24}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{9.25}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{9.26}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \tag{9.27}$$

The objective function of model **M4** is a generalization of (9.1) to multiple clients, servers and objects. The first component denotes the total cost of caching server establishment. The second component has two parts, where the first part corresponds to the costs of serving the clients from the caching servers and the second part reflects the additional costs that occur in accessing the origin server when the requested object is not found in the corresponding caching server. Constraints (9.22) ensure that each client is assigned to a single caching server and constraints (9.23) dictate that this assignment is only possible when the server is active. Overcapacity usage in placing the objects onto each caching server is prohibited by constraints (9.24).

Bektaş et al. [8] have later considered the problem from an operational level by excluding the caching server deployment decisions, but at the same time, imposing a QoS constraint that imposes a limit on end-to-end object transfer delays, and propose the following model.

$$\textbf{(M5)} \qquad \text{Minimize} \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} (b_k d_{ik} c_{ij} z_{jk} x_{ij} + b_k d_{ik} (1 - z_{jk})(c_{jS} + c_{ij}) x_{ij})$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \tag{9.28}$$

$$\sum_{j\in J} L_{ij} x_{ij} z_{jk} + \sum_{j\in J} (L_{ij} + L_{j0}) x_{ij} (1 - z_{jk}) \le \Delta_d \qquad \forall i \in I, j \in J, k \in K \tag{9.29}$$

$$\sum_{k\in K} b_k z_{jk} \le s_j y_j \qquad \forall j \in J \tag{9.30}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{9.31}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \tag{9.32}$$

Model **M5** has a similar structure to **M4**. However, it excludes the caching server deployment decisions, but incorporates QoS restrictions represented by constraints (9.29). It has been observed that one can write constraints (9.29) in a much simpler form as $x_{ij} \le z_{jk}$, $\forall i \in I, j \in J, k \in \mathscr{Q}_{ij}$, where $\mathscr{R}_{ij} = \{k \in K | t(b_k, d_{ij}) > \Delta_d\}$ and $\mathscr{Q}_{ij} = \{k \in K | L_{ij}) \le \Delta_d$ and $(L_{ij} + L_{j0}) > \Delta_d\}$ for each pair $(i, j)$ [8]. The former relates to objects $k$ for which the time required to transfer such objects $k$ from

caching server $j$ exceeds the allowable delay limit, and the latter consists of a subset of objects $k$ for which the time required to transfer such objects from caching server $j$ is within allowable delay limit, but does not allow for retrieval of this object from the caching server due to the QoS constraint.

All of the above models are based on the assumption that the CDN operates with a single origin server (i.e. $|S| = 1$). While this is most often the case in practice, there are situations where a content provider may deploy multiple origin servers (possibly on the same site) for a variety of reasons, such as increasing system reliability or the storage capacity. To take into account multiple origin servers, a model is proposed by Bektaş et al. [6] for the joint problem of caching server placement, request routing, and object placement. We present this model in the following,

$$\textbf{(M6)} \quad \text{Minimize} \sum_{j\in J} f_j y_j + \sum_{i\in I}\sum_{j\in J}\sum_{s\in S}\sum_{k\in K} (b_k d_{ik} c_{ij} z_{jk} x_{ij} + b_k d_{ik}(1-z_{jk})(c_{ij}+c_{js}t_{js})x_{ij}) \tag{9.33}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \tag{9.34}$$

$$x_{ij} \le y_j \qquad \forall i \in I, j \in J \tag{9.35}$$

$$\sum_{k\in K} b_k z_{jk} \le s_j y_j \qquad \forall j \in J \tag{9.36}$$

$$\sum_{s\in S} t_{js} = 1 \qquad \forall j \in J \tag{9.37}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{9.38}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{9.39}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.40}$$

$$t_{js} \in \{0,1\} \qquad \forall j \in J, s \in S. \tag{9.41}$$

Model **M6** is an extension of **M4** to multiple origin servers and uses an additional binary variable $t_{js}$ that is equal to 1 if caching server $j \in J$ is assigned to an origin server $s \in S$, and 0 otherwise. Note that the objective function has been augmented so as to consider all the available origin servers, and an extra constraint (9.37) has been added that dictates each caching server should be assigned to a single origin server to further forward the requests for objects which they do not hold, in the event that these objects are requested by their clients.

## 9.3 Solution Algorithms

There are two classes of algorithms for the solution of the above mentioned problems and the associated mathematical models. The first is the class of exact

algorithms which are able to yield optimal solutions at the expense of rather significant computational times, and the second is the class of heuristic algorithms which usually require relatively small amount of computational effort, but unfortunately unable to guarantee the identification of the optimal solution. Amongst a number of available exact solution methods, we will focus here on two methods that are based on decomposition, since they allow for a break-down of the original model into smaller sized and easier-to-solve subproblems.

### *9.3.1 Benders' Decomposition*

Benders Decomposition [10] is a technique that allows a model to be split into two subproblems. More specifically, given a model of the following form,

$$(\mathscr{P}) \qquad \text{Minimize } \mathbf{cx}+\mathbf{fy} \text{ subject to } \mathbf{Ax}+\mathbf{By}=\mathbf{d}, \mathbf{x}\in\mathbf{X}, \mathbf{y}\in\mathbf{Y}, \tag{9.42}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the column vectors of variables, $\mathbf{c}$ and $\mathbf{f}$ are the row vectors of cost coefficients, $\mathbf{A}$ and $\mathbf{B}$ are the constraint coefficient matrices, and $\mathbf{d}$ is the column vector of right hand side values, all with appropriate dimensions. $\mathbf{X}$ and $\mathbf{Y}$ are nonempty (we assume that the former to be continuous and the latter integer) sets in which variables $\mathbf{x}$ and $\mathbf{y}$ are defined, respectively.

To illustrate the application of Benders' decomposition on problem $\mathscr{P}$, we first rewrite problem $\mathscr{P}$ in the following form.

$$(\mathscr{P}_1) \qquad \underset{\tilde{\mathbf{y}}\in\mathbf{Y}}{min}\{\mathbf{f}\tilde{\mathbf{y}}+\underset{\mathbf{x}\in\mathbf{X}}{min}\{\mathbf{cx}:\mathbf{Ax}=\mathbf{d}-\mathbf{B}\tilde{\mathbf{y}}\}\}, \tag{9.43}$$

where $\mathbf{y}$ is preset as $\mathbf{y}=\tilde{\mathbf{y}}$. Since the inner minimization problem in $\mathscr{P}_1$ (denoted by $\mathscr{S}$) expressed in terms of the $x$ variables only is linear and continuous, one can replace it by its dual using dual variables $\mathbf{w}$ to each of the constraints of $\mathscr{S}$:

$$\underset{\tilde{\mathbf{y}}\in\mathbf{Y}}{min}\{\mathbf{f}\tilde{\mathbf{y}}+\underset{\mathbf{w}}{max}\{\mathbf{w}(\mathbf{d}-\mathbf{B}\tilde{\mathbf{y}}):\mathbf{wA}\le\mathbf{c}\}\}, \tag{9.44}$$

Assuming that the feasible region of the dual of $\mathscr{S}$ is nonempty (as otherwise this would imply the primal problem being either infeasible or unbounded), the original problem $\mathscr{P}$ can be rewritten as,

$$\text{Minimize} \qquad z+\mathbf{fy} \tag{9.45}$$

subject to

$$z \ge \boldsymbol{\tau}^{\mathbf{r}}(\mathbf{d}-\mathbf{By}) \qquad \tau^r\in\Upsilon \tag{9.46}$$

$$\boldsymbol{\varsigma}^{\mathbf{u}}(\mathbf{d}-\mathbf{By})\le 0 \qquad \varsigma^u\in\Psi \tag{9.47}$$

$$\mathbf{y}\in\mathbf{Y}, \tag{9.48}$$

called the Master Problem. Sets $\Upsilon$ and $\Psi$ denote extreme points and extreme rays of the feasible space of the dual problem, respectively. Constraints (9.46) are those defined for each extreme point of the feasible region of the dual of $\mathscr{S}$, and constraints (9.47) are those written for each extreme ray of the dual of $\mathscr{S}$, whenever it is infeasible.

The authors of [8] observe that model **M5** has a special structure which makes it suitable for the application Benders' decomposition. We illustrate this on a linearization of model **M5** using auxiliary linearization variables $\varphi_{ijk}$. These variables correspond to the product $x_{ij}z_{jk}$ in the objective function of **M5** (see [8] for linearization details):

$$\text{Minimize} \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}\left(b_k\lambda_{ik}\left(c_{ij}+c_{j0}\right)x_{ij}-b_k\lambda_{ik}c_{j0}\varphi_{ijk}\right) \tag{9.49}$$

subject to

$$\sum_{j\in J}x_{ij}=1, \qquad \forall i\in I \tag{9.50}$$

$$\sum_{k\in K}b_k z_{jk}\le s_j \qquad \forall j\in J \tag{9.51}$$

$$\varphi_{ijk}-x_{ij}\le 0 \qquad \forall i\in I, j\in J, k\in K \tag{9.52}$$

$$\varphi_{ijk}-z_{jk}\le 0 \qquad \forall i\in I, j\in J, k\in K \tag{9.53}$$

$$x_{ij}-z_{jk}\le 0 \qquad \forall i\in I, j\in J, k\in \mathscr{Q}_{ij} \tag{9.54}$$

$$x_{ij}\ge 0 \qquad \forall i\in I, j\in J \tag{9.55}$$

$$z_{jk}\in\{0,1\} \qquad \forall j\in J, k\in K \tag{9.56}$$

$$\varphi_{ijk}\in[0,1] \qquad \forall i\in I, j\in J, k\in K. \tag{9.57}$$

Upon fixing the object location variables that appear in this linearization to some feasible configuration as $z_{jk}=\bar{z}_{jk}$, the resulting subproblem further decomposes into smaller problems for each client $i\in I$, shown as follows:

$$\text{Minimize} \sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij}-b_k\lambda_{ik}c_{j0}\varphi_{ijk}) \tag{9.58}$$

subject to

$$\sum_{j\in J}x_{ij}=1 \tag{9.59}$$

$$\varphi_{ijk}-x_{ij}\le 0 \qquad \forall j\in J, k\in K \tag{9.60}$$

$$\varphi_{ijk}\le z^*_{jk} \qquad \forall j\in J, k\notin \mathscr{Q}_{ij} \tag{9.61}$$

$$x_{ij}\le z^*_{jk} \qquad \forall j\in J, k\in \mathscr{Q}_{ij} \tag{9.62}$$

$$x_{ij}\ge 0 \qquad \forall j\in J.$$

Each subproblem, although still integer, is observed to bear the integrality property. This distinctive feature allows one to relax the integrality restrictions on the $x_{ij}$

variables and solve its dual. Let $\alpha_i$, $\theta_{ijk}$, $\omega_{ijk}$ and $\zeta_{ijk}$ be the dual variables corresponding to constraints (9.59), (9.60), (9.61) and (9.62), respectively. One can then construct the master problem as follows,

$$\text{Minimize} \sum_{i \in I} \xi_i \tag{9.63}$$

subject to

$$\xi_i + \sum_{j \in J} \sum_{k \in K} z_{jk}(\tilde{\omega}_{ijk} + \tilde{\zeta}_{ijk}) \geq \tilde{\alpha}_i \qquad (\alpha, \theta, \omega, \zeta) \in \mathscr{P}_i^{\mathscr{D}} \tag{9.64}$$

$$\tilde{\alpha}_i - \sum_{j \in J} \sum_{k \in K} z_{jk}(\tilde{\omega}_{ijk} + \tilde{\zeta}_{ijk}) \leq 0, \qquad (\alpha, \theta, \omega, \zeta) \in \mathscr{W}_i^{\mathscr{D}} \tag{9.65}$$

$$\sum_{k \in K} b_k z_{jk} \leq s_j \qquad \forall j \in J$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K,$$

where (9.64) are the optimality constraints with the coefficients corresponding to an optimal solution to the dual problem and calculated as follows,

$$\tilde{\alpha}_i = \min_{j \in \mathscr{F}_i \cup \mathscr{H}_i} \left\{ \sum_{k \in K} b_k \lambda_{ik}(c_{ij} + c_{j0}) - \sum_{k \in K : z^*_{jk} = 1} b_k \lambda_{ik} c_{j0} \right\}$$

$$\tilde{\omega}_{ijk} = \begin{cases} b_k \lambda_{ik} c_{j0}, & \text{if } z^*_{jk} = 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\sum_{k \in K : z^*_{jk} = 0} \tilde{\zeta}_{ijk} = \tilde{\alpha}_i + \sum_{k \in K} \tilde{\theta}_{ijk} - \sum_{k \in K} b_k \lambda_{ik}(c_{ij} + c_{j0}) \qquad \forall j \in J,$$

where $\mathscr{F}_i = \{j \in J | \mathscr{Q}_{ij} = \mathscr{R}_{ij} = \emptyset\}$ and $\mathscr{H}_i = \{j \in J | z^*_{jk} = 1, \forall k \in \mathscr{Q}_{ij}\}$. Constraints (9.65) are written for every extreme ray that correspond to an infeasible solution to the dual problem. Due to the number of optimality and infeasibility constraints that are present in the master problem, it is not practically possible to solve it as is. One therefore needs to resort to a strategy where one starts with a restricted master problem including only a limited number constraints, and additional constraints are iteratively added to this restricted problem until the optimal solution is reached. The reader is referred to [8] for details of this approach along with various refinements that are used to increase the efficiency of the algorithm, such as the use of Pareto-optimal cuts and cut elimination.

### *9.3.2 Lagrangean Relaxation and Decomposition*

Lagrangean relaxation is an approach where some of the constraints in a model are dualized (or relaxed) in a Lagrangean fashion so as to obtain a problem that is easier

to solve. Consider problem $\mathscr{P}$ presented in the previous section and assume that constraints $\mathbf{Ax}+\mathbf{By}=\mathbf{d}$ are those that "complicate" the model. Using a vector of Lagrangean multipliers denoted by $\mu$, these constraints can be dualized as shown in the following,

$$\begin{array}{lll} (\mathscr{P}_\mu) & \text{minimize} & \mathbf{cx}+\mathbf{fy}+\mu(\mathbf{Ax}+\mathbf{By}-\mathbf{d}) \\ & \text{subject to} & \mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}, \end{array}$$

which yields an immediate decomposition of $\mathscr{P}_\mu$ into two subproblems, one being

$$\min_{x\in\mathbf{X}}(\mathbf{c}+\mu\mathbf{A})\mathbf{x}),$$

defined only in x variables, and the other being

$$\min_{y\in\mathbf{Y}}(\mathbf{f}+\mu\mathbf{B})\mathbf{y}),$$

that is defined only in y variables. The solution value of $\mathscr{P}_\mu$, for any given $\mu$, is a lower bound on the optimal solution value of $\mathscr{P}_\mu$. To find the best possible lower bound, one has to solve the following piecewise linear concave optimization problem, $\max_\mu \mathscr{P}_\mu$, usually named as the Lagrangean dual problem and solved by means of nondifferentiable optimization techniques.

There are several applications of Lagrangean relaxation to tackle some of the problems mentioned earlier, including the one proposed by Qiu et al. [26] for the server placement problem, by Nguyen et al. [23] for the overlay distribution network provisioning problem, and by Bektaş et al. [8] for the joint problem of object placement and request routing in a CDN. In this section, we will illustrate the use of this technique on an integer linear programming model proposed in [8], but with a different type of relaxation. The model we present below is an alternative linearization of **M5** using an auxiliary linearization variable $v_{jk}$.

$$\text{Minimize} \qquad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij}-\sum_{j\in J}\sum_{k\in K} v_{jk} \tag{9.66}$$

subject to

$$\sum_{j\in J} x_{ij}=1 \qquad \forall i\in I \tag{9.67}$$

$$\sum_{k\in K} b_k z_{jk}\le s_j \qquad \forall j\in J \tag{9.68}$$

$$v_{jk}-Mz_{jk}\le 0 \qquad \forall j\in J,k\in K \tag{9.69}$$

$$v_{jk}-\sum_{i\in I} b_k\lambda_{ik}c_{j0}x_{ij}\le 0 \qquad \forall j\in J,k\in K \tag{9.70}$$

$$x_{ij}-z_{jk}\le 0 \qquad \forall i\in I,j\in J,k\in\mathscr{Q}_{ij} \tag{9.71}$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J \tag{9.72}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.73}$$

$$v_{jk} \geq 0 \qquad \forall j \in J, k \in K. \tag{9.74}$$

By dualizing constraints (9.67), (9.69) and (9.70) using respectively $\sigma_i$, $\pi_{jk}$ and $\eta_{jk}$ as the Lagrangean multipliers, we obtain the following relaxed problem denoted by $\mathscr{R}$:

$$(\mathscr{R}) \qquad \text{Minimize} \qquad \sum_{i\in I}\sum_{j\in J}\left(\sum_{k\in K}(b_k d_{ik}(c_{ij}+c_{j0}(1-\eta_{jk})))-\sigma_i\right)x_{ij} + \sum_{j\in J}\sum_{k\in K}\left(\pi_{jk}+\eta_{jk}-1\right)v_{jk} - M\sum_{j\in J}\sum_{k\in K}\pi_{jk}z_{jk} - \sum_{i\in I}\sigma_i \tag{9.75}$$

subject to

$$\sum_{k\in K} b_k z_{jk} \leq s_j \qquad \forall j \in J \tag{9.76}$$

$$x_{ij} - z_{jk} \leq 0 \qquad \forall i \in I, j \in J, k \in \mathscr{Q}_{ij} \tag{9.77}$$

$$x_{ij} \geq 0 \qquad \forall i \in I, j \in J \tag{9.78}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.79}$$

$$v_{jk} \geq 0 \qquad \forall j \in J, k \in K. \tag{9.80}$$

Problem $\mathscr{R}$ decomposes into two subproblems, one in $x$ and $z$ variables, and the other in $v$ variables. The latter is solvable through inspection by setting $v_{jk} = 0$ if $\pi_{jk} + \eta_{jk} - 1$ is nonnegative, and $v_{jk} = 1$ otherwise. As for the former subproblem, notice that the $x$ variables only appear in constraints (5), and thus can be fixed to $x_{ij} = 0$ if $\sum_{k\in K}(b_k d_{ik}(c_{ij} + c_{j0}(1-\eta_{jk}))) - \sigma_i$ is nonnegative and $x_{ij} = \hat{z}_{jk}$ otherwise, where $\hat{z}_{jk}$ is the solution to the following problem,

$$\text{Maximize} \qquad \sum_{j\in J}\sum_{k\in K}\pi_{jk}z_{jk}$$

subject to

$$\sum_{k\in K} b_k z_{jk} \leq s_j \qquad \forall j \in J$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K,$$

which further decomposes into a series of binary knapsack problems, one for each $j \in J$, each of which can be solved efficiently in $\mathscr{O}(|K|s_j)$ time using dynamic programming.

### 9.3.3 Heuristic Algorithms

Contrary to exact algorithms, heuristic algorithms are fast and scalable solutions methods for instances that are beyond the reach of exact algorithms for problems of a CDN provider. *Greedy heuristics* are heuristic algorithms that search for a solution to a given problem by choosing the best possible alternative at each iteration (i.e. the option that reduces the cost by the greatest amount) but neglects the effect decision on the overall search. These algorithms therefore yield locally optimal solutions most of the time. The advantage of such heuristics lie in their computational speed at tackling problems and scalability in being applicable to very large instances. This explains the popularity of greedy heuristics in the CDN literature and we refer the reader to, amongst many others, [18, 21, 26, 27, 31]. *Approximate algorithms*, on the other hand, are heuristics that guarantee to find a solution with a value that is within a constant factor of the optimal solution and for which applications within the CDN domain can be found [4, 20]. *Simulated annealing* and *Tabu search* belong to a class of more sophisticated heuristic techniques, named as metaheuristics, in that they make use of special mechanisms to prevent the search from being trapped in the local minima. We refer the reader to [7] for a two-level implementation of simulated annealing for the joint problem of object placement and request routing, and to [15] for an application of a tabu search algorithm on the same problem.

**Table 9.2** A categorization of the existing models and solution approaches

| SP | RR | OP | CD | Reference | Solution Approaches |
|---|---|---|---|---|---|
| x | | | | [22] | Dynamic programming |
| x | | | | [26] | Lagrangean relaxation, greedy heuristics |
| x | | | | [27] | Greedy heuristics |
| x | | | | [5] | Greedy heuristics |
| x | | | | [17] | Dynamic programming |
| | x | | | [14] | Integer programming models |
| | | x | | [13] | Greedy heuristics |
| | | x | | [18] | Greedy heuristics |
| | | x | x | [20] | Exact and approximate algorithms |
| | | x | x | [21] | Greedy heuristics |
| x | x | | | [1] | Heuristic algorithms |
| x | | x | | [29] | Dynamic programming |
| x | | x | | [30] | Heuristic algorithms |
| | x | x | | [4] | Approximate algorithm |
| | x | x | | [3] | Heuristic algorithms |
| | x | x | | [28] | Analytic and heuristic algorithms |
| | x | x | | [7] | Simulated annealing |
| | x | x | | [8] | Benders' decomposition, Lagrangean relaxation |
| | x | x | | [15] | Tabu search |
| x | x | x | | [23] | Lagrangean relaxation |
| x | x | x | | [9] | Benders' decomposition, greedy heuristic |

Table 9.2 presents a categorization of the existing models and solution approaches for a variety of resource allocation and management problems (i.e. Server Placement (SP), Request Routing (RR), Object Placement (OP), and Content Delivery (CD)) in CDNs, including additional references. It is not meant to be a complete list but rather a representation of the wide variety of tools that have been used up to now to solve these problems.

## 9.4 New Models for Alternative CDN Architectures

Most of the models described in the previous sections are based on various CDN architectures and have their limitations due to the restrictive assumptions made to facilitate the modeling. In this section, we propose new models for more general situations that -to the best of our knowledge- have not been considered before in terms of mathematical modeling. For the purposes of illustrations, we present the models using a sample small-scale instance. The instance has a network structure consisting of a single origin server ($|S| = 1$), three active caching servers ($J = \{1,2,3\}$) and ten clients ($I = \{1,2,\ldots,10\}$). The architecture of the sample network topology is depicted in Fig. 9.1. In this sample instance, we assume that there are five objects to be distributed ($K = \{1,2,\ldots,5\}$) with their sizes (in, for instance, GBs) $b_1 = 94, b_2 = 75, b_3 = 96, b_4 = 61$ and $b_5 = 82$. The capacities of the caching servers are given as $s_1 = 156, s_2 = 162$ and $s_3 = 85$ (again, in GBs), which range from 20% to 40% of the total size of the objects. The distances $c_{ij}$ for all $i \in I, j \in J$ are randomly distributed between 1 and 5 (see Table 9.4 in the Appendix for the full matrix), whereas the distances between each caching server and the origin server are given as $c_{1,0} = 20$, $c_{2,0} = 15$ and $c_{3,0} = 18$. For simplicity, we assume that the each client has a uniform request rate for each object (i.e. $d_{ik} = 1$ for all $i \in I, k \in K$).

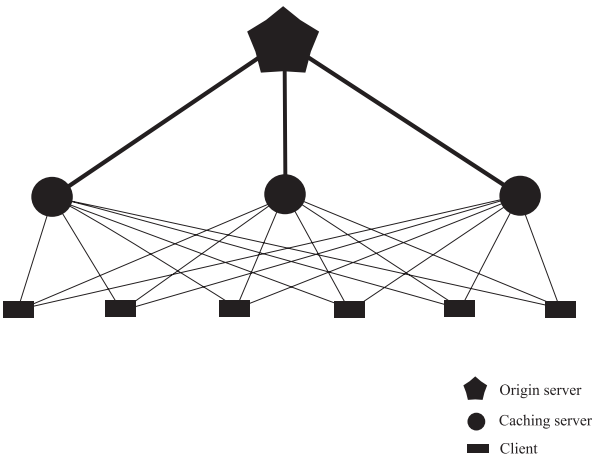



**Fig. 9.1** Architecture of the sample network topology

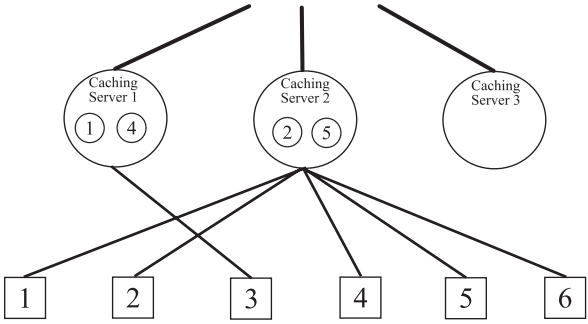



**Fig. 9.2** Solution for the sample problem obtained by model **M4**

As an initial scenario to benchmark the ones that follow, we have used model **M4** to solve the sample problem. All models in this section have been solved using the state-of-the-art nonlinear integer programming solver `BONMIN`[2] through the online NEOS server.[3] The optimal solution of **M4** is depicted in Fig. 9.2, with the client-server assignments represented by the bold links and the object placements are shown within each caching server. The total cost of this solution is 31229.

### *9.4.1 Object Retrieval from Multiple Servers*

The models presented in the previous sections generally assume a CDN architecture where a client $i \in I$ is assigned to a single caching server $j \in J$ (which will be henceforth referred to as the *primal server*) from which it retrieves the requested objects, and when the requested object is not available in server $j \in J$, the request is forwarded to the origin server by the primal server from where the object is fetched. While such a strategy may be appropriate where the number of caching servers is high and the administrative costs of requesting from other caching servers is significant, it may not always prove to be a viable option when there exists a high number of objects with similar request rates and when the storage capacities of caching servers are limited (meaning that there will be many requests forwarded to the origin server). In order to prevent this, an alternative strategy may be to direct a client's request for an object to the origin server only when the object is not available in *any* other caching server (as suggested by Datta et al. [14]). This means that each client would be allowed to retrieve objects from other caching servers. A second question arises here as to what happens when the requested object is not found in any of the caching servers. To address this, as a first step, we will restrict ourselves to the situation where a client's request would be forwarded to the origin server only via its

[2] Available at `https://projects.coin-or.org/Bonmin`

[3] Available at `neos.mcs.anl.gov/neos/solvers/minco:Bonmin/AMPL.html`

primal server, but we will also discuss when this assumption is relaxed. The model for the former case is as follows:

(**M7**) Minimize
$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k d_{ik} c_{ij} x_{ijk} + \sum_{i\in I}\sum_{k\in K}\left[\sum_{j\in J}\left(b_k d_{ik}(c_{ij}+c_{j0})x_{ij}(1-\sum_{t\in J} x_{itk})\right)\right] \quad (9.81)$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \quad (9.82)$$

$$\sum_{j\in J} x_{ijk} \leq 1 \qquad \forall i \in I, k \in K \quad (9.83)$$

$$\sum_{k\in K} b_k z_{jk} \leq s_j \qquad \forall j \in J \quad (9.84)$$

$$\sum_{j\in J} x_{ijk} \geq z_{jk} \qquad \forall i \in I, j \in J, k \in K \quad (9.85)$$

$$x_{ijk} \leq z_{jk} \qquad \forall i \in I, j \in J, k \in K \quad (9.86)$$

$$x_{ij} + z_{jk} - x_{ijk} \leq 1 \qquad \forall i \in I, j \in J, k \in K \quad (9.87)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \quad (9.88)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in I, j \in J, k \in K \quad (9.89)$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \quad (9.90)$$

In model **M7**, the objective function is composed of two cost components. The first represents the cost of serving each client directly from (multiple) caching servers. The second component, on the other hand, models the situation where a request is forwarded to the origin server by the primal server only if the object is not located at any other caching server (i.e. when $z_{jk} = 0$ for all $j \in J$ implying $\left(1 - \sum_{t\in J} x_{itk}\right) = 1$). Constraints (9.82) represent the assignment of each client to its primal server. Constraints (9.83) dictate the condition that each client receives each object from at most one caching server and constraints (9.86) make sure that the request is served only from a single caching server that holds the requested object. Storage capacity limitations for each caching server are implied by constraints (9.87). For any request, constraints (9.87) give priority that the request be served by the primal server if it holds the object (i.e. $x_{ijk} = 1$ if $x_{ij} = z_{jk} = 1$). The solution of model **M7** outputs a solution that is depicted in Fig. 9.3 with an optimal solution value of 14001, which is a solution that is about 55% less costly than that obtained by model **M4**.

In Fig. 9.3, client assignments to primal servers are represented by bold links whereas requests that are routed to other servers are represented by the lighter links (i.e. clients 1, 3, 4 and 5 are assigned to caching server 2, but they retrieve objects 1 and 4 from caching server 1, since their primal server does not hold this object). Any request for object 2 in this case has to be further requested by the origin server,

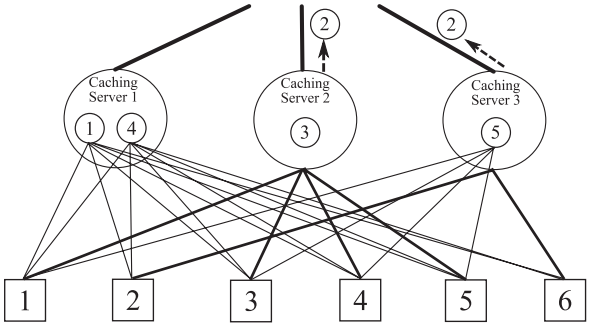



**Fig. 9.3** Solution for the sample problem obtained by model **M7**

as none of the caching servers hold this specific object. In this case, clients 1, 3, 4 and 5 receive object 2 through their primal server (no. 2), and clients 2 and 6 receive it through their primal server (no. 3).

To incorporate more flexibility into the distribution strategy, we provide below another model which allows a client's request for an object to be forwarded (and thus served to the client) by any caching server in the network. For this model, we define a new binary variable $v_{ijk}$ that equals 1 if object $k$ is served to client $i$ from the origin server *via* caching server $j$, and 0 otherwise.

$$\textbf{(M8)} \qquad \text{Minimize} \qquad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}\left(b_k d_{ik} c_{ij} x_{ijk} + b_k d_{ik}(c_{ij}+c_{j0})v_{ijk}\right)$$

subject to

$$\sum_{j\in J}(x_{ijk}+v_{ijk}) = 1 \qquad \forall i\in I, k\in K \tag{9.91}$$

$$\sum_{k\in K} b_k z_{jk} \le s_j \qquad \forall j\in J \tag{9.92}$$

$$x_{ijk} \le z_{jk} \qquad \forall i\in I, j\in J, k\in K \tag{9.93}$$

$$v_{ijk} \le 1 - z_{jk} \qquad \forall i\in I, j\in J, k\in K \tag{9.94}$$

$$x_{ijk}, v_{ijk} \in \{0,1\} \qquad \forall i\in I, j\in J, k\in K \tag{9.95}$$

$$z_{jk} \in \{0,1\} \qquad \forall j\in J, k\in K. \tag{9.96}$$

The objective function of model **M8** is composed of two components. The first represents the total cost of object transfer from the caching servers to the clients. The second represents the cost of fetching a requested object from the origin server. Constraints (9.91) ensure that a client receives any object either directly from or through one of the caching servers. Constraints (9.92) impose capacity restrictions on the caching servers, (9.93) state that a client can not be served by a caching server unless the requested object is held therein, (9.94) enforce the condition that an object can not be requested from the origin server if there exists at least one caching server $j \in J$ that holds it. The solution of model **M8** outputs a solution that is depicted in Fig. 9.4 with

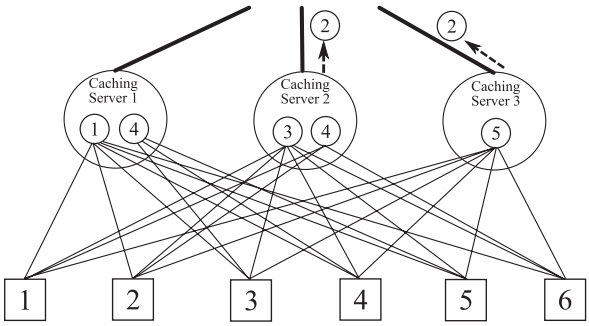



**Fig. 9.4** Solution for the sample problem obtained by model **M8**

an optimal solution value of 13940, which is even less costly than that of **M7**. The solution shown in Fig. 9.4 indeed illustrates the flexibility afforded to the distribution process where any client can receive any object from (or through) any of the servers. As an example, we note that client 1 receives object 1 from caching server 1, objects 3 and 4 from caching server 2, object 5 from caching server 3, and object 2 through caching server 2 (which further forwards this request to the origin server).

### *9.4.2 Survivability in CDN Design*

Survivability of a telecommunications network is defined as its ability to operate under a link or a server failure. As for the former case, there already exists a rather significant literature (e.g. see [25]) which can be adapted to CDN design by establishing back-up links between the clients and the servers that can be activated whenever the primal link fails. The latter case, however, is quite relevant as most of the previously stated models are based on the assumption that each client is connected to and served from or via a single caching server. In the event that its primal server should fail, the client need immediately be served by another caching server (even if the requested object is not located there since the caching server acts as a pathway to the origin server). Therefore, for a CDN to be 'survivable', one needs to design it such that each client should be assigned a *back-up* (or *stand-by*) server, to which its requests should be redirected in the event of a primal server failure. In this light, we offer here a model which extends **M7** to the survivable case. The model for this case is presented as follows:

(**M9**) Minimize $\sum_{j\in J} f_j y_j$

$$+\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k d_{ik} c_{ij} z_{jk} x^p_{ij} + b_k d_{ik}(1-z_{jk})(c_{jS}+c_{ij})x^p_{ij})$$

$$\gamma\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k d_{ik} c_{ij} z_{jk} x^p_{ij} + b_k d_{ik}(1-z_{jk})(c_{jS}+c_{ij})x^b_{ij}) \quad (9.97)$$

subject to

$$\sum_{j\in J} x_{ij}^{p} = 1 \qquad \forall i \in I \tag{9.98}$$

$$\sum_{j\in J} x_{ij}^{b} = 1 \qquad \forall i \in I \tag{9.99}$$

$$x_{ij}^{p} + x_{ij}^{b} \leq y_j \qquad \forall i \in I, j \in J \tag{9.100}$$

$$\sum_{k\in K} b_k z_{jk} \leq s_j y_j \qquad \forall j \in J \tag{9.101}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{9.102}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{9.103}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K. \tag{9.104}$$

In **M9**, $x_{ij}^{p}$ is a binary variable that is equal to 1 if server $j$ acts as a primal server for client $i$, and 0 otherwise; and $x_{ij}^{b}$ is another binary variable that is equal to 1 if caching $j$ acts as a back-up server for client $i$, and 0 otherwise. The first two components of the objective function (9.97) are similar to that of **M4**. The third component represents the cost of providing back-up service to the clients in the event of a break-down. Since the break-downs are not very likely to occur frequently, this cost will not arise very often. The parameter $0 \leq \gamma \leq 1$ is therefore provided to adjust the impact of the back-up service cost on the CDN design. Thus, when $\gamma = 0$, the CDN provider will not take into account the cost of providing back-up service to its clients, although the CDN itself will be designed in such a way. When $\gamma = 1$, then the total cost will include the additional cost of providing the back-up service, even though this service may never be used. In this model, constraints (9.98) are associated with the primal server assignments, whereas constraints (9.99) ensure that each client is also assigned to a back-up server. Constraints (9.101) impose capacity restrictions on the active caching servers. The output of the solution of model **M9** on the sample problem is depicted in Fig. 9.5. The optimal solution value in this case is 55561.7 for $\gamma = 0.7$ and 41657.3 for $\gamma = 0.3$.

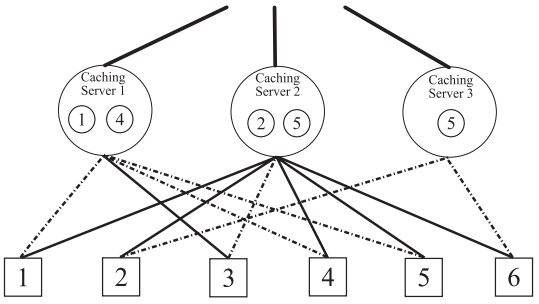



**Fig. 9.5** Solution for the sample problem obtained by model **M9**

The solution given in Fig. 9.5 shows the primal server assignments by bold links and back-up assignments by dotted links (i.e. caching server 2 acts as a primal server for client 1, but in the event that it fails, client 1 is immediately routed to caching server 1).

## 9.5 Performance Results

To give the reader a flavour of the computational performance of the new models **M7**-**M9**, we present the results of a limited set of computational experiments carried out on a set of instances. These instances have been generated in the same way as described by Bektaş et al. [9]. The instances are based on a network with three caching servers and ten clients. The number of objects to be distributed ranges from 20 to 90, in increments of 10. We note that the request rates for the objects are not uniform in this case, but have been generated using a Zipf-like distribution (see [12, 32]) in the form $P_K(i) = \Omega i^{-\alpha}$ where $\Omega = \left(\sum_{j=1}^{K} j^{-\alpha}\right)^{-1}$ is a normalization constant and the distribution parameter is set as $\alpha = 0.733$.

The results of these experiments are given in Fig. 9.6, which shows the corresponding solution values obtained with models **M4**, **M7**, **M8** and **M9** (run twice with $\gamma = 0.3$ and $\gamma = 0.7$). As the figure shows, the performance of models **M7** and **M8** are quite similar and both provide better results than that of **M4**. On the other hand, **M9** results in solutions with substantially higher costs due to the addition of the survivability component. The time required for the solution of these models are given in Table 9.3. These values imply that, even with very small-scale problems as

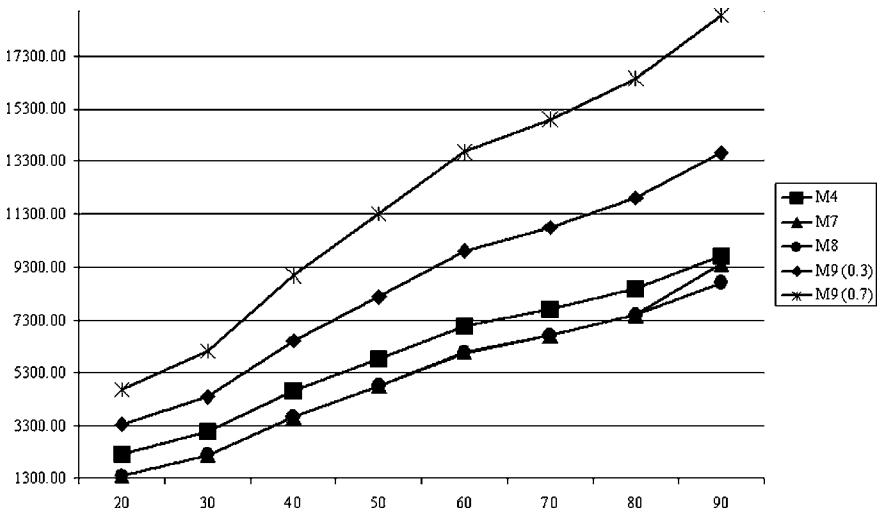



**Fig. 9.6** Cost comparison of the models on the sample instances

**Table 9.3** Computational solution times (in seconds) of the models on the sample instances

| $\|I\|$ | $\|J\|$ | $\|K\|$ | **M4** | **M7** | **M8** | **M9**(0.3) | **M9**(0.7) |
|---|---|---|---|---|---|---|---|
| 3 | 10 | 20 | 0.02 | 230.48 | 161.29 | 0.05 | 0.23 |
| 3 | 10 | 30 | 0.05 | 256.54 | 679.54 | 0.09 | 0.42 |
| 3 | 10 | 40 | 0.31 | 118.17 | 88.29 | 1.19 | 0.50 |
| 3 | 10 | 50 | 0.34 | 161.58 | 418.15 | 0.29 | 0.21 |
| 3 | 10 | 60 | 0.16 | 1764.42 | 1867.74 | 1.01 | 0.77 |
| 3 | 10 | 70 | 0.13 | 384.59 | 395.48 | 0.13 | 0.79 |
| 3 | 10 | 80 | 0.27 | 3600.00 | 3600.00 | 1.80 | 1.94 |
| 3 | 10 | 90 | 0.45 | 3600.00 | 3600.00 | 1.23 | 3.90 |

the ones considered here, solving models **M7** and **M8** prove to be quite difficult. In fact, for instances with $|K| = 80$ and $|K| = 90$, the optimal solutions of these models could not be obtained within a time limit of one hour (the values shown in Fig. 9.6 for these instances are the best possible values attained within the time limit). The other models, however, are easily solved for these instances, although the values shown in Table 9.3 imply that the solution times will expectedly increase as the size of the instances grow larger.

## 9.6 Visionary Thoughts for Practitioners

It is clear that, for a dynamic and active environment such as a CDN, most applications call for the use of fast and scalable methods among which heuristics are the popular choice. For instance, greedy heuristics (e.g. see [19]), topology-informed heuristic methods [16] or hot-spot algorithms [26] are known to be widely used for caching server location problems. However, although one may show the superiority of one heuristic method to another, one has no indication the quality of the solutions obtained with such methods. Our intention through this chapter is to stress the importance of using mathematical models and especially exact solution approaches in solving CDN problems and to recognize that there are benefits to reap in using these approaches. Indeed, mathematical modeling can be used as benchmarks to assess a variety of heuristic methods in terms of solution quality. This would certainly aid in choosing the correct type of a heuristic method in practice. Such an approach, for instance, has been taken by Laoutaris et al. [21], where the authors propose and evaluate the performance of a greedy method (and its variations) by comparing it with an exact solution approach.

Mathematical modeling techniques can also be used to gain insight to a variety of CDN problems arising in practice and to determine what mitigating actions can be taken. For instance, Nguyen et al. [23] use a Lagrangean-based solution algorithm based on a mathematical model to evaluate the effect of object clustering on the total revenue of a CDN provider using this algorithm.

Finally, we believe that the flexibility of mathematical models in easily accommodating additional constraints or the change in the problem parameters would

facilitate the analysis of a variety of scenarios and help the decision maker choose the right alternative. For instance, a CDN provider may wish to assess a number of differing request routing or object placement strategies under certain parameter settings. While one may argue that this can also be performed using heuristic methods, we believe that these may not yield as precise solutions as those which may be obtained through the use of mathematical models, since the quality of the solutions found by the former is not always known.

## 9.7 Future Research Directions

We believe that further research on CDN modeling lies in two main directions: new model realization and algorithm development. As for the former, the new models proposed here show that there are indeed situations that have not been modeled before and even hint for the possibility of developing of other models for even more complex situations that are most likely to arise in practice. Some suggestions in this respect would be to incorporate survivability issues or caching server placement decisions into models **M7** or **M8**, or the addition of QoS restrictions (such as those proposed by Bektaş et al. [8]) models **M7**-**M9**. Such attempts will undoubtedly result in more complex models, which we expect mostly to be in the form of nonlinear integer programming formulations.

As demonstrated in this chapter through numerical experiments, obtaining solutions to models such as **M7** or **M8** can prove to be quite difficult even for very small-scale instances. This further necessitates devising new exact algorithms that are able to efficiently tackle these complex models. This chapter suggests that, in terms of exact solution methods, decomposition based methods coupled with linearization strategies for the nonlinear models are a promising direction. However, these exact methods will most likely be unable to cope with large-scale instances, which further indicates the need for fast and scalable heuristic methods that can address these problems. To our belief, the development of heuristic and exact solution techniques should go hand-to-hand, in that one approach should be used as a complementary to the other. Such strategies have proven to be of good use in developing even better methodologies for some problems (e.g. see [7, 9, 15]).

## 9.8 Conclusions

In this chapter, we have outlined the fundamental problems in managing and allocating resources (the network, caching servers, and objects) in a CDN faced by the CDN provider. We have presented the existing mathematical models proposed earlier for these problems in a common framework. Discussions and examples have been provided on how several exact and heuristic methods can be tailored in solving the problems and the associated models. This chapter also offers novel mathematical

models for a variety of situations that have not yet been investigated in depth, such as designing a survivable CDN.

This chapter shows that mathematical modeling is a powerful tool to address the problems faced by the CDN provider and obtain a deeper understanding into the nature of the problem. As mentioned in the previous section, mathematical models also facilitate the solution of problems they represent, by providing a generic framework on which efficient exact solution algorithms can be devised. This chapter suggests that, in terms of exact solution algorithms, those that are based on decomposition ideas are most likely to be successful for the solution of CDN problems. Exact algorithms are also crucial in assessing the quality of heuristic approaches, especially heuristics of a greedy nature, which are known to be widely used in solving many problems of a CDN.

**Acknowledgements** Some of the materials presented in this chapter appear in a preliminary form in Computers & Operations Research Journal [8, 9].

## Appendix

The distance matrix (can be interpreted as the number of hops between each $i \in I, j \in J$) for the sample problem is given below.

**Table 9.4** The distance matrix for the sample instance

| $c_{ij}$ | $j=1$ | $j=2$ | $j=3$ |
|---|---|---|---|
| $i=1$ | 1 | 1 | 3 |
| $i=2$ | 5 | 4 | 1 |
| $i=3$ | 1 | 5 | 5 |
| $i=4$ | 1 | 4 | 5 |
| $i=5$ | 1 | 3 | 2 |
| $i=6$ | 5 | 5 | 1 |

## References

1. Almeida, J., Eager, D., Vernon, M., Wright, S.: Minimizing delivery cost in scalable streaming content distribution systems. IEEE Transactions on Multimedia **6**, 356–365 (2004)
2. Avella, P., Sassano, A., Vasil'ev, I.: Computational study of large-scale p-median problems. Mathematical Programming **109**, 89–114 (2007)
3. Backx, P., Lambrecht, T., Dhoedt, B., DeTurck, F., Demeester, P.: Optimizing content distribution through adaptive distributed caching. Computer Communications **28**, 640–653 (2005)
4. Baev, I., Rajaraman, R.: Approximation algorithms for data placement in arbitrary networks. In: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 661–670 (2001)
5. Bassali, H., Kamath, K., Hosamani, R., Gao, L.: Hierarchy-aware algorithms for CDN proxy placement in the Internet. Computer Communications **26**, 251–263 (2003)

6. Bektaş, T.: Discrete location models for content distribution. Unpublished PhD Dissertation, Bilkent University, Ankara, Turkey (2005)
7. Bektaş, T., Cordeau, J.F., Erkut, E., Laporte, G.: A two-level simulated annealing algorithm for efficient dissemination of electronic content. Journal of the Operational Research Society **35**, 3860–3884 (2008)
8. Bektaş, T., Cordeau, J.F., Erkut, E., Laporte, G.: Exact algorithms for the joint object placement and request routing problem in content distribution networks. Computers & Operations Research (2008). In press
9. Bektaş, T., Oğuz, O., Ouveysi, I.: Designing cost-effective content distribution networks. Computers & Operations Research **34**, 2436–2449 (2007)
10. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**, 238–252 (1962)
11. Berman, O., Krass, D.: An improved IP formulation for the uncapacitated facility location problem: Capitalizing on objective function structure. Annals of Operations Research **136**, 21–34 (2005)
12. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and Zipf-like distributions: evidence and implications. In: Proceedings of IEEE INFOCOM'99, Vol. 1, pp. 126–134. New York (1999)
13. Cidon, I., Kutten, S., Soffer, R.: Optimal allocation of electronic content. Computer Networks **40**, 205–218 (2002)
14. Datta, A., Dutta, K., Thomas, H., VanderMeer, D.: World Wide Wait: a study of Internet scalability and cache-based approaches to alleviate it. Management Science **49**, 1425–1444 (2003)
15. Dubuc, G., Bektaş, T., Cordeau, J.F., Laporte, G.: Une heuristique de recherche avec tabous pour la conception de réseaux de distribution de contenu électronique INFOR **45**, 175–185 (2007)
16. Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: On the placement of Internet instrumentation. In: Proceedings of IEEE INFOCOM'00, pp. 295–304 (2000)
17. Jia, X., Li, D., Hu, X., Wu, W., Du, D.: Placement of web-server proxies with consideration of read and update operations on the Internet. The Computer Journal **46**(4), 378–390 (2003)
18. Kangasharju, J., Roberts, J., Ross, K.: Object replication strategies in content distribution networks. Computer Communications **25**, 376–383 (2002)
19. Krishnan, P., Raz, D., Shavitt, Y.: The cache location problem. IEEE/ACM Transactions on Networking **8**, 568–582 (2000)
20. Laoutaris, N., Zissimopoulos, V., Stavrakakis, I.: Joint object placement and node dimensioning for Internet content distribution. Information Processing Letters **89**, 273–279 (2004)
21. Laoutaris, N., Zissimopoulos, V., Stavrakakis, I.: On the optimization of storage capacity allocation for content distribution. Computer Networks **47**, 409–428 (2005)
22. Li, B., Golin, M., Italiano, G., Deng, X., Sohraby, K.: On the optimal placement of web proxies in the Internet. In: Proceedings of IEEE INFOCOM'99, Vol. 3, pp. 1282–1290. New York (1999)
23. Nguyen, T., Safaei, F., Boustead, P., Chou, C.: Provisioning overlay distribution networks. Computer Networks **49**, 103–118 (2005)
24. Oliveira, C., Pardalos, P.: A survey of combinatorial optimization problems in multicast routing. Computers & Operations Research **32**, 1953–1981 (2005)
25. Ouveysi, I., Wirth, A., Yeh, A., Oguz, O.: Large scale linear programs and heuristics for the design of survivable telecommunication networks. Annals of Operations Research **124**, 285–293 (2003)
26. Qiu, L., Padmanabhan, V., Voelker, G.: On the placement of web server replicas. In: Proceedings of IEEE INFOCOM'01, Vol. 3, pp. 1587–1596 (2001)
27. Radoslavov, P., Govindan, R., Estrin, D.: Topology informed Internet replica placement. Computer Communications **25**, 384–392 (2002)
28. Wauters, T., Coppens, J., De Turck, F., Dhoedt, B., Demeester, P.: Replica placement in ring based content delivery networks. Computer Communications **29**, 3313–3326 (2006)

29. Xu, J., Li, B., Lee, D.: Placement problems for transparent data replication proxy services. IEEE Journal on Selected Areas in Communications **20**, 1383–1398 (2002)
30. Xuanping, Z., Weidong, W., Xiaopeng, T., Yonghu, Z.: Data Replication at Web Proxies in Content Distribution Network, *Lecture Notes in Computer Science*, Vol. 2642, pp. 560–569. Springer-Verlag, Xian (2003)
31. Yang, M., Fei, Z.: A model for replica placement in content distribution networks for multimedia applications. In: Proceedings of IEEE International Conference on Communications (ICC '03), Vol. 1, pp. 557 –561 (2003)
32. Zipf, G.: Human Behavior and the Principle of Least-Effort. Addison-Wesley, Cambridge, MA (1949)

# Chapter 10
# Performance and Availability Benefits of Global Overlay Routing

Hariharan S. Rahul, Mangesh Kasbekar, Ramesh K. Sitaraman,
and Arthur W. Berger

## 10.1 Introduction

There have been several inflection points in human history where an innovation changed every aspect of human life in a fundamental and irreversible manner. There is no doubt that we are now in the midst of a new inflection point: the Internet revolution. However, if the Internet is to realize its promise of being the next revolutionary global communication medium, we need to achieve the five grand challenges that this technology offers: perfect *availability*, high *performance*, "infinite" *scalability*, complete *security*, and last but not the least, affordable *cost*.

As the Internet was never designed to be a mission-critical communication medium, it is perhaps not surprising that it does not provide much of what we require from it today. Therefore, significant scientific and technological innovation is required to bring the Internet's potential to fruition. Content Delivery Networks (CDNs) that overlay the traditional Internet show great promise and is projected as the technology of the future for achieving these objectives.

### *10.1.1 Architecture of CDNs Revisited*

To set the context, we briefly review the evolution and architecture of commercial CDNs. A more detailed overview can be found in Chap. 1.

Hariharan S. Rahul
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139 USA,
e-mail: rahul@csail.mit.edu

Mangesh Kasbekar
Akamai Technologies, Staines, TW18 4EP, UK, e-mail: mkasbeka@akamai.com

Ramesh K. Sitaraman
Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA,
e-mail: ramesh@cs.umass.edu

Arthur W. Berger
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA and
Akamai Technologies, Cambridge, MA 02142, USA, e-mail: awberger@csail.mit.edu

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

Before the existence of CDNs, content providers typically hosted a centralized cluster of Web and streaming servers at a data center and served content to a global audience of end users (a.k.a clients). However, this solution falls significantly short of meeting the critical requirements of availability, performance, and scalability. It suffers from both the *first-mile bottleneck* of getting content from the origin servers into the Internet, and the *middle-mile bottleneck* of transporting the content across multiple long-haul networks and peering points to the access network of the client. On the first-mile, the data center itself is a single point of failure. Any connectivity problems at the data center such as an overloaded or faulty switch can result in reduced availability or even a complete outage. On the middle mile, transporting the content over the long-haul through potentially congested peering points significantly degrades both availability and performance by increasing round-trip latencies and loss. Further, there is no protection against a flash-crowd, unless the data center is grossly over-provisioned to start with.

One can alleviate some of the shortcomings of the traditional hosting solution by *multihoming* the data center where the content is hosted [3]. This is achieved by provisioning multiple links to the data center via multiple network providers and specifying routing policies to control traffic flows on the different network links. A different but complementary approach to alleviate the problems of centralized hosting is *mirroring* the content in multiple data centers located in different networks and geographies. Both of these approaches ameliorate some of the first-mile availability concerns with centralized hosting where the failure of a single data center or network can bring the Web site down. But, middle-mile degradations and scalability remain issues. Additionally, the operational cost and complexity are increased as multiple links and/or data centers must be actively managed. Further, network and server resources need to be over-provisioned, since a subset of the links and/or data centers must be able to handle the entire load in case of failures. As the quest for more availability and greater performance drive up the need for more multi-homed mirrors with larger server-farms, all of which mean more infrastructure costs, a CDN with a large shared distributed platform becomes attractive.

As we have learnt from previous chapters, a CDN is a distributed network of servers that act as an *overlay* on top of the Internet with the goal of serving content to clients with high performance, high reliability, high scalability and low cost. A highly-simplified architectural diagram of a CDN consisting of five major components is shown in Fig. 10.1.

**Edge system.** This system consist of Web, streaming, or application edge servers located close to the clients at the "edges" of the Internet. A major CDN has tens of thousands of servers situated in thousands of networks (ISPs) located in all key geographies around the world. The edge system downloads content from the origin system (Arrow 1 in Fig. 10.1), caches it when relevant, and serves it out to the clients. A more sophisticated system may also perform application processing to dynamically construct the content at the edge before delivering it to the client.

**Monitoring system.** This system monitors in real-time both the "Internet weather" and the health of all the components of the CDN, including the edge servers. Input (5) in Fig. 10.1 from the Internet cloud could consist of slow-changing

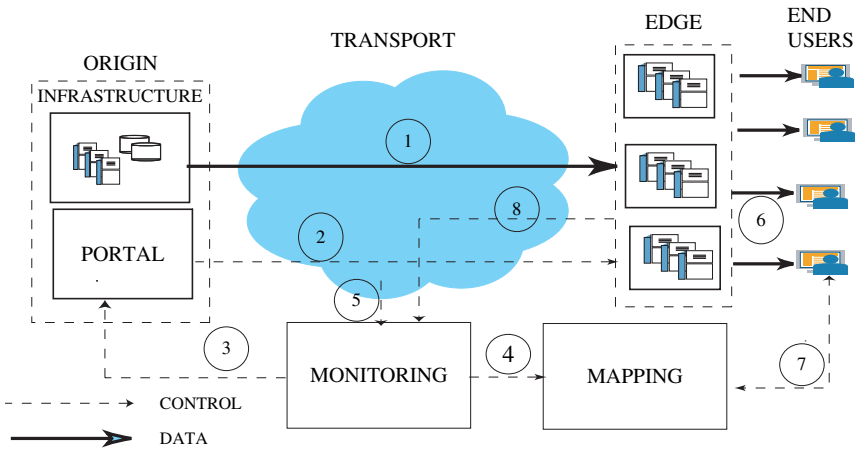



**Fig. 10.1** High-level architecture of a CDN

information such as BGP feeds from tens of thousands of networks, and fast-changing performance information collected through traceroutes and "pings" between hundreds of thousands of points in the Internet. Input (8) consists of detailed information about edge servers, routers, and other system components, including their liveness, load, and resource usage.

**Mapping system.** The job of the mapping system is to direct clients to their respective "optimal" edge servers to download the requested content (Arrow 6). The common mechanism that mapping uses to direct clients to their respective target edge servers is the Domain Name System (DNS, Arrow 7). Typically, a content provider's domain www.cp.com is aliased (i.e. CNAME'd) to a domain hosted by the CDN, such as www.cp.com.cdn.net. A name lookup by a client's nameserver of the latter domain results in the target server's ip being returned [10]. Mapping must ensure that it "maps" each client request to an "optimum" target server that possesses the following properties: (a) the target server is live and is likely to have the requested content and is capable of serving it; (b) the target server is not overloaded, where load is measured in terms of CPU, memory, disk, and network utilization; (c) the target server has good network connectivity to the client, i.e. little or no packet loss and small round-trip latencies. To make its decisions, mapping takes as input both the Internet weather and the condition of the edge servers from the monitoring system (Input 4), and an estimate of traffic generated by each nameserver on the Internet and performs a complex optimization to produce an assignment.

**Transport system.** This system is responsible for transporting data over the long-haul across the Internet. The types of content transported by the system is varied and have different Quality of Service (QoS) requirements, which makes the design of this system very challenging. For instance, transporting live streaming content from the origin (i.e. encoders) to the edge servers has a different set of requirements, as compared to transporting dynamic Web content from origin to the edge. The

challenge of course is designing a small and maintainable set of general-purpose mechanisms and abstractions that can satisfy the diverse requirements.

**Origin system.** This system originates the content that is served out to a global audience of the clients, and as such a large CDN could have tens of thousands of origin systems (one or more per content provider) that interact with the rest of the CDN. The origin Web infrastructure may include applications, databases, and Web servers. The origin infrastructure for streaming media could include large fault-tolerant replicated storage servers for storing on-demand (i.e. pre-recorded) content or equipment for video capture and encoding for live content. The origin infrastructure is usually (but not always) operated by the content provider, typically out of a single data center that is in a some cases multihomed and/or mirrored. The origin system also includes the portal operated by the CDN that is the "command center" for the content provider to provision and control their content (Arrows 2 and 3).

## *10.1.2 Transport Systems*

In this section, we review different types of transport systems and the optimizations that they perform to enhance performance. A transport system is distinguished by the end-to-end requirements of the transported content. We review some of the optimizations performed by transport systems.

### 10.1.2.1 Live Streaming

A transport system for live streaming transmits live media content from the source of the stream (encoder) to end users, so as to optimize a end user's experience of the stream (See Fig. 10.2). An *encoder* encodes the live event and sends out a sequence of encoded data packets for the duration of the live event. This data stream is first sent from the encoder to a cluster of servers called the *entry point*. It is important that the entry point can be reached from the encoder with low network latency and little

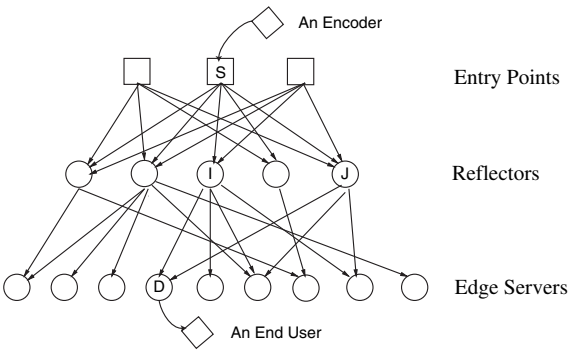



**Fig. 10.2** A transport system for live streaming

or no loss. The connectivity between the encoder and its entry point is constantly monitored, and if the connectivity degrades or if the entry point fails for any other reason, the transport system automatically diverts the stream to a different entry point that is functioning well. From the entry point, the stream is sent to one or more server clusters called *reflectors*. Each reflector, in turn, sends the data stream to one or more edge server clusters. Finally, each end user obtains the live stream from a nearby edge server using the mapping system.

The goal of the transport system is to transmit live streams in a manner that stream quality is enhanced and distortions are minimized. Distortions experienced by end users include large delays before the stream starts up, information loss leading to degraded audio and video, and freezes during playback. Each stream is sent through one or more paths adaptively by using the reflectors as intermediate nodes between the entry point and the edge server. As an example, the stream entering entry point *S* can be duplicated across one path through reflector *I* and an additional path through reflector *J* to reach edge server *D* (see Fig. 10.2). If a data packet is lost on one path, the packet may be recovered at the edge if its duplicate is received through the other path. A more sophisticated technique would be to use a coding scheme to encode the data packets, and send the encoded stream across multiple paths. Even if some packets are lost in transit, they may be recovered at the edge servers using a decoding process.

Another example of an optimization is *pre-bursting*, where the initial portion of the stream is transported to the end user at a rate higher than the encoded bit rate, so as to fill the buffer of the end user's media player quickly. This allows the media player to start the stream up quicker and also decreases the likelihood of a freeze in the middle of a playback. For more discussion of the algorithmic and architectural issues in the design of streaming transport systems, readers are referred to [6] and [12] respectively.

#### 10.1.2.2 Web and Online Applications

A transport system for the Web carries dynamically-generated content between the origin and the edge. Such content includes both dynamic Web pages downloaded by end users and user-generated content that is uploaded to a Website. A goal of such a transport system is to optimize the response times of Web transactions performed by end users. As with streaming, the transport system may use one more intermediate node to efficiently transmit information from the origin to edge. The transport system also performs several application-specific optimizations. For instance, a transport system for accelerating dynamic Web content may pre-fetch the embedded content on a Web page from the origin to the edge, so as to "hide" the communication latency between the origin and the edge.

A transport system for ip-based applications is focused on accelerating specific (non-http) application technologies such as Virtual Private Networks (VPNs) and Voice-over-IP (VOIP). The architectural issues in such systems are qualitatively different from that of the Web due to the highly-interactive real-time nature of the end user experience.

#### 10.1.2.3 Overlay Routing Schemes

A transport system uses a number of *application-specific* enhancements to meet the end-to-end requirements. For instance, as noted, transport systems use coding for loss recovery, pre-bursting for fast stream startup, and pre-fetching for fast downloads [6, 12]. These types of application-specific enhancements play a significant part of the overall performance benefit offered by the transport system. However, a fundamental benefit of *all* transport system is finding a "better path" through the Internet from the point where the content originates (origin, encoder, etc.) to the point where the content is served to the end user (edge). This purely network-level benefit is achieved through an *overlay routing scheme* that is implemented as a part of the transport system.

A generic overlay routing scheme computes one or more *overlay paths* from each source node *S* (typically the origin) to each destination node *D* (typically the edge server) such that the overlay path(s) have high availability and low latency. The overlay routing scheme typically computes overlay paths for millions of source-destination pairs using Internet measurement data. Often, the BGP-determined Internet path from a source *S* to a destination *D*, also called the *direct path*, is not the "best path" between those two nodes. This should not be surprising as the Internet protocols that select the route are largely policy-based rather than performance-based. It could well be that an *indirect path*[1] that goes from *S* to an intermediate node *I* (typically another server cluster belonging to the CDN) and then goes from *I* to *D* is faster and/or more available! An overlay routing scheme exploits this phenomenon to choose the best overlay path (direct or indirect) to route the content, thereby enhancing the end-to-end availability and performance experienced by end users. The benefits of a global overlay routing schemes is our focus for the rest of this chapter.

### *10.1.3 Our Contributions*

We present an empirical evaluation of the performance and availability benefits of global overlay routing. There has been much recent work [4, 11, 22] on improving the performance and availability of the Internet using overlay routing, but they have one of the following limitations:

- Prior work was performed on a platform hosted largely on Internet2,[2] whose capacity and usage patterns, as well as policies and goals, differ significantly from the commercial Internet.

[1] An indirect path may have more than one intermediate node if necessary.

[2] Internet2 is an advanced networking consortium consisting of several major research and educational institutions in the US. Internet2 operates an IP network that can be used for research purposes.

- Overlays used in prior work have a footprint primarily in North America. However, it is well known that network interconnectivity and relationships in Europe and Asia are different than the continental United States.

In this chapter, we present the results of the first empirical study of the performance and availability benefits of routing overlays on the commercial Internet. We use a global subset of the Akamai CDN for data collection. Specifically, we collect measurements from 1100 locations distributed across many different kinds of ISPs in 77 countries, 630 cities, and 6 continents. We address the problem of picking optimum overlay paths between the edge servers situated near end users and origin servers situated in the core of the Internet. We investigate both performance characterized by round trip latency as well as path availability. Applications such as large file downloads whose performance is more accurately characterized by throughput are not addressed in this study.

The key contributions of this chapter are the following:

- It is the first evaluation of an overlay that utilizes data from the commercial Internet. Our study provides useful cross validation for the currently deployed testbeds such as PlanetLab [18] and RON [22], and indicates that, while these deployments provide qualitatively similar data for the commercial Internet in North America, they do not capture the global diversity of network topology, especially in Asia.
- We show that randomly picking a small number of redundant paths (3 for Europe and North America, and 5 for Asia) achieves availability gains that approach the optimal. Additionally, we demonstrate that for reasonable probing intervals (say, 10 minutes) and redundancy (2 paths), over 90% of the source-destination pairs outside Asia have latency improvements within 10% of the ideal, whereas paths that originate or end in Asia require 3 paths to reach the same levels of performance.
- We provide strong evidence that overlay choices have a surprisingly high level of persistence over long periods of time (several hours), indicating that relatively infrequent network probing and measurements can provide optimal performance for almost all source-destination pairs.

### *10.1.4 Roadmap*

The rest of the chapter is organized as follows. Section 10.2 presents an overview of related work, and outlines the context of our present study. Section 10.3 describes our testbed and how the measurement data is collected. Sections 10.4 and 10.5 provide detailed metrics on the ideal performance and availability gains that can be achieved by overlays in a global context. Section 10.6 addresses issues in real overlay design, and explores structural and temporal properties of practical overlays for performance and availability. In Sects. 10.7 and 10.8, we provide directions for further research and a vision for the future.

## 10.2 Related Work

There have been many measurement studies of Internet performance and availability, for example, the work at the Cooperative Association for Internet Data Analysis (CAIDA) [7], and the National Internet Measurement Infrastructure (NIMI) [16, 17]. Examples of routing overlay networks built in academia include the Resilient Overlay Networks project at MIT [22] and the Detour project at U. Washington [11]. Commercial delivery services offered by Akamai Technologies [1] incorporate overlay routing for live streaming, dynamic Web content, and application acceleration.

Andersen et al. [5] present the implementation and performance analysis of a routing overlay called Resilient Overlay Networks (RON). They found that their overlay improved latency 51% of the time, which is comparable to the 63% we obtain for paths inside North America. Akella et al. [2] investigate how well a simpler route-control multi-homing solution compares with an overlay routing solution. Although the focus of that study is different from our current work, it includes results for a default case of a single-homed site, and the authors find that overlay routing improves performance as measured by round-trip latency by 25% on average. The experiment was run using 68 nodes located in 17 cities in the U.S., and can be compared with the 110 node, intra-North-America case in our study, where we find that the overall latency improvement is approximately 21%. However, we show that the improvement varies significantly for other continents. Savage et al. [23] used data sets of 20 to 40 nodes and found that for roughly 10% of the source-destination pairs, the best overlay path has 50% lower latency than the direct path. We obtain the comparable value of 9% of source-destination pairs for the case of intra-North America nodes, though again significantly disparate results for other continent pairs. In parallel with our evaluation, Gummadi et al. [13] implemented random one-hop source routing on PlanetLab and showed that using up to 4 randomly chosen intermediaries improves the reliability of Internet paths.

## 10.3 Experimental Setup

In this section, we describe the experimental setup for collecting data that can be used to optimize Internet paths between edge networks (where end users are located) and enterprise origin servers. End users are normally located in small lower-tier networks, while enterprise origin servers are usually hosted in tier-one networks. We consider routing overlays comprised of nodes deployed in large tier-one networks that function as intermediate nodes in an indirect path from the source (enterprise origin server) to the destination (edge server).

### *10.3.1 Measurement Platform*

The servers of the Akamai CDN are deployed in clusters in several thousand geographic and network locations. A large set of these clusters is located near the edge of the Internet, i.e. close to the end users in non-tier-one providers. A smaller set exists near the core ISPs directly located in tier-one providers, i.e. in locations that are suitable for enterprise origin servers. We chose a subset of 1100 clusters from the whole CDN for this experiment, based on geographic and network location diversity, security, and other considerations. These clusters span 6 continents, 77 countries, and 630 cities. Machines in one cluster get their connectivity from a single provider. Approximately 15% of these clusters are located at the core, and the rest are at the edge. The intermediate nodes of the overlay (used for the indirect paths) are limited to the core set. Table 10.1 shows the geographic distribution of the selected nodes. All the data collection for this work was done in complete isolation from the CDN's usual data collection activity.

**Table 10.1** Geographic distribution of the platform

| Continent (Mnemonic) | Edge Set | Core Set |
|---|---|---|
| Africa (AF) | 6 | 0 |
| Asia (AS) | 124 | 11 |
| Central America (CA) | 13 | 0 |
| Europe (EU) | 154 | 30 |
| North America (NA) | 624 | 110 |
| Oceania (OC) | 33 | 0 |
| South America (SA) | 38 | 0 |

### *10.3.2 Data Collection for Performance and Availability*

Each of the 1100 clusters ran a task that sent ICMP echo requests (pings) of size 64 bytes every 2 minutes to each node in the core set (this keeps the rate of requests at a core node to less than 10 per second). Each task lasted for 1.5 hours. If a packet was lost, specifically if no response is received within 10 seconds, then a special value was reported as the round-trip latency. Three tasks were run every day across all clusters, coinciding with peak traffic hours in East Asia, Europe, and the east coast of North America. These tasks ran for a total of 4 weeks starting 18 October, 2004. Thus, in this experiment, each path was probed 3,780 times, and the total number of probes was about 652 million. A small number of nodes in the core set became unavailable for extended periods of time due to maintenance or infrastructure changes. A filtering step was applied to the data to purge all the data for these nodes. A modified version of the standard all-pairs shortest path algorithm [9] was executed on the data set to determine the shortest paths with one, two, and three intermediate nodes from

the core set. We obtained an archive of 7-tuples <timestamp, source-id, destination-id, direct RTT, one-hop shortest RTT, two-hop shortest RTT, three-hop shortest RTT>. The archive was split into broad categories based on source and destination continents.

We consider a path to be unavailable if three or more consecutive pings are lost. Akella et al. [2] use the same definition, where the pings were sent at one minute intervals. The alternative scenario that three consecutive pings are each lost due to random congestion occurs with a probability of order $10^{-6}$, assuming independent losses in two minute epochs with a probability of order 1%. We consider the unavailability period to start when the first lost ping was sent, and to end when the last of the consecutively lost pings was sent. This is likely a conservative estimate of the length of the period, and implies that we only draw conclusions about Internet path failures of duration longer than 6 minutes.

We filtered out all measurements originating from edge nodes in China for our availability analysis. Their failure characteristics are remarkably different from all other Internet paths as a consequence of firewall policies applied by the Chinese government.

### *10.3.3 Evaluation*

We aggregate our results based on the continents of the source and destination nodes, motivated by the fact that enterprise Websites tend to specify their audience of interest in terms of their continent. The categories are denoted by obvious mnemonics such as AS-NA (indicated in Table 10.1), denoting that the edge servers are in Asia and origin servers are in North America.

## 10.4 Performance Benefits of Overlay Routing

In this section, we evaluate the performance benefits of overlay routing in the ideal situation where all possible indirect paths are considered for each source-destination pair, and the optimal indirect path is chosen in real time. Recall that our metric of performance is latency which is the round-trip time (abbreviated to RTT) from source to destination.

We compare the direct and the fastest indirect path for each source-destination pair and present the results in Table 10.2. We divide the data set into buckets based on its category and the percentage improvement in the latency of the fastest indirect path as compared to the direct path. Table 10.2 shows the percentage of source-destination pairs that fell in each of the buckets. The rows of the table sum to 100%. As an explanatory example for Table 10.2, consider the AS-AS row. The "$< -10\%$" bucket shows the cases where the best indirect paths are at least 10% slower than the direct path. 15.5% of the AS-AS paths fell in this bucket. The "$\pm 10\%$" bucket

**Table 10.2** Histogram of latency reduction percentages

| Category | $< -10\%$ (Slower) | $\pm 10\%$ (Comparable) | 10–30% (Marginal) | 30–50% (Significant) | $> 50\%$ (Large) |
|---|---|---|---|---|---|
| AF-AS | 4.0 | 44.5 | 44.2 | 5.7 | 1.6 |
| AF-EU | 0.6 | 69.3 | 18.1 | 9.7 | 2.3 |
| AF-NA | 0.0 | 74.2 | 21.6 | 3.5 | 0.6 |
| AS-AS | 15.5 | 24.7 | 23.4 | 13.2 | 23.2 |
| AS-EU | 0.9 | 33.9 | 45.5 | 12.5 | 7.2 |
| AS-NA | 0.1 | 43.2 | 42.4 | 7.6 | 6.7 |
| CA-AS | 0.0 | 40.5 | 53.5 | 4.6 | 1.4 |
| CA-EU | 1.4 | 53.2 | 42.3 | 2.5 | 0.7 |
| CA-NA | 1.7 | 44.1 | 41.3 | 11.2 | 1.8 |
| EU-AS | 0.6 | 24.5 | 63.8 | 7.8 | 3.2 |
| EU-EU | 10.5 | 36.4 | 30.5 | 12.6 | 10.0 |
| EU-NA | 0.0 | 50.6 | 45.1 | 3.3 | 0.9 |
| NA-AS | 0.0 | 34.0 | 57.9 | 5.4 | 2.6 |
| NA-EU | 0.1 | 43.1 | 51.1 | 4.4 | 1.4 |
| NA-NA | 2.4 | 34.7 | 39.0 | 15.0 | 9.0 |
| OC-AS | 6.1 | 38.9 | 18.9 | 22.9 | 13.2 |
| OC-EU | 0.0 | 60.4 | 35.1 | 3.9 | 0.7 |
| OC-NA | 0.0 | 66.7 | 25.6 | 6.3 | 1.4 |
| SA-AS | 0.1 | 43.1 | 47.9 | 5.5 | 3.4 |
| SA-EU | 0.4 | 66.1 | 28.9 | 2.3 | 2.2 |
| SA-NA | 0.9 | 55.1 | 35.1 | 5.7 | 3.3 |

represents the cases where the best indirect path and the direct path are comparable, in the sense that their latencies are within 10% of each other. 24.7% of the paths in the AS-AS category fell in this bucket. Out of the remaining direct paths, 23.4% saw a marginal (10–30%) improvement, 13.2% of the paths saw significant (30–50%) improvements, and 23.2% of the paths saw large latency reductions of a factor of two or better from the indirect paths found by the overlay.

Overall, about 4%–35% of all source-destination pairs see improvements of over 30% latency, depending on the category. Additionally, high numbers of source-destination pairs see over 50% improvement for the AS-AS and EU-EU categories, which indicates the presence of many cases of pathological routing between ISPs in these continents. A nontrivial number of AS-AS paths are routed through peering locations in California, for example, the path between Gigamedia, Taipei and China Telecom, Shanghai. All the traceroutes in our snapshot that originated at Gigamedia, Taipei and ended at other locations in Asia went via California, except the path to China Telecom, Shanghai, which went directly from Taipei to Shanghai. The Taipei-Shanghai path thus sees little or no improvement with an overlay, since all the alternatives are very convoluted. At the same time, all the paths that originate in Gigamedia, Taipei and end in other locations in Asia see *large* improvements, since their direct routes are very convoluted, but there exists a path via China Telecom, Shanghai, which is more than 50% faster.

### *10.4.1 Source-Destination Pairs with Poor Connectivity*

Enterprises are particularly interested in enhancing the worst-case performance of their Website, by speeding up the clients who see the worst performance. Therefore, the benefits provided by overlay routing in minimizing the worst path latencies in each category are especially interesting. We compare the latency reduction enjoyed by a "typical" source-destination pair in a given category with that of a "poorly connected" source-destination pair in the same category. We bucketed the data set for each category into 10 millisecond buckets based on the latency of the direct path. We then looked at the 50th percentile bucket ("typical" source-destination pairs) and the 90th percentile bucket ("poorly-connected" source-destination pairs). For each of these buckets, we determined the average improvements provided by the fastest indirect path over the direct path. Table 10.3 shows the comparison of the benefits seen by the typical and the poorly-connected source-destination pairs in each category. For the typical source-destination pairs, the latency reduction exceeds 20% only for AS-AS, OC-AS and CA-NA out of the 21 categories. Comparatively, the poorly-connected source-destination pairs see a benefit over 20% for half of the categories. The important categories of AS-AS, AS-NA, and EU-EU show significant improvements for the poor source-destination pairs, while, in contrast for paths originating from Africa the latency for 90th percentile bucket is both high and not

**Table 10.3** Latency reduction for typical and poorly-connected source-destination pairs

| Category | 50th Percentile | | | 90th Percentile | | |
|---|---|---|---|---|---|---|
| | Direct (ms) | Fastest (ms) | Reduction (%) | Direct (ms) | Fastest (ms) | Reduction (%) |
| AF-AS | 350 | 290 | 17 | 740 | 700 | 5 |
| AF-EU | 150 | 120 | 20 | 620 | 620 | 0 |
| AF-NA | 200 | 180 | 10 | 560 | 550 | 2 |
| AS-AS | 230 | 110 | 52 | 590 | 350 | 41 |
| AS-EU | 320 | 260 | 19 | 500 | 360 | 28 |
| AS-NA | 230 | 200 | 13 | 470 | 280 | 40 |
| CA-AS | 230 | 200 | 13 | 300 | 250 | 17 |
| CA-EU | 160 | 140 | 12 | 200 | 170 | 15 |
| CA-NA | 90 | 70 | 22 | 130 | 100 | 23 |
| EU-AS | 300 | 260 | 13 | 390 | 300 | 23 |
| EU-EU | 30 | 30 | 0 | 80 | 60 | 25 |
| EU-NA | 130 | 120 | 8 | 190 | 160 | 16 |
| NA-AS | 190 | 160 | 16 | 260 | 210 | 19 |
| NA-EU | 130 | 110 | 15 | 180 | 150 | 17 |
| NA-NA | 50 | 40 | 20 | 90 | 70 | 22 |
| OC-AS | 200 | 140 | 30 | 340 | 220 | 35 |
| OC-EU | 330 | 300 | 9 | 400 | 330 | 17 |
| OC-NA | 220 | 200 | 9 | 280 | 230 | 18 |
| SA-AS | 320 | 280 | 12 | 470 | 340 | 28 |
| SA-EU | 230 | 210 | 9 | 290 | 250 | 14 |
| SA-NA | 160 | 150 | 6 | 240 | 190 | 21 |

helped with the overlay. For the AS-AS category, both the typical and poor source-destination pairs see significant improvement via the overlay, but the improvement are even greater for the typical paths. However, in general we can conclude that poorly-connected source-destination pairs benefit more from overlay routing, compared to a typical source-destination pair.

Next, we provide a more in-depth evaluation of what fraction of the poorly-connected source-destination pairs derive marginal, significant, or a large benefit from overlay routing. We bucket all the source-destination pairs in a given category whose direct path latency ever exceeded the 90th percentile latency of that category as shown in Table 10.3 to derive the histogram of the latency reduction for poorly-connected source-destination pairs. This histogram of the latency reduction for poorly-connected source-destination pairs is shown along side the same values for all source-destination pairs in that category in Fig. 10.3. (Note that the data charted in Fig. 10.3 for all source-destination pairs was presented in the last three columns of Table 10.2). Poorly-connected source-destination pairs see at least marginal benefits in over 80% of the samples, while 67% of the samples see significant or large benefits. Some categories do deviate from this observation in the figure. For example, even poorly-connected source-destination pairs with destinations in Africa do not derive much help from an overlay.

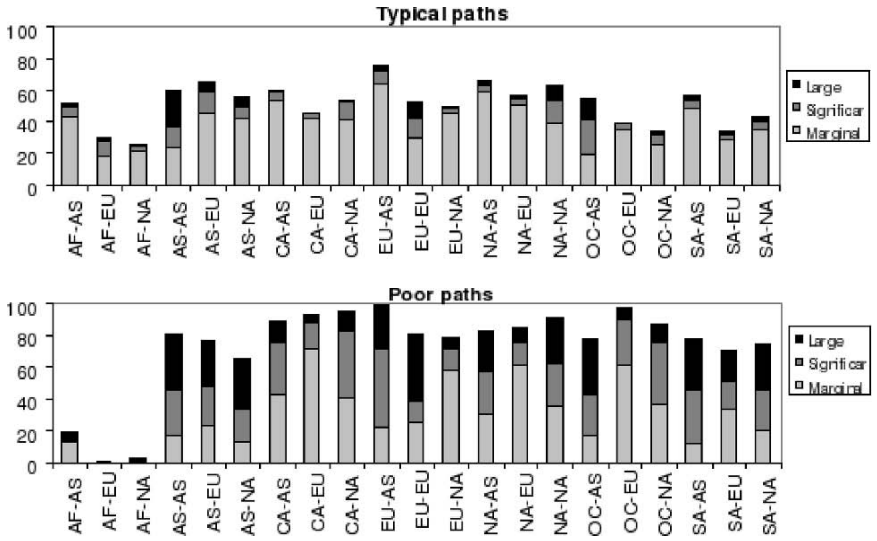



**Fig. 10.3** Latency reduction for all and poorly-connected source-destination pairs

## 10.5 Availability Gains of Overlays

In this section, we evaluate the availability benefits of overlay routing in the ideal situation, where all possible indirect paths are considered for each source-destination

pair, and when possible an indirect path that is available is chosen in real time to mitigate failures.

We study how often the direct path from each source-destination pair fails, and during these failures what percentage of times at least one indirect path was functional. This provides a best-case estimate of the availability gains that overlay routing can provide. Figure 10.4 shows the percentage samples where the direct path between the source and destination failed for each category. The failure percentage of the direct paths ranges from 0.03% to 0.83%. Asia has the poorest availability: nine of the ten categories with the largest failure percent have an endpoint in Asia. In the presence of overlay routing, the failure percent goes down by 0.3–0.5% for most categories, indicating that the indirect paths help mask failures of the direct path. In fact, the high-failure categories involving Asia show dramatic availability improvements.

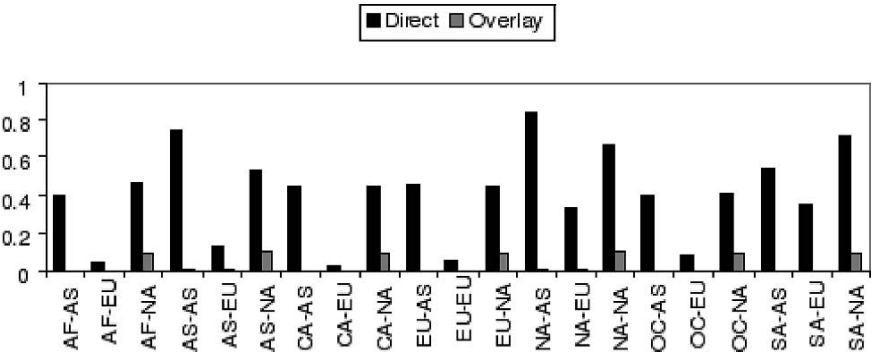



**Fig. 10.4** Reduction in failure percentages with overlay routing

### *10.5.1 Source-Destination Pairs with Poor Connectivity*

As with Sect. 10.4.1, we study how overlay routing benefits source-destination pairs with direct paths that exhibit the most failures. Again, this is of great interest to enterprises that are typically interested in using CDNs to enhance the availability of their least available end users and clients. It is commonly understood that a small number of paths contribute to a large number of path failures on the Internet. As evaluated in [15], 3% of Internet paths give rise to 30% of failures. We identified a similar pattern in our data as shown in Table 10.4. We see that about 3% of the direct paths caused 30% of the failures, and that 10% of the direct paths gave rise to 50% of the failures.

We identified the least-available source-destination pairs in each category that cumulatively gave rise to 30% of the failures, and re-ran the availability analysis for only these source-destination pairs. The results are shown in Table 10.4. A failure rate higher than 20% for direct paths for a source-destination pair is indicative of some specific chronic trouble, rather than random, transient failures or short-lived

**Table 10.4** Availability statistics for poor paths

| Category | % paths with 30% Failures | Failure % no Overlay | Failure % Overlay |
|---|---|---|---|
| AF-AS | 4.5 | 25.8 | 0 |
| AF-EU | 1.7 | 8.8 | 0 |
| AF-NA | 0.6 | 36.2 | 0 |
| AS-AS | 2.7 | 31.4 | 0 |
| AS-EU | 1.5 | 9.8 | 0 |
| AS-NA | 0.4 | 30 | 0 |
| CA-AS | 3.5 | 28.2 | 0 |
| CA-EU | 1.6 | 10.9 | 0 |
| CA-NA | 0.5 | 30.3 | 0 |
| EU-AS | 3 | 30.1 | 0 |
| EU-EU | 0.9 | 10.8 | 0 |
| EU-NA | 0.4 | 30.1 | 0 |
| NA-AS | 2.7 | 32.3 | 0 |
| NA-EU | 0.4 | 13.2 | 0 |
| NA-NA | 0.2 | 40.2 | 0 |
| OC-AS | 3.1 | 30.8 | 0 |
| OC-EU | 1.4 | 10.7 | 0 |
| OC-NA | 0.4 | 29.3 | 0 |
| SA-AS | 3.3 | 28.8 | 0 |
| SA-EU | 2.2 | 9.5 | 0 |
| SA-NA | 0.8 | 23 | 0 |

congestion. Almost all these source-destination pairs with a chronic availability problem saw perfect availability with overlay routing! Enhancing the availability of the least available origin-destination pairs is a key benefit of overlay routing.

## 10.6 Achieving the Benefits in a Practical Design

The analysis presented in Sects. 10.4 and 10.5 characterizes an ideal case where network measurements are used in the computation of indirect paths in real-time. In addition, we assumed that an unlimited number of indirect paths can be probed and utilized as indirect routes. Therefore, this analysis is a best-case estimate on the performance and availability gains that can be expected from overlay routing. However, in a practical system, measurements made at a given time $t$ is used for constructing overlay paths that are utilized by the transport system till some time $t + \tau$ into future. And, only a small number of indirect paths can be constructed and used at any given time for a given source-destination pair (call the number of paths $\kappa$). This section incorporates these practical considerations into the analysis and evaluates its impact on the results. As $\kappa$ increases and $\tau$ decreases, the cost of constructing the overlay paths goes up but one would expect the quality of constructed overlay paths to increase and approach the best-case routes constructed in Sects. 10.4 and 10.5.

First, we evaluate a simple multi-path memoryless overlay routing scheme that *randomly* selects a subset of $\kappa$ paths, purely based on static information and uses it to route content. It is natural to expect that this overlay will likely be inferior to the ideal, but our goal is to develop a straw man to validate the importance of intelligence and adaptiveness in overlay path selection. Surprisingly, we found that random selection is successful in providing near optimal availability for $\kappa = 3$, substantiating the fact that the Internet offers very good path diversity, and generally has low rates of failure. The policy, however, fails in improving performance, suggesting that careful path selection is very important in building overlays for performance gains. Such performance-optimizing overlay routing schemes are the focus of the rest of this section.

### *10.6.1 Stability of Optimal Paths*

To the extent that a performance-optimizing overlay routing scheme selects a subset of paths to use, it will deviate from optimality as a result of variations in path latencies over time that cause a reordering of the best paths. Source-destination pairs tend to fall into two categories:

1. The best paths from the source to the destination are quite persistent, and do not change, regardless of variations in the latencies of all paths between them.
2. Latency variations of the paths over time cause a significant reordering of the best paths between source and destination, which in turn causes changes in the optimal paths.

Source-destination pairs in the first category do not require a very dynamic overlay design for selecting indirect paths for performance improvement. For example, consider the path from Pacific Internet, Singapore to AboveNet, London. The direct path, which hops from Singapore through Tokyo, San Francisco, Dallas, and Washington D.C. to London takes approximately 340 millisecond. However, there exists an indirect path through an intermediate node in the ISP Energis Communications in London. The path between Pacific Internet, Singapore and Energis, London is one hop long (possibly a satellite link), and has a latency of 196 millisecond. The subsequent traversal from Energis, London to AboveNet, London takes just 2 millisecond. The indirect path is therefore faster than the direct path by over 140 millisecond, or 41.2%. While the latencies vary, the ordering of the paths seldom change.

For source-destination pairs in the second category, latency variations are more important. We systematically examine the extent of the latency variation across paths by computing a statistic called *churn* that measures the extent to which sets of best $\kappa$ paths at two different time instants vary. Formally, for a given pair of nodes,

$$Churn_t(\kappa,\tau) \stackrel{\Delta}{=} |S(\kappa,t) - S(\kappa,t+\tau)|/\kappa,$$

where $S(\kappa,t)$ is the set of the $\kappa$ best performing paths between those nodes at time $t$. $Churn(\kappa,\tau)$ for a node pair is then computed as an average of $Churn_t(\kappa,\tau)$ over all valid values of $t$. $Churn(\kappa,\tau)$ is a number between 0 and 1, that is 0 for paths with a persistent set of best paths, and tend to be closer to 1 for paths with a fast changing set of best paths. We found that the majority of source-destination pairs have values of $Churn(\kappa,\tau)$ larger than 10%, even when selecting up to $\kappa = 5$ best performing paths and using this prediction for only $\tau = 2$ minutes into the future.

To examine path churn more closely, one can define a relaxed measure called $RelaxChurn(\kappa,\tau)$ that counts only paths $\pi \in S(\kappa,t) - S(\kappa,t+\tau)$ whose latency at $t+\tau$ is higher than 110% of the latency of the path with the worst latency in $S(\kappa,t+\tau)$, i.e. keeping path $\pi$ would worsen the performance at time $t+\tau$ by more than 10%. Interestingly, $RelaxChurn(\kappa,\tau)$ is less than 10% on average for over 80% of source-destination pairs in most categories. This indicates that a path selection algorithm that makes predictions into the future based on current measurements, can achieve performance close to the ideal.

Figure 10.5 shows the percentage of source-destination pairs that have *Churn* $(\kappa,\tau)$ and $RelaxChurn(\kappa,\tau)$ of less than 10% for $\kappa = 1$ and $\tau = 2$ minutes. Note that paths with both the end points in Asia do have a higher value of *RelaxChurn* than *Churn*, but still only 63% AS-AS source-destination pairs have low-churn paths. Thus, potentially higher performance benefits for AS-AS paths are likely only obtainable at a higher cost in terms of network measurement.

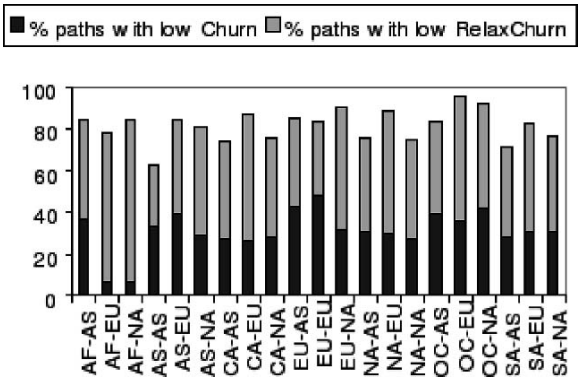



**Fig. 10.5** Percentage of source-destination pairs with low *Churn* and *RelaxChurn* for $\tau = 2$ minutes and $\kappa = 1$

## *10.6.2 Performance Gains of a Predictive Overlay*

The analysis in Sect. 10.6.1 examined stability using purely structural properties. In this section, we compare the performance of overlay routing with parameters $\kappa$ and $\tau$ with the performance of the ideal case where the optimal path is always chosen. Note that this measure holds overlays to a higher standard, as the optimal path at a given time is at least as fast as the direct path.

**Table 10.5** Percentage of paths within 10% of the optimal latency

| Category | Percentage of Paths | | | |
|---|---|---|---|---|
| | $\kappa = 1$ $\tau = 2$ | $\kappa = 1$ $\tau = 10$ | $\kappa = 2$ $\tau = 2$ | $\kappa = 3$ $\tau = 2$ |
| AS-AS | 62.4 | 59.5 | 84.6 | 89.4 |
| AS-EU | 76.2 | 74.1 | 92.2 | 94.5 |
| AS-NA | 74.8 | 71.6 | 94.0 | 96.0 |
| EU-AS | 74.4 | 72.3 | 88.4 | 92.8 |
| EU-EU | 80.1 | 78.1 | 91.6 | 93.1 |
| EU-NA | 83.0 | 82.2 | 94.7 | 96.2 |
| NA-AS | 68.1 | 66.2 | 88.8 | 93.7 |
| NA-EU | 82.3 | 81.3 | 95.4 | 97.2 |
| NA-NA | 71.6 | 69.6 | 92.0 | 95.0 |

A natural case to examine in some detail would be $\kappa = 1$. This corresponds to just using the best path choice in future iterations. Table 10.5 in the second and third columns shows our results for $\tau = 2$ and 10 minutes. As an explanatory example, consider the NA-NA category. The table shows that when using $\tau = 2$ minutes, 71.6% of the paths came within 10% of the optimal latency for that observation. Even when using stale data, with $\tau = 10$ minutes, 69.6% of the paths managed to achieve the same result. Paths originating in Asia again show a greater deviation from optimality than paths originating in Europe, whereas paths originating in North America span the full range of deviations.

Given that the performance gains with $\kappa = 1$ do not seem adequate everywhere, we then explored higher values of $\kappa$. As an explanatory example, consider the category NA-EU. The table shows that 82.3% of the paths came within 10% of the optimal when choosing $\kappa = 1$. Increasing $\kappa$ to 2 enables approximately 13.1% more paths to achieve the same result. Increasing $\kappa$ to 3 provides only a marginal benefit for the remaining paths, and only 1.8% more paths achieved the result with this value of $\kappa$. From Table 10.5, we immediately see that choosing $\kappa = 2$ provides disproportionately high gains over choosing $\kappa = 1$, and the marginal benefit of choosing $\kappa = 3$ is much lower. In fact, apart from paths with their destination in Asia, over 90% of all source-destination pairs are within 10% of the ideal performance when selecting $\kappa = 2$, and this fact remains true even with increasing $\tau$. The results also suggest that an overlay routing scheme where either $\kappa = 1$ or 2 paths are used would work well. For example, 95.4% of all NA-EU source-destination pairs are within 10% of optimal for overlays with $\kappa = 2$. Combining this with the fact that 82.3% of these pairs require only one choice to come within the same limits, it is conceivable that an overlay routing scheme could potentially use two paths only for the excess 13.1% of pairs, for an average overhead of just 1.09 paths per pair.

Source-destination pairs where both are in Asia show a different behavior. For example, the proportion of AS-AS source-destination pairs within 10% of optimal jumps from 62.44% to 84.57% when going from $\kappa = 1$ to $\kappa = 2$ (for a weighted average set size of 1.31). However, achieving within 10% of optimal for close to 90% of the source-destination pairs requires $\kappa = 3$.

Note that although Table 10.5 shows results for $\tau = 2$ minutes for $\kappa = 2$, these values remain relatively stable for higher values of $\tau$ between 2 and 10 minutes (similar to the case of $\kappa = 1$). This implies that increasing the rate of probing does not lead to gains in latency for a significantly higher number of paths. We expand on the sensitivity of the results to $\tau$ in Sect. 10.6.3.

Interestingly, overlays designed for high performance show reduced availability as compared to the ideal situation. This is because, as illustrated in earlier examples in this chapter, better performing paths are typically constrained to share a small set of common links, leading to less path diversity and a greater vulnerability that all these shared links will simultaneously fail.

### *10.6.3 Persistence*

The analysis in Sect. 10.6.2 indicates that the benefits of overlays are only mildly sensitive to the value of $\tau$, at least in the range of 2–10 minutes. In this section, we explore the time sensitivity of predictive overlays by using some extreme cases. Our daily 1.5 hour samples are separated by a gap of 4 to 11 hours. We used overlays based on measurements in one 1.5 hour sample, and evaluated their performance on the next sample. While it is entirely possible that the overlay might have been suboptimal in the intervening time period, we see that around 87% of NA-NA, and 74% of AS-AS paths are within 10% of ideal even with these long term predictions. These statistics point to a high degree of consistency in the relative performance of alternative paths between a source-destination pair, for most pairs. In contrast, there is a small number of paths [20] with high short term variations, and it is difficult for a predictive overlay to optimize these paths even with $\kappa$ going up to 5 or 6.

## 10.7 Future Research Directions

In this chapter, we quantified the performance and availability benefits achievable by overlay routing, and how it differs from continent to continent. The inefficiencies of the Internet have deep roots in economic considerations of the individual ISPs and are here to stay for a long time. Further, the significant geographical variations in behavior may well be artifacts of a deeper structural nature, and are not expected to even out over time as connectivity and economies improve. These facts point to a continued rapid growth in high-value traffic routed by overlay networks of CDNs. As overlay routing optimizations become more and more prevalent, the impact of these optimizations on individual ISPs operating the "underlay" and the optimizations they perform within their own networks become an interesting topic of future study [8, 14, 19].

## 10.8 Visionary Thoughts for Practitioners

After a decade of evolution, there is no doubt that CDNs now play a central role in enabling business on the Internet. Businesses in every vertical, including technology, media, entertainment, commerce, software, and government, have adopted CDN technology. The traffic hosted on CDNs continue grow by leaps and bounds, year after year. The dual challenges of enhancing the performance and availability of web sites, streaming media and applications has been a fundamental driving force of CDN evolution over the past decade. We end the chapter by refocusing our vision on those challenges and the road ahead.

- Consider that there are now retailers selling billions of dollars of goods on the Internet for whom even a 10-minute downtime of their Website during a peak period translates to millions of dollars of lost revenue and can also result in poor user perception [24]. Further, e-commerce revenue is growing at a significant rate and is expected to double every two to three years! In addition, there is growing evidence that fast downloads of Web pages are linked to larger conversion rates at e-commerce sites, leading to greater revenue. *We need to deliver content on the Internet to provide ever higher levels performance with little or no downtime.*
- Consider that there are large media and entertainment companies who rely on the Internet to disseminate content to vast numbers of end users. While they like the on-demand and ubiquitous nature of Internet streaming, they want a true television-like experience, where the video starts up immediately and never freezes! *We need to deliver content on the Internet with higher performance than traditional methods.*
- As the Internet becomes more and more entrenched as a primary source of entertainment and news, a number of content providers face the so-called flash crowd problem. *We need to deliver content on the Internet in a scalable fashion to end users even during a flash crowd, without loss of availability or performance.*
- New business trends such as outsourcing and workforce consolidation, as well as government communications necessitate exacting performance and availability standards, not just within a single country or small group of countries, but globally. It is becoming more common to have large virtual teams with individuals across the world collaborating in real-time on a single project via the Internet. Further, many novel Internet applications have more stringent performance requirements than ever. Interactive applications, such as remote shells over virtual private networks (VPNs) and multi-user games, and emerging technologies such as voice over IP (VoIP) are highly latency sensitive. *We need to meet novel and more stringent availability and performance requirements to support the next-generation of Internet applications.*

These challenges will continue to drive the field forward and shape the future CDN in the coming years.

**Acknowledgements** The experimental results presented in this chapter appeared as a technical report in [20] and as a conference paper in [21]. Ramesh Sitaraman was supported in part by NSF Award CNS-0519894.

## References

1. Akamai Technologies, Inc. http://www.akamai.com.
2. Akella, A., Pang, J., Maggs, B., Seshan, S., and Shaikh, A. A comparison of overlay routing and multihoming route control. In *Proceedings of the ACM SIGCOMM*, pp. 93–106, Portland, OR, August 2004.
3. Akella, A., Maggs, B., Seshan, S., Shaikh, A., and Sitaraman, R. A Measurement-Based Analysis of Multihoming. *Proceedings of the 2003 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, August 2003.
4. Andersen, D. G. *Improving End-to-End Availability Using Overlay Networks*. PhD thesis, MIT, 2005.
5. Andersen, D. G., Balakrishnan, H., Kaashoek, F., and Morris, R. Resilient Overlay Networks. In *18th ACM SOSP*, Banff, Canada, October 2001.
6. Andreev, K., Maggs, B., Meyerson, A., and Sitaraman, R. Designing Overlay Multicast Networks for Streaming. *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA),* June 2003.
7. Home page of the Cooperative Association for Internet Data Analysis (CAIDA). http://www.caida.org.
8. Clark, D., Lehr, B., Bauer, S., Faratin, P., Sami, R., and Wroclawski, J. The Growth of Internet Overlay Networks: Implications for Architecture, Industry Structure and Policy. In *33rd Research Conference on Communication, Information and Internet Policy*, Arlington, Virginia, September 2005.
9. Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. Introduction to Algorithms. MIT Press and McGraw-Hill, 2001.
10. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing,* September 2002, pp. 50–58.
11. Home page of the Detour Project http://www.cs.washington.edu/research/networking/detour/.
12. Kontothanassis, L., Sitaraman, R., Wein, J., Hong, D., Kleinberg, R., Mancuso, B., Shaw, D., and, Stodolsky, D. "A Transport Layer for Live Streaming in a Content Delivery Network". *Proceedings of the IEEE, Special issue on evolution of internet technologies,* pp. 1408–1419, Vol. 92, Issue 9, August 2004.
13. Gummadi, K. P., Madhyastha, H., Gribble, S., Levy, H., and Wetherall, D. Improving the Reliability of Internet Paths with One-hop Source Routing. In *Symposium on Operating System Design and Implementation (OSDI)*, San Diego, CA, 2003.
14. Keralapura, R., Taft, N., Chuah, C., and Iannaccone, G. Can ISPs take the heat from overlay networks? In *ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2004.
15. Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., and Diot, C. Characterization of failures in an ip backbone. In *IEEE Infocom*, Hong Kong, 2004.
16. Home page of the National Internet Measurement Infrastructure (NIMI). http://ncne.nlanr.net/nimi/.
17. Paxson, V., Mahdavi, J., Adams, A., and Mathis, M. An Architecture for Large-Scale Internet Measurement. *IEEE Communications*, August 1998.
18. Home page of PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org/.
19. Qiu, L., Yang, Y. R., Zhang, Y., and Shenker, S. On Selfish Routing in Internet-Like Environments. In *ACM SIGCOMM*, 2003.

20. Rahul, H., Kasbekar, M., Sitaraman, R., and Berger, A. Towards Realizing the Performance and Availability Benefits of a Global Overlay Network. *MIT CSAIL TR 2005-070*, December 2005.
21. Rahul, H., Kasbekar, M., Sitaraman, R., and Berger, A. Towards Realizing the Performance and Availability Benefits of a Global Overlay Network. *Passive and Active Measurement Conference*, Adelaide, Australia, March, 2006.
22. Home page of the Resilient Overlay Networks (RON) project. http://nms.csail.mit.edu/ron/.
23. Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. The End-to-End Effects of Internet Path Selection. In *Proc. ACM SIGCOMM*, pp. 289–299, Cambridge, MA, 1999.
24. Zona Research. The need for speed II. Zona Market Bulletin 5 (April 2001).

# Part III
# Advanced CDN Platforms and Applications

# Chapter 11
# Dynamic CDN Against Flash Crowds

Norihiko Yoshida

## 11.1 Introduction

With the rapid spread of information and ubiquitous accesses of browsers, new congestion phenomena on the Internet, *flash crowds*, makes traditional techniques fail to solve. Flash crowds are sudden, unanticipated surges in traffic volume of request rates towards particular Web sites. Differing from the consistent Internet congestions, flash crowds produce short-term congestions. It makes Web sites over-provisioned and the hosting Content Delivery Network (CDN) inefficient and uneconomical. Thus, they pose new challenges to the today's Internet.

The term "flash crowd" was coined in 1973 by a science fiction writer Larry Niven in his short novel "Flash Crowd" [31]. In the novel, cheap and easy teleportation enabled tens of thousands of people worldwide to flock to the scene of anything interesting almost instantly, incurring disorder and confusion.

The term was then applied to similar phenomena on the Internet in the late 1990's. When a Web site catches the attention of a large number of people, it gets an unexpected and overwhelming surge in traffic, usually causing network saturation and server malfunction, and consequently making the site temporarily unreachable. This is the "flash crowd" phenomenon on the Internet, which is also sometimes referred to as the "SlashDot effect" [2] or a "Web hotspot" [50]. An example of a flash crowd is shown in Fig. 11.1 [34], which occurred on the "LIVE! ECLIPSE" Web site [26] on November 3rd, 2005.

Flash Crowds are not frequent phenomena. They differ from those workloads that vary over time, such as time-of-day effects [13], e.g. more people enjoy the Web during lunch hours, where long-term periodic trends can be predicted. However, they are triggered relatively easily, in that even the mere mention of a popular Web site will produce one. Due to an increase in the frequency of flash crowds and their overall unpredictability, flash crowds have now become a bane of most Web sites.

A conventional CDN works well if the request load is relatively constant. However, it is static in the sense that it uses a fixed number of surrogates all the time,

Norihiko Yoshida
Division of Mathematics, Electronics and Informatics, Saitama University, Saitama 338-8570, Japan, e-mail: yoshida@mail.saitama-u.ac.jp

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

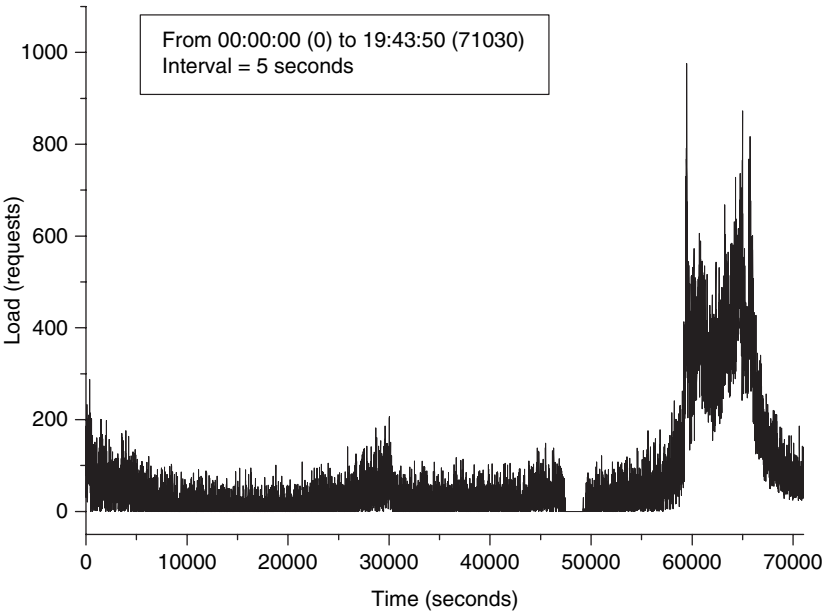



**Fig. 11.1** Flash crowd to the "LIVE! ECLIPSE" site on Nov. 3, 2005

and it is permanently prepared for the congested state. Considering the situation of flash crowds, resorting to a high level of over-provision suffers from low efficiency. Due to the infrequency of the high load, static CDNs lead to under-utilization of resources, and the surrogate servers will remain idle most of the time.

Moreover, Web requests can be bursty. It is not easy to predict the peak load of a flash crowd. Even a very well-configured CDN site may be crippled due to the demand unpredictability associated with a traffic surge.

Some solutions have already been proposed to address the problem of flash crowds. Promising solutions should incorporate a certain principle: changing the static CDN into an adaptive dynamic CDN. The network adaptively changes its architecture to reach the best optimum according to the observed traffic load. When the load is under control, the normal client/server (C/S) configuration manages the requests well. When the load exceeds a threshold or fulfills a certain condition, then the set of surrogate servers kicks in to absorb the huge number of requests. In this way, the network is supposed to be more efficient as regards resource utilization, practical in its ability to address flash crowds and affordable by more Web sites.

There are three main challenges in addressing this issue:

1. How to organize a temporary overlay of surrogate servers quickly. The surrogate servers should be utilized efficiently and need to cooperate with each other almost immediately when faced with a flash crowd. When the flash crowd departs, everything should go back to normal operation without involving much overhead. The impact of being a potential surrogate during the normal period of operation should be controlled, so that it is minimized as much as possible.

2. How to detect the arrival and departure of a load spike properly. Flash crowds are different from the normal workloads, whose magnitude and duration depend on the people's interest toward some triggering events, and it is difficult to make long-term predictions in advance. Thus, the network must be reactive to the arrival of a flash crowd by relying on short-term quick predictions. The detection must be careful because any improper detection may result in a waste of resources or oscillations of the network architecture.
3. How to redirect client requests transparently to the temporary overlay. Once the dynamic CDN is ready for the flash crowds, the flooded client requests must be redirected to any of the surrogates, and they should preferably be redirected in a load-balancing manner. Different from single site schemes where a local load balancer works, this redirection must be performed within a wide-area temporary environment.

We advocate FCAN (Flash Crowds Alleviation Network), a dynamic CDN network that adaptively optimizes the network architecture between C/S and CDN configurations. We utilize an Internet infrastructure of cache proxies to organize a temporary overlay of surrogate servers. This mode is invoked on the fly when a flash crowd comes, but pulled out of action when the normal C/S configuration works adequately [7, 33].

This chapter is organized as follows: Sect. 11.2 provides a brief overview of flash crowds and analyzes their triggering, types, and characteristics. It also discusses how to distinguish flash crowds from other similar Internet phenomena, Denial of Service (DoS) and Distributed DoS (DDoS) attacks. It then examines state-of-the-art research works. By analyzing and comparing several related solutions, it clarifies their advantages and disadvantages. Sect. 11.3 presents FCAN. It explains how the network reacts to the beginning and ending of a flash crowd, how the temporary surrogates are organized and cooperate with each other, and how redirection based on DNS (Domain Name System) offloads the burden on the origin Web site. It then exhibits simulation-based evaluations using real trace workloads. Section 11.4 summarizes some visionary thoughts for practitioners. Section 11.5 presents the future research directions with discussions on open issues. Section 11.6 comprises the conclusion.

## 11.2 Background and Related Work

We first study the characteristics of flash crowds. We show that network bandwidth is the most serious bottleneck, and a small number of objects is responsible for a greater percentage of requests (i.e. heavy-tailed behavior). These observations imply that flooded requests must be redirected away from the server and caching these flash-crowd objects could be a possible solution. Study of related work in this context show that there is still room for improvement to the solutions for handling the problem of flash crowds.

### *11.2.1 Flash Crowds*

Usually, sudden events of great interest trigger flash crowds, whether planned or unplanned. Some well-analyzed ones include: World Cup 1998 [6], RedHat Linux image distribution [8], Play-along TV show 2000 and Chilean presidential election broadcast 1999 [21], and CNN broadcast on the terrorist attacks of September 11, 2001 [13]. In addition, a Web site which is referred to on a popular site, news or blog often experiences an unusual amount of accesses unexpectedly.

Due to resource limits and/or the network bandwidth, the servers are unable to handle the high volume of requests. As a result, most users perceive unacceptably poor performance. Moreover, flash crowds unintentionally deny service for other users who either share common paths with the flash crowd traffic or who try to retrieve unrelated information from the same servers [30, 51].

Through analyses of such real traces as mentioned above and other research efforts [19, 27], some significant characteristics can be concluded, as stated below. These observations allow us to tell when a flash crowd arrives; how long (or short) a time we have to take defensive action; how different it is from a malicious attack; how we can utilize the locality of reference; and more.

1. The increase in the request rate is dramatic, but relatively short in duration. A flash crowd lasts as long as the attention span of the concerned audience, from hours to days, which is relatively short compared to the life span of a Web application. Therefore, if we make an over-provision or switch to the conventional CDN, the results can lead to under-utilization of resources during the normal operational period, especially for small or personal Web sites, which might experience flash crowds only once or twice in their lifetime.
2. The increase in the requests is rapid but not instantaneous. In the case of the Play-along TV show, the rate increase continued for 15 min. before it reached its peak. Another case, the September 11, 2001 event, resulted in a massive load on the CNN Web site which doubled every 7 min., finally reaching a peak of 20 times higher than the normal load [24]. This property suggests that we still have adequate time to detect a flash crowd and react.
3. Network bandwidth is the primary constraint bottleneck. CPU may be a bottleneck if the server is serving dynamically generated contents. For instance, on the morning of September 11, dynamic pages on the MSNBC news Web site consumed 49.4% of "500" (server busy) error codes [32]. However, MSNBC quickly switched to serving static HTML pages, and the percentage of the error status codes dropped to 6.7%. Observations also revealed that network bandwidth became the primary constraint bottleneck, and the closer paths are to the server, the worse they are affected [32]. It is reported that modern PCs could sustain more network throughput than 1 Gbps when serving static files [20], while the network bandwidth of a Web site is typically much lower [40]. Accordingly, we should focus on alleviating the bandwidth bottleneck around the servers.
4. A small number of contents, less than 10%, is responsible for a large percentage, more than 90%, of requests. For instance, the MSNBC traces from September 11

showed that 141 files (0.37%) accounted for 90% of the access, and 1086 files (2.87%) for 99% of the access [32]. Moreover, the set of hot contents during a flash crowd tends to be small to fit in a cache. This is a promising result implying that the caching of these 10% contents can be a solution to flash crowds. We also observe that this "10/90" rule of reference follows the Zipf-like distribution, in which the relative probability of a request for the $i$'th most popular content is proportional to $1/i^a$ [10]. This property distinguishes flash crowds from attack traffic which is generated automatically by "bots".

5. More than 60% of contents are accessed only during a flash crowd. In addition, among the 10% hot contents, more than 60% are new to being cached. For instance, 61% of contents were uncached in the Play-along case, and 82% in the Chile case [21]. This implies usual Web caches may not provide the desired level of protection. Most cache proxies on the Internet will not have the requested contents at the beginning of a flash crowd. Therefore, most requests would miss in the caches, and be forwarded to the origin server. Although subsequent requests would be served from the caches, a large number of initial cache misses will be generated to the origin server within a short period of time.
6. The number of clients in a flash crowd is commensurate with the request rate. This feature can be used to rule out malicious requests. During a flash crowd, spikes in requested volumes correspond closely with spikes in the number of clients accessing the site. The increase in traffic volume occurs largely because of the increase in the number of clients, and most requests come from a large number of client clusters. However, because a server usually slows down during a flash crowd, per-client request rates are lower than usual. This indicates that legitimate clients are responsible for the performance of a server.

While studying the behavior of flash crowds, we need to identify and distinguish related but distinct phenomena, DoS attacks. A DoS attack is "an explicit attempt by attackers to prevent legitimate users of a service from using that server" [12]. It overwhelms a target server with a huge amount of packets in primarily a brute force manner, so as to saturate the target's connection bandwidth or deplete the system resources to subvert the normal operation. Some well-known DoS attacks include: SYN attack [11], Code Red attack [29], and Password Cracking [18]. Recently, DDoS attacks, which employ a large number of "bots" emitting requests to the target, have also been frequently reported [22].

DoS attacks share several characteristics with flash crowds. They both overload a server's Internet connection and result in partial or complete failure. However, the server should ignore DoS attacks during flash crowd protection, and handle legitimate requests only. There are some ways to distinguish DoS attacks from flash crowds [21] : (1) Client distribution across ISPs and networks does not follow population distribution; (2) Cluster overlap which a site sees before and during the attack is very small; (3) Per-client request rate is stable during the attack and deviates significantly from normal; and (4) The distribution of files (which may not even exist on the server) targeted by attackers is unlikely to be Zipf-like.

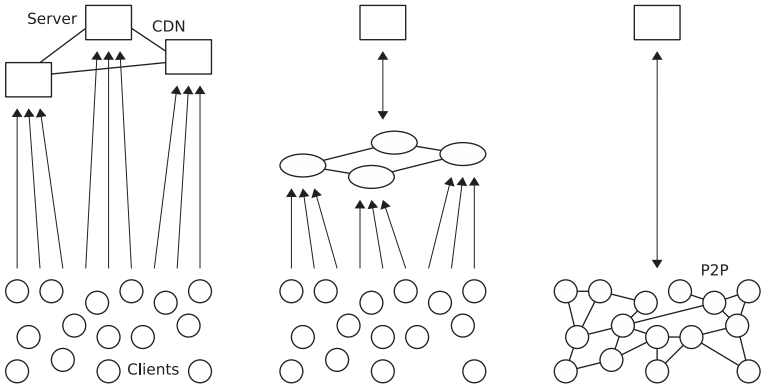
By exploiting these differences, a server may take a strategy for distinguishing DoS attacks from flash crowds, and discard these malicious requests as early as

possible. It may monitor clients that access the site and their request rates, and perform some checks on the content of packets, HTTP headers, and arrival rates.

More details and implementations on how to distinguish malicious requests from legitimate ones are beyond the scope here, as exclusive coverage of works in this respect can be found in literature [21, 22, 35]. This chapter assumes that servers have already ruled out malicious requests of DoS attacks by using some mechanisms.

## *11.2.2 Possible Solutions*

Solutions proposed so far for addressing flash crowds are classified into three categories according to the typical architecture of networks: server-layer, intermediate-layer and client-layer solutions. Figure 11.2 shows their schematic overviews.




(a) Server Layer (b) Intermediate-Layer (c) Client-Layer

**Fig. 11.2** Three solutions

### 11.2.2.1 Server-Layer Solutions

As mentioned above, traditional over-provisioning and use of static CDN [3, 25, 43] on the server-side are straightforward but costly approaches. They are inefficient and difficult to deal with short-term Internet congestion. Due to the unpredictability of flash crowds, any imperfectly provisioned system is likely to fail under sustained overload conditions.

**CDN with Dynamic Delegation** J. Jung, et al. [21] proposed an adaptive CDN using dynamic delegation to improve the protection of origin servers under heavy load conditions. They organize surrogate servers into groups, with one surrogate within a group selected to be the primary surrogate. Usually, a DNS server for the CDN

assigns client requests only to primary surrogates. When the load on the primary surrogate reaches a alarming level, the primary surrogates asks DNS to reassign requests to other members in the group called delegates. When a delegate receives a missing request, it forwards the request not to the origin server but to the delegate's primary. This mechanism is called "dynamic delegation". When delegates are engaged, the system behaves like cooperative caching.

Dynamic delegation takes 60% uncached objects into consideration at the beginning of flash crowds, as mentioned above, and improves the efficiency of the system. It pulls the popular objects from the origin server, and absorbs the cache-miss requests by hierarchical caching. However, the surrogate groups with primaries should be configured manually and permanently even during peaceful periods.

**DotSlash** DotSlash [50] allows different Web sites to form a mutual-aid community and to use spare capacity within the community so as to relieve flash crowds experienced by any individual site. As a rescue system, DotSlash continuously monitors the workload at each Web server; when a server becomes heavily loaded, rescue services are activated, and once the server's load returns to normal, the rescue services cease. As a result, a Web site has a dynamic server set which includes a single or a cluster of fixed origin servers, and a changing set of rescue servers.

Different from most other systems mentioned here and in 11.2.2.2, which use permanent and fixed resources, DotSlash triggers its rescue system on a temporary mutual-aid community. However, DotSlash needs clients to connect with the origin server first, and then issues a redirected URI containing the virtual host name for DNS redirection. Consequently, there is a risk that the bandwidth and processing needed to send the redirection messages may itself overwhelm the origin server.

#### 11.2.2.2 Intermediate-Layer Solutions

There have been some intermediate-layer solution proposals for dealing with flash crowds, which utilize network resources to perform offload. Caching techniques help to alleviate server load during flash crowds by filtering out repeated requests from groups of clients which share a proxy cache.

In general, proxies on the Internet are divided into two types: forward proxies and reverse proxies. Forward proxies are placed near clients and thus far from the server end. Their typical functionality includes a firewall, and caching of static contents. They are usually shared by many clients and are reasonably powerful and stable. However, content providers do not have much control over them. Reverse proxies are placed near the back-end server farm, and act as agents of application providers. They serve requests on behalf of the back-end servers. Content providers can fully control their behavior. However, the scale of reverse proxies only goes as far as a content provider's network bandwidth allows [47].

**Multi-Level Caching** The solution using multi-level caching [4] argues that with proper replacement algorithms, a caching infrastructure designed to handle normal Web loads can be enough to handle flash crowds. It studies the effects of using

different cache replacement algorithms, changing the placement of caches, using heterogeneous multi-level caching, and partitioning the ID space based on document size. The work concludes that using GDSF algorithm [5], the replacement policy in caches results in significant improvements to the client response times, and server and network loads.

Multi-Level Caching offers promising results for using caching to address flash crowds for small and static objects. The system needs a dedicated deployment of hierarchical caching placement, and complete control over the infrastructure of forward cache proxies. The system does not address the problem of 60% uncached objects, and thus it may not provide the desired level of protection to the origin server at the initial stage. In addition, it currently lacks an adaptive mechanism for handling flash crowds flexibly.

**BackSlash** Backslash [44] uses content-addressable P2P overlays based on distributed hash tables (DHTs) to build distributed Web server systems. It places copies of contents on mirror servers which are specified by content providers. DHTs provide bases for the self-organization of participants, for routing requests, and for load balancing.

BackSlash uses Web servers and proxies to offload the network traffic. However, in BackSlash, the contents on mirror servers must be pre-placed and well-organized in advance, which incurs operation complexity and restricted extensibility of the system.

**CoralCDN** CoralCDN [17] leverages the aggregate bandwidth of volunteers to absorb and dissipate most of the traffic for Web sites using the system. As we have seen in Chap. 1, CoralCDN exploits overlay routing techniques on top of a key/value indexing infrastructure: a P2P distributed sloppy hash table, or DSHT, which allows nodes to locate nearby cached copies of Web objects without querying more distant nodes and which prevents hot spots in the infrastructure, even under degenerate loads. We also know that to use CoralCDN, a content publisher or someone posting a link to a high-traffic portal simply appends ".nyud.net:8090" to the hostname in a URL.

Coral uses volunteers' additional capacities to absorb the overwhelming traffic. It combines a set of P2P-based reverse proxies to create cache objects on demand, and adopts DNS to redirect client requests transparently. CoralCDN is always waiting for the incoming requests, whose URL needs to be manually configured by appending ".nyud.net:8090" in advance. With a modified URL, CoralCDN is capable of object-oriented redirection, however, it sacrifices user unawareness of the system.

#### 11.2.2.3 Client-Layer Solutions

Client-side solutions make clients help each other in sharing objects so as to distribute the load burden from a centralized server. An origin Web server can mediate client cooperation by redirecting a client to another client that has recently downloaded the objects, as in Squirrel [28], Pseudoserving [23] and CoopNet [32]. Clients can also form P2P overlay networks and use search mechanisms to locate re-

sources. For example, PROOFS [45] employs randomization to build client side P2P overlay networks, and BitTorrent [9] breaks large files into small parts for efficient retrieval. In general, these solutions rely on the client-side cooperation. They must be deployed on users' desktop PCs, which are thus likely to prevent their widespread deployment.

**CoopNet** Cooperative networking [32] is a P2P caching solution that complements traditional client-server and client-Web proxy communication rather than replacing it. It has previously-registered clients who have already downloaded content, and they in turn serve the content to other clients. CoopNet uses HTTP-based redirection to route requests, and to select peers according to their nearby location.

In CoopNet, P2P communication kicks in during flash crowds to share the load, and gets out of the way when the C/S communication works fine. CoopNet uses a server-based redirection, which has the risk of a "single point of failure".

**PROOFS** PROOFS [39, 45] is comprised of two protocols. The first forms and maintains a network overlay. The second performs a series of randomized, scoped searches for objects atop the overlay formed by the first protocol. Nodes continually perform what is called a "shuffle operation". The shuffle is an exchange of a subset of neighbors between a pair of clients, and can be initiated by any client. Shuffling is used to produce an overlay that is "well-mixed", in that a client's neighbors are essentially drawn at random from the set of all clients that participate in the overlay. Once a random state is reached, scoped searches for objects can be performed atop the overlay. Objects are located by randomly visiting sets of neighbors until a node is reached that contains the object. Through combination of theoretical results and simulation, PROOFS claims to be robust for the overlay partitioning, for peer dynamic joining/leaving, and for limiting participation in the system.

PROOFS uses an unstructured first generation P2P system, and thus requires a lower preparation cost, and it offers good performance under the condition of flash crowds. A significant amount of attention has been paid to second generation P2P architectures such as CAN [36], CHORD [46], and Pastry [38], in which participants have a sense of direction as to where to forward requests. They provide benefit over their first generation counterparts in terms of the amounts of network bandwidth utilized and the time taken to locate those documents. However, to be able to handle documents whose popularity suddenly spikes without inundating those nodes responsible for serving these documents, the first generation architectures (which are simpler and more lightweight) are preferable.

#### 11.2.2.4 Other Works

Grid technologies allow "coordinated resource sharing and problem solving in dynamic, multi-institutional organizations" [16], with a focus on large-scale computational problems and complex applications that involve many participants and different types of activities and interactions. Internet data centers host multiple Web applications on shared hardware resources. A. Chandra, et al. suggest reacting to

changing application loads by reallocating resources to overloaded applications, borrowing these resources from other under-utilized applications if necessary [13].

As a last resort, a Web site can use admission control [14, 15, 49] to prevent itself from being overloaded, by rejecting a fraction of the client requests and only admitting preferred clients.

## 11.3 FCAN: Flash Crowds Alleviation Network

FCAN [7, 33] is an intermediate-layer solution, using a CDN-like wide-area overlay network of caching proxies which stores objects, and delivers them to clients, like the surrogate servers in a CDN. Considering the short duration and unpredictability of flash crowds, FCAN invokes the overlay only when a server is overwhelmed by a large amount of requests, and reorganizes the overlay when necessary. This dynamicity is the most prominent characteristic of FCAN compared to most of the above-mentioned related works. The only exception is DotSlash, however, it lacks an adaptive reorganization feature.

FCAN aims at complementing an existing Web server infrastructure to handle short-term load spikes effectively, but it is not intended to support a request load that is constantly higher than the planned capacity of a Web site. It targets small Web sites, although large Web sites can also gain some benefit from it.

### *11.3.1 Requirements*

Below are the functional and nonfunctional requirements which we analyzed in order to make FCAN flexible, reliable, and cost-effective.

**Object Delivery** First and foremost is the timely delivery of content objects. FCAN should maintain high availability of the delivery service at all times. Moreover, accessibility to non-flash-crowd objects on the same target server should also be ensured.

**Workload Control** FCAN should monitor changes in the increasing load and control it so that the server does not become overwhelmed. At the same time, when flooded requests are offloaded to the temporary surrogates, FCAN should also have a workload monitor on each surrogate to detect the leaving of flash crowds, and to control the redirected requests so as not to overload the surrogate.

**Adaptive Transition** FCAN should be sensitive to the load increase and transit its architecture in a flexible fashion in order to obtain optimum performance output. The duration time should be short to take the transition into effect. Both the detection and transition should be conducted automatically.

**Request Redirection** There should be a mechanism to direct the flooded requests by finding temporary surrogates. Moreover, the most appropriate surrogate should

be selected. It would be ideal if the redirection being carried out is uniformly balanced.

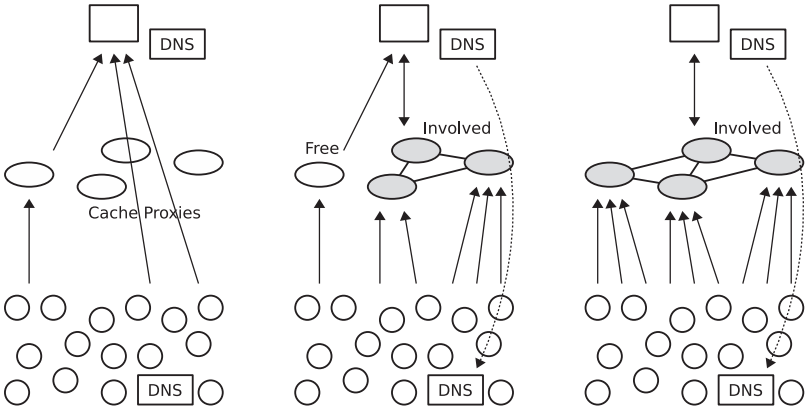
**Client Transparency** FCAN will be more acceptable if clients could remain unchanged. It is better for the clients to remain completely unaware of the existence of FCAN.

**Scalability** Because Internet-based infrastructures have the potential to reach the entire world-wide Internet community, FCAN requires the capability to expand its infrastructure easily, with minimal effort and disruption.

### *11.3.2 Design Overview*

In peaceful times, the conventional C/S architecture satisfies most of the client requests. A member server and member cache proxies, both of which comprise FCAN, do little more than what normal ones do. When a flash crowd comes, the member server detects the increase in traffic load. It triggers a subset of the member proxies to form an overlay, through which all requests are conducted. All subsequent client requests are routed to this overlay by DNS-based redirection. If the subset of proxies is not large enough to handle the amount of requests, new proxies are invited, and the overlay is enlarged. When the flash crowd declines, some proxies leave, so that the overlay shrinks and is eventually released. Figure 11.3 gives an overview of FCAN at three different states, namely, usual, initial, and enlarged state.

FCAN is not dedicated to a single member server. It is designed to be shared by several servers in need, even at the same time. Each member server uses its own overlay, small or large, and servers try mutually to prevent their overlays from overlapping as much as possible.




(a) Usual State (b) Initial Sate (c) Enlarged state

**Fig. 11.3** FCAN overview

Each member proxy is primarily a regular forward cache proxy during its normal mode, however it acts as a surrogate, somewhat similar to a reverse cache proxy, serving requests from any user during the anti-flash-crowd mode. In reality, a member proxy serves for several servers, and it is possible that any one server suffers from flash crowds while the others do not. Therefore, each member proxy has the functionality of mixed-mode operations for the forward proxy mode and the surrogate (similar to reverse proxy) mode.

### *11.3.3 Flash Crowd Detection*

As different resources such as network bandwidth, CPU and memory at a server may potentially become the bottleneck during a flash crowd, separate workload metrics should ideally be used for different resources. Each member server should do monitoring and overload/underload detection, and perform dynamic transition accordingly. The current design uses only the access arrival rate as the load metric, and uses a threshold-based scheme to trigger dynamic transition.

To detect the coming of a flash crowd, a server observes the volume of its load periodically. Once the server load exceeds the predefined threshold, $T_{high}$, the server treats it as the coming of a flash crowd.

During the flash crowd, each proxy involved in the overlay has a load monitor, which observes the number of client accesses, and the member server collects the load information from all the involved proxies periodically. When the load on the overlay of proxies decreases under a predefined threshold, $T_{low}$ ($< T_{high}$), the member server treats it as the ending of the flash crowd.

### *11.3.4 Network Transition*

When the member server detects the beginning of a flash crowd, it carries out the following procedure, in order to make some member cache proxies transit into the anti-flash-crowd mode in order to form a temporary overlay.

1. Selects a subset of proxies to form a CDN-like overlay of surrogates;
2. Triggers an update of DNS records to change the look-up entries of the Web site from the server's address to those of the proxies, so that subsequent requests are gradually redirected to the proxies along with DNS propagation;
3. Disseminates ("pushes") the flash-crowd object to the selected proxies, because more than 60% of the flash-crowd objects are uncached prior to the arrival of the flash crowd, as mentioned above;
4. Prepares to collect and evaluate statistics for the object from the involved proxies, so as to determine dynamic reorganization and release of the overlay;

Every member cache proxy carries out the following procedure upon request from the member server:

1. Changes its mode from a proxy to a surrogate (or, in the strict sense, a mixed mode of a forward proxy and a surrogate, as mentioned above);
2. Stores flash-crowd objects permanently, which should not expire until the flash crowd is over;
3. Begins monitoring the statistics of request rate and load, and reporting them to the server periodically,

The server selects the subset of proxies by probing them one by one first, because any proxy may already be involved in another flash crowd alleviation, or it may be overloaded due to some other reason. This prevents overlapping of more than one subsets for independent flash crowds. The subset can be small, even consisting of only one proxy, because FCAN has the feature of dynamic reorganization, as mentioned below. The current design expects network administrators to assign priorities to proxies for probing orders.

When the member server detects the leaving of the flash crowd, the involved proxies are dismissed one by one, in the reverse order of probing, with the following procedure:

1. The server updates the DNS records;
2. The server notifies the proxy to be dismissed;
3. The proxy changes its mode from a surrogate to a proxy.

The CDN-like overlay transits back to the normal C/S mode when all the proxies are dismissed. They are not all dismissed at once, since low load may be just temporary, and the system should therefore remain in the anti-flash-crowd mode.

### *11.3.5 Dynamic Reorganization*

Every proxy has its local monitor, which observes the request rate and the overall load on itself. Proxies involved in the overlay, whether initial or additional, send feedback information to the server periodically, including the request rate for the flash-crowd object and the overall load on the proxy.

When the request rate on the proxy is close to $T_{high}$, the proxy informs the server that the request rate is close to critical and increasing. When most of the proxies send the same information, the server starts inviting more proxies from the pool of other "free" member proxies which are not yet being involved in any overlay. The server probes the free proxies one by one to select a new proxy which can become utilized. Then the server and the new proxy carry out the same procedure as in 11.3.4.

When the load on any proxy is below $T_{low}$ and no other proxies are suffering from a high load, the system dismisses them. The selection is done in the reverse order of invitation. Since DNS propagation may take a longer time, the change in proxy mode should be done later. If clients still reach the proxy after the DNS update, the proxy will act as a normal forward proxy, and retrieve the content from its local cache, or redirect the request to the member server or any temporary surrogate which is still involved in the overlay.

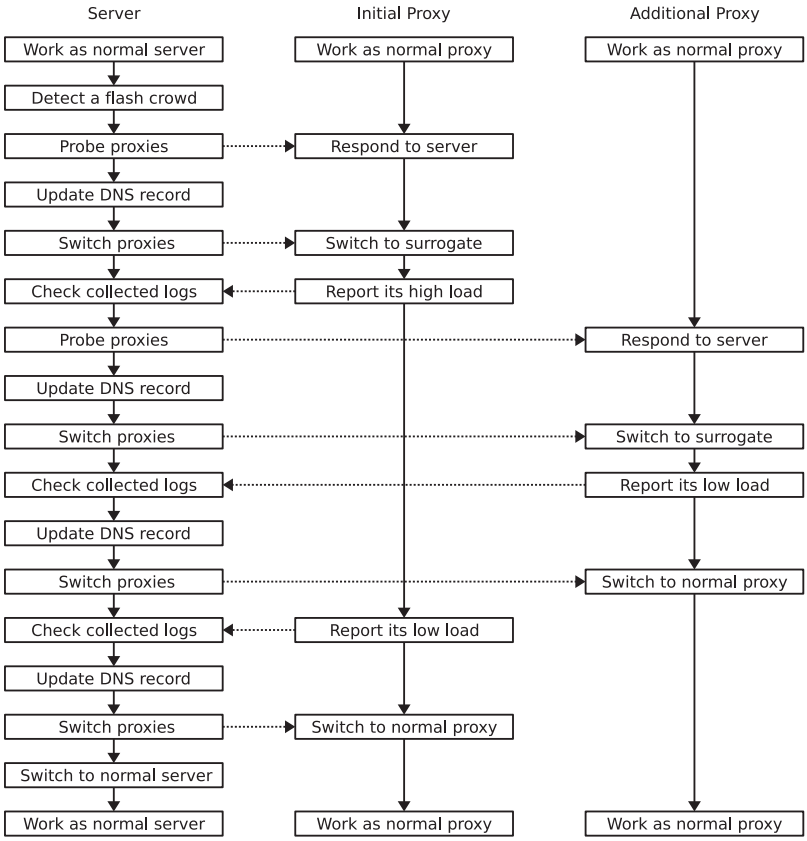



**Fig. 11.4** Process flow overview in FCAN

Figure 11.4 overviews the process flows of the server, and the initial and additional proxies, including network transition (presented in 11.3.4) and dynamic reorganization.

### *11.3.6 DNS-Based Redirection*

To protect the server and network from overload, flooded requests must be redirected. In contrast to single site schemes where local load balancers work, this redirection is done within a wide-area environment, inside which the proxies may be geographically distributed. As mentioned above, we use DNS-based request redirection. DNS is an infrastructure for all the Internet applications including the Web, it is ubiquitous across the Internet, and it is transparent to clients.

Authoritative DNS of the member server gives out the addresses of the involved member cache proxies instead of the address of the origin server when a client tries

to resolve the server name through its local DNS server. The address given to the client may be any of the proxies under a certain selection policy, possibly in a simple round-robin manner or preferably in a load-balancing and proximity-based manner. Redirected requests for flash-crowd objects are conducted by the target proxy [48].

We use a specialized DNS server (or, in the strict sense, a DNS wrapper), called TENBIN [41, 42], on the server site which allows DNS look-up entries to be modified dynamically. TENBIN is one of our research products, and has already been used in practice, for example, in the "Ring Server" [37] and "LIVE! ECLIPSE" [26] projects. TENBIN also supports policy configuration for selecting an "appropriate" address. The policy could be based on a load-weighted algorithm, a proximity-based algorithm, a cache locality-based algorithm, or it could be conducted as simply as in a round-robin fashion.

Once being modified, the new addresses are propagated through the Internet to the client side DNS servers. One problem is that DNS caches may delay the propagation, with the result that the requests still continue to go to the origin server. This can be controlled by setting DNS records with a short expiration time, i.e. zero Time to Live(TTL). We have amassed much experience on DNS propagation both from experiments and from the practical use of TENBIN. It requires $10 \sim 15$ min. to complete worldwide propagation, but this is negligible compared to a typical flash crowd which may last for several hours or several days [12].

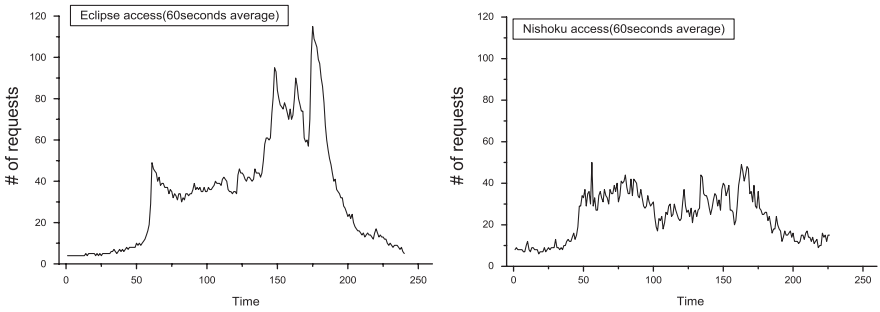
### *11.3.7 Simulation-Based Evaluations*

For preliminary verification and evaluation of FCAN, we built a thread-based simulator of a virtual network with TCP/UDP and application layers. We have run experiments considering several scenarios of flash crowds, and below we present one with real access logs which were provided from the "LIVE! ECLIPSE" project [26]. On March 29th, 2006, from 9:00 to 11:30 GMT, the project delivered Web streaming, from two different server sites, for the solar eclipse that took place in Turkey, Libya, and Egypt. The two sites were:

- http://www.live-eclipse.org
- http://www.nishoku.jp

While the former was accessed by clients from all over the world, the latter was accessed mostly by clients in Japan. There was a difference in access patterns for these two sites, since the expected access rate for the Live-Eclipse site was much higher than for the Nishoku site. Figure 11.5 shows the log data of the accesses for these sites for the period during which the eclipse was in process.

When fed to the simulator, these logs for the two sites were scaled down. The log of Live-Eclipse has been scaled down by 30, and the log for Nishoku by 10. Every simulation second corresponds to one minute of real time. Our experiment used two different member servers: one (SVR01) for Live-Eclipse, and the other (SVR02) for Nishoku. The experiment used ten member cache proxies for alleviation. The priorities (probing order) of the proxies for these member servers were set differently,




(a) Access to www.live-eclipse.org (b) Access to www.nishoku.jp

**Fig. 11.5** Accesses to two eclipse streaming sites

and the initial subsets of proxies were also different between the member servers according to their priorities and the magnitude of the flash crowd.

Figures 11.6 and 11.7 show the results of the simulation, where Fig. 11.6 shows the "Live Eclipse" overlay around SVR01, and Fig. 11.7 shows the "Nishoku" overlay around SVR02. The left graphs in Figs. 11.6 and 11.7 include average loads of the proxies, while the right graphs include individual loads.

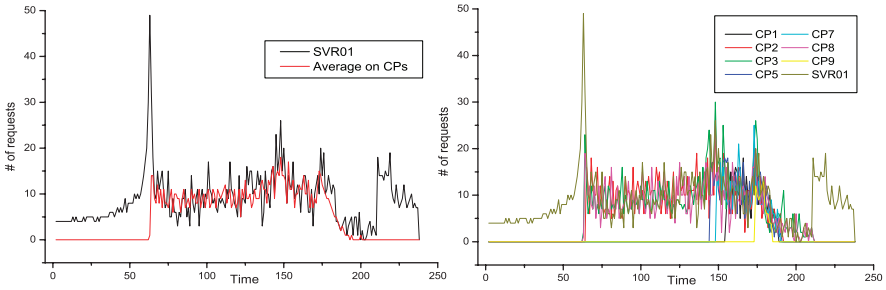
In the "Live Eclipse" overlay, seven proxies, two initials and five additionals, were involved, as shown below:

| SVR01 | Joins at: | Leaves at: |
|---|---|---|
| CP8 (initial) | 63 | 211 |
| CP3 (initial) | 63 | 211 |
| CP2 | 65 | 191 |
| CP5 | 145 | 190 |
| CP7 | 149 | 189 |
| CP1 | 155 | 188 |
| CP9 | 174 | 184 |

For the first 60 sec., the server SVR01 handles the client requests by itself. The flash crowd to SVR01 starts at around the 60th second, and the server first invites two proxies to join in the alleviation process. These two and an additional one handle the load until the next rapid increase starting at around the 150th second. Then four more proxies are invited one by one. Using all of them, the average load on the system is kept below the threshold. After the 180th second, the amount of client requests starts decreasing, and the system dismisses the proxies one by one until the system is switched back to the C/S mode. The mode change occurs around the 200th second.

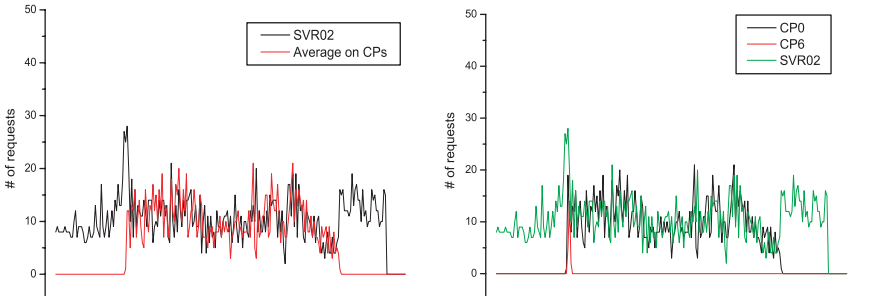
In the "Nishoku" overlay, only two proxies, one initial and one additional, are involved because of the relatively lower load, as presented below:

| SVR02 | Joins at: | Leaves at: |
|---|---|---|
| CP0 (initial) | 48 | 189 |
| CP6 | 49 | 51 |




(a) Server and Average of Proxies (b) Server and Individual Proxies

**Fig. 11.6** Load alleviation in "Live Eclipse" overlay




(a) Server and Average of Proxies (b) Server and Individual Proxies

**Fig. 11.7** Load alleviation in "Nishoku" overlay

The flash crowd to SVR02 starts at around the 50th second. At this moment, the highest peak of client requests is reached. CP0 is initially involved in the overlay, then immediately CP6 is invited, but only for 2 sec.

### *11.3.8 Concluding Remarks*

The most prominent characteristics of FCAN are its dynamic and adaptive organization and reorganization features of the CDN-like overlay, which, as far as we know, cannot be found in any other related works. Here, we presented here that these unique features of FCAN are effective. The intermediate-layer solution using cache proxies in FCAN is, compared to the client-layer solutions, easier to manage and control. Moreover, compared to the server-layer solutions, it is more flexible and closer to clients.

The current design is the first version, and it still has some features which need to be improved. Threshold-based flash crowd detection should be more sophisticated,

and this will be discussed later. Priority-based proxy grouping is now being replaced by an autonomous decentralized clustering scheme.

Another issue is the coarse granularity of redirection. A flash crowd is object-oriented, while DNS-based redirection is machine-oriented, since DNS deals only with machine names. It would be preferable to direct only the requests for flash-crowd objects to the proxy overlay and to pass other requests for the non-flash-crowd objects through, as usual. HTTP-based redirection and URL rewriting techniques offer fine-grained object-oriented redirection, however, they are not transparent to clients.

Quantitative and rigorous evaluations of FCAN are not included in the preliminary simulations so far. Real implementation on the Internet will be consisting of:

- A specialized Web server with a wrapper module for FCAN functions. Its core could be Apache for example, and the wrapper would intercept requests to the core server.
- A specialized cache proxy with a wrapper module for FCAN functions. Its core could be Squid for example, and the wrapper would intercept requests to the core proxy.
- An enhanced DNS server. TENBIN is a good candidate, which is actually already used in practice.

FCAN was originally designed for flash crowd protection, but in fact, it is not only limited to this. It adjusts server load under a predefined threshold facing against any unexpected traffic surges, and we can thus assume that some kinds of DDoS attacks could also be handled.

## 11.4 Visionary Thoughts for Practitioners

Table 11.1 summarizes some significant related research efforts (most of them are mentioned so far in this chapter) and compares them with FCAN. Our observations are presented in the following:

1. Over-provisioning based on peak demand or using CDNs to increase server locations in advance is costly and inefficient.
2. A client-side P2P overlay addresses flash crowds reasonably well, but not perfectly, since it loses client transparency and controllability.
3. In addition, some P2P systems have overheads, such as the flooding problem, which cannot be neglected while facing the ending of flash crowds.
4. Intermediate-layer solutions have advantages over other layer solutions. This is because the caching technique is promising in its ability to address flash crowds whose target objects are supposed to be small-sized and static.
5. However, most intermediate-layer solutions neglect the problem that more than 60% of objects are uncached at the beginning of a flash crowd, which results in the origin server being at risk for a surge of cache misses.

**Table 11.1** Summary of design issues adopted by related systems

| Design Issue | | DD | DS | ML | BS | CO | CP | PF | FC |
|---|---|---|---|---|---|---|---|---|---|
| System Archtecture | Server-layer | √ | √ | | | | | | |
| | Proxy-layer | | | √ | √ | √ | | | √ |
| | Client-layer | | | | | | √ | √ | |
| Surrogate Servers | Dedicated Servers | √ | | | | | | | |
| | Existing Servers | | √ | | √ | | | | |
| | Existing Proxies | | | √ | √ | √ | | | √ |
| | Existing Clients | | | √ | | | √ | √ | |
| Client Transparency | Client Unaware | √ | | √ | | | | | √ |
| | Browser Unchanged | √ | √ | | √ | √ | | √ | √ |
| Client Redirection | DNS-based | √ | √ | | √ | √ | | | √ |
| | URL Rewrite | √ | √ | | √ | √ | | | |
| | HTTP-based | | | | | | √ | | |
| Replica Placement | Mirror Replica | √ | √ | | | | | | |
| | Caching on Demand | √ | √ | √ | | √ | | √ | √ |
| Object Locating | DHT-based P2P | | | | √ | √ | | | |
| | Unstructured P2P | | | | | | √ | √ | √ |
| | Cooperative Caching | | | √ | | | | | |
| Cache Miss Avoidance | Dynamic Delegation | √ | | | | | | | |
| | Push Service | | | | | | | | √ |
| Adaptive Transition | Temporary Servers | | √ | | | | | | |
| | Temporary Proxies | | | | | | | | √ |
| | Temporary Clients | | | | | | √ | | |

Note. DD: CDN with Dynamic Delegation, ML: Multi-Level Caching, BS: BackSlash, DS: DotSlash, CO: CoralCDN, CP: CoopNet, PF: PROOFS, FC: FCAN.

6. Forward proxies rather than servers are better employed as surrogate servers. Proxies are nearer to clients, and are thus more beneficial to client response time and network congestion.
7. To handle flash crowds flexibly and efficiently, an adaptive transition technique is necessary, which organizes the potential resources along the way, rather than occupying them all the time.

To sum up, each of the current research works have both merits and demerits. Through the comparison, we have come to conclude that there is still a lack of an efficient approach that can handle flash crowds in a flexible, reliable, and cost-effective manner, while remaining transparent to the end users.

## 11.5 Future Research Directions

While there are many research problems required to be addressed in the context of flash crowds alleviation, as future research directions, we focus on two main issues.

**Early Detection of Flash Crowds** We have already noticed the phenomenon that shortly before a flash crowd comes to a server, a number of requests sometimes floods to DNS servers to resolve the server's name. This must imply that if we had a technique for collecting the amount of requests from distributed DNS servers and for analyzing them, we could possibly predict the coming of a flash crowd, and thus give an advance warning to the target server.

**Handling of Dynamic Objects** It must be common to all the CDN systems to address dynamic object dissemination. Dynamic objects can be divided into two categories:

- Dynamically generated contents (mostly using script codes and a back-end database)
- Frequently updated contents (as often found in News sites)

The simplest way would be to replace a dynamic object with its trimmed static version under a heavily-loaded situation at the cost of its service quality [1].

It must be relatively easy to handle a dynamic object in the former category, if the back-end database is read-only. If not, or if a dynamic object falls in the latter category, we must provide a fast and reliable scheme for updating all the replicas in a consistent manner. This topic, update synchronization and coherence, has been investigated extensively in the area of distributed databases, distributed caches, and distributed shared memories. Achievements out of these studies could be applied in this context.

Finally, some integration among server-layer, intermediate-layer and client-layer solutions could be interesting and promising.

## 11.6 Conclusion

Short-term Internet congestion, known as flash crowds, poses new challenges for designing scalable and efficient distributed server systems. This chapter analyzed the major characteristics of flash crowds, studied the related research works extensively, and pointed out the need for a dynamic network to handle short-term Internet congestion. Then we presented our original and unique idea of a dynamic CDN network which adaptively optimizes its own network architecture between C/S and CDN configurations to alleviate flash crowds. Our observations suggest that FCAN could be a good basis for performing early detection of flash crowds, and handling of dynamic objects. Therefore, we conclude that it could be a pathway to realize future innovations for handling flash crowds efficiently.

**Acknowledgements** This chapter is based on joint work with Prof. Toshihiko Shimokawa (Kyushu Sangyo University, Japan), Dr. Chenyu Pan (China), and Dr. Merdan Atajanov (Turkmenistan). We also thank Ms. Kate Miller for English proof reading.

## References


1. Abdelzaher TF, Bhatti N (1999) Web Server QoS Management by Adaptive Content Delivery. In: Computer Networks, 31(11–16):1563–1577
2. Adler S (1999) The Slashdot Effect, an Analysis of Three Internet Publications. http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html
3. Akamai Technologies Inc. http://www.akamai.com
4. Ari I, Hong B, Miller EL, Brandt SA, Long DE (2003) Managing Flash Crowds on the Internet. In: Proc. 11th IEEE/ACM Int. Symp. on Modeling, Analysis, and Simulation of Comp. and Telecomm. Sys., 246–249
5. Arlitt M, Cherkasova L, Dilley J, Friedrich R, Jin T (1999) Evaluating Content Management Techniques for Web Proxy Caches. In: ACM SIGMETRICS Performance Evaluation Review, 27(4):3–11
6. Arlitt M, Jin T (2000) A Workload Characterization of the 1998 World Cup Web Site. In: IEEE Network, 14(3):30–37
7. Atajanov M, Shimokawa T, Yoshida N (2007) Autonomic Multi-Server Distribution in Flash Crowds Alleviation Network. In: Proc. IFIP 3rd Int. Symp. on Network Centric Ubiquitous Systems (LNCS 4809, Springer), 309–320
8. Barford P, Plonka D (2001) Characteristics of Network Traffic Flow Anomalies. In: Proc. ACM SIGCOMM Internet Measurement Workshop, 69–73
9. BitTorrent Website. http://www.bittorrent.com/
10. Breslau L, Cue P, Fan L, Phillips G, Shenker S (1999) Web Caching and Zipf-like Distributions: Evidence and Implications. In: Proc INFOCOM 1999, 126–134
11. CERT (1996) TCP SYN Flooding and IP Spoofing Attacks. Advisory CA-1996-21, http://www.cert.org/advisories/CA-1996-21.html
12. CERT (1999) Denial of Service Attacks. http://www.cert.org/tech_tips/denial_of_service.html
13. Chandra A, Shenoy P (2003) Effectiveness of Dynamic Resource Allocation for Handling Internet Flash Crowds. Tech. Report, TR03-37, Dept. of Computer Science, Univ. of Massachusetts Amherst
14. Chen X, Heidemann J (2002) Flash Crowd Mitigation via an Adaptive Admission Control Based on Application-Level Measurement. Tech. Report, ISI-TR-557, USC/ISI
15. Cherkasova L, Phaal P (2002) Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites. In: IEEE Trans. on Computers, 51(6):669–685
16. Foster I, Kesselman C, Tuecke S (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: Int. J. of High Performance Computing Applications, 15(3):200–222
17. Freedman MJ, Freudenthal E, Mazieres D (2004) Democratizing Content Publication with Coral. In: Proc. 1st USENIX/ACM Symp. on Networked Systems Design and Implementation
18. Houle KJ, Weaver GM, Long N, Thomas R (2001) Trends in Denial of Service Attack Technology. CERT Coordination Center White Paper, http://www.cert.org/archive/pdf/DoS trends.pdf
19. Iyengar AK, Squillante MS, Zhang L (1999) Analysis and Characterization of Large-Scale Web Server Access Patterns and Performance. In: World Wide Web, 2(1–2):85–100
20. Joubert P, King R, Neves R, Russinovich M, Tracey J (2001) High-Performance Memory-Based Web Servers: Kernel and User-Space Performance. In: Proc. USENIX 2001, 175–188
21. Jung J, Krishnamurthy B, Rabinovich M (2002) Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In: Proc. 11th Int. World Wide Web Conf., 252–262
22. Kandula S, Katabi D, Jacob M, Berger A (2005) Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In: Proc. USENIX 2nd Symp. on Networked Systems Design and Implementation, 287–300
23. Kong K, Ghosal D (1999) Mitigating Server-Side Congestion in the Internet through Pseudoserving. In: IEEE/ACM Trans. on Networking, 7(4):530–544
24. LeFebvre W (2002) CNN.com: Facing a World Crisis. In: USENIX Annual Tech. Conf., http://tcsa.org/lisa2001/cnn.txt

25. LimeLight Networks. http://www.limelightnetworks.com/
26. LIVE! ECLIPSE. http://www.live-eclipse.org/index_e.html
27. Lorenz S (2000) Is Your Web Site Ready for the Flash Crowd? In: Sun Server Magazine 2000/11, http://www.westwindcos.com/pdf/sunserver_11900.pdf
28. Lyer S, Rowstron A, Druschel P (200) Squirrel: A Decentralized Peer-to-Peer Web Cache. In: Proc. 21th ACM Symp. on Principles of Distributed Comp., 213–222
29. Moore D (2001) The Spread of the Code-Red Worm (CRv2). http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml
30. Nah F (2004) A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait? In: Behaviour and Information Technology, 23(3):153–163
31. Niven L (1973) Flash Crowd. In: The Flight of the Horse, Ballantine Books, 99–164
32. Padmanabhan VN, Sripanidkulchai K. (2002) The Case for Cooperative Networking. In: Proc. 1st Int. Workshop on Peer-to-Peer Systems, 178–190
33. Pan C, Atajanov M, Hossain MB, Shimokawa T, Yoshida N (2006) FCAN: Flash Crowds Alleviation Network Using Adaptive P2P Overlay of Cache Proxies. In: IEICE Trans. on Communications, E89-B(4):1119–1126
34. Pan C (2006) Studies on Adaptive Network for Flash Crowds Alleviation. Ph. D. Thesis, Saitama University
35. Park K, Lee H (2001) On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In: Proc. ACM SIGCOMM 2001, 15–26
36. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A Scalable Content-Addressable Network. In: Proc. ACM SIGCOMM 2001, 161–172
37. The Ring Server Project. http://ring.aist.go.jp/index.html.en
38. Rowstron A, Druschel P (2001) Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In: Proc. ACM 18th Symp. on Operating Systems Principles, 188–201
39. Rubenstein D, Sahu S (2001) An Analysis of a Simple P2P Protocol for Flash Crowd Document Retrieval. Tech.Report, EE011109-1, Columbia Univ.
40. Saroiu S (2001) Bottleneck Bandwidths. http://www.cs.washington.edu/homes/tzoompy/sprobe/webb.htm
41. Shimokawa T, Yoshida N, Ushijima K (2000) Flexible Server Selection Using DNS. In: Proc. Int.Workshop on Internet 2000, in conjunction with IEEE-CS 20th Int. Conf. on Distributed Computing Systems, A76–A81
42. Shimokawa T, Yoshida N, Ushijima K (2006) Server Selection Mechanism with Pluggable Selection Policies. In: Electronics and Communications in Japan, III, 89(8):53–61
43. Sivasubramanian S, Szymaniak M, Pierre G, Steen M (2004) Replication for Web Hosting Systems. In: ACM Comp. Surveys, 36(3):291–334
44. Stading T, Maniatis P, Baker M (2002) Peer-to-peer Caching Schemes to Address Flash Crowds. In: Proc. 1st Int. Workshop on Peer-to-Peer Systems, 203–213
45. Stavrou A, Rubenstein D, Sahu S (2004) A Lightweight, Robust P2P System to Handle Flash Crowds, In: IEEE J. on Selected Areas in Comm., 22(1):6–17
46. Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H (2001) Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. ACM SIGCOMM 2001, 149–160
47. Wang J (1999) A Survey of Web Caching Schemes for the Internet. In: ACM Comp. Comm. Review, 29(5):36–46
48. Wang L, Pai V, Peterson L (2002) The Effectiveness of Request Redirection on CDN Robustness. In: ACM Operating Systems Review, 36(SI):345–360
49. Welsh M, Culler D (2003) Adaptive Overload Control for Busy Internet Servers. In: Proc. USENIX Conf. on Internet Technologies and Systems
50. Zao W, Schulzrinne H (2004) DotSlash: A Self-Configuring and Scalable Rescue System for Handling Web Hotspots Effectively. In: Proc. Int. Workshop on Web Caching and Content Distribution, 1–18
51. Zona Research, Inc.(1999) The Economic Impacts of Unacceptable Web-Site Download Speeds. White Paper, http://www.webperf.net/info/wp_downloadspeed.pdf

# Chapter 12
# Collaborative Media Streaming Services Based on CDNs

Giancarlo Fortino, Carlo Mastroianni, and Wilma Russo

## 12.1 Introduction

In recent years, Content Delivery Networks (CDNs) have been demonstrated to be a highly efficient solution to provide media streaming services over the Internet ranging from TV broadcasts to video on-demand [1]. However, modern multimedia applications do not just perform retrieval or access operation on content but also create content, modify and manage content, and actively place content at appropriate locations to provide new, added-value services [2].

CDNs can be effectively used to support collaborative media streaming services and in particular, the collaborative playback service [4] which allows an explicitly-formed group of clients to request, watch, and control a streamed multimedia session in a shared way.

This chapter introduces a CDN-based architecture supporting the collaborative playback service which provides significant performance improvements from the point of view of media streaming delivery and control, with respect to the available centralized architectures supporting the collaborative playback service [4]. In particular, this chapter presents the Hierarchical COoperative COntrol Protocol (HCOCOP) which enables the shared media streaming control in collaborative playback sessions supported by CDNs. HCOCOP is mapped on the hierarchical control structure which is formed and supported by the CDN-based architecture when a collaborative playback session is set up. This hierarchical control structure is composed of a coordination server at the root level, one or more control servers at the intermediate level, and clients at the leaf level.

HCOCOP is implemented and evaluated through discrete-event simulation and in particular, the performance evaluation phase, which has involved symmetric and

Giancarlo Fortino
DEIS – Università della Calabria, Rende (CS), Italy, e-mail: g.fortino@unical.it

Carlo Mastroianni
ICAR-CNR (Italian National Research Council), Rende (CS), Italy, e-mail: mastroianni@icar.cnr.it

Wilma Russo
DEIS – Università della Calabria, Rende (CS), Italy, e-mail: russow@si.deis.unical.it

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

asymmetric topologies of the control structure and three different versions of HCOCOP (NoCoop, LocalCoop, and GlobalCoop), allows to analyze two significant performance indices (blocking probability and denial probability) which characterize the protocol performance.

The remainder of this chapter is organized as follows. We start with describing the architectures for providing collaborative playback services. Then we present an overview of the academic streaming CDN called COMODIN. We describe HCOCOP in Sect. 12.4, which is followed by its performance evaluation in Sect. 12.5. In Sect. 12.6, we describe two application domains enabled by a CDN-based cooperative playback system. Section 12.7 delineates future research directions. Finally, we conclude the chapter summarizing the main contributions.

## 12.2 Background and Related Work

The collaborative playback service enables an explicitly-formed synchronous group of users to select, watch and cooperatively control a remote media playback. Sessions supported by this service are named collaborative playback sessions (CPSs) [4].

In particular, a CPS includes three tightly correlated sessions:

- *Multimedia session*: A media playback in the form of a recorded audio/video presentation (e.g. a seminar), a movie, or a more complex synthetic multimedia object is synchronously transmitted to all members of the group to allow each member to watch it.
- *Control session*: The media playback is controlled by typical commands of a VCR (e.g. play, pause, seek, etc) that any member of the group can issue to change the state of the multimedia session.
- *Interaction session*: Group members can exchange messages among them for constructing and sharing knowledge on the basis of the content of the media playback.

An architecture supporting the collaborative playback service requires the following core services for organizing and running CPSs:

- *Group formation and management (GFM)*: The GFM service supports the formation and the management of collaborative groups. In particular, a group is formed around a media object selection made by the CPS group organizer and the explicit subscription of invited users.
- *Media streaming (MS)*: The MS service supports the streaming-based delivery of a selected media object to all the group members.
- *Streaming control (SC)*: The SC service allows the group members to control the multimedia session supported by the MS service.
- *Group interaction (GI)*: The GI service supports knowledge exchange among the group members through text-based messaging.

In particular, the SC service is based on a playback control protocol which allows for sending VCR-like control commands and handling the change of the CPS state when a control command affects the CPS. To regulate the activity of the group members in sending control commands the streaming control protocol has to employ coordination mechanisms. These mechanisms allow deciding which member of the group has the rights to send a control command (in case of floor-based coordination [3]) or which transmitted control command is accepted (in case of random-based coordination mechanisms [3]). Floor-based coordination relies on the concept of floor (or token) which must be explicitly acquired by a group member to send a control command. Conversely, according to random-based coordination, each group member can send a control command without requesting the floor so that contentions among control commands, simultaneously transmitted by different group members, can arise and a decision on which command should be accepted can be taken.

Moreover, the handling of the CPS state change is a crucial operation as it involves modifying the status of the playback and consistently propagating this modification to all group members.

In order to describe CPS control and state change handling, the reference Star-based abstract architecture shown in Fig. 12.1 is used. The components are: (i) the media streaming and control server (MCS), which incorporates the MS and SC services; (ii) the multicast communication channel (MCC) which supports the transmission of media streams and control commands; (iii) the collaborative client (CC) which interfaces a group member. The playback status of the CPS is managed by the MCS and changes when a control command is received. The automaton of the playback status, shown in Fig. 12.2, consists of two states (Playing and Paused) and related transitions labeled by the control commands (Play, Seek, Pause, Stop). Each

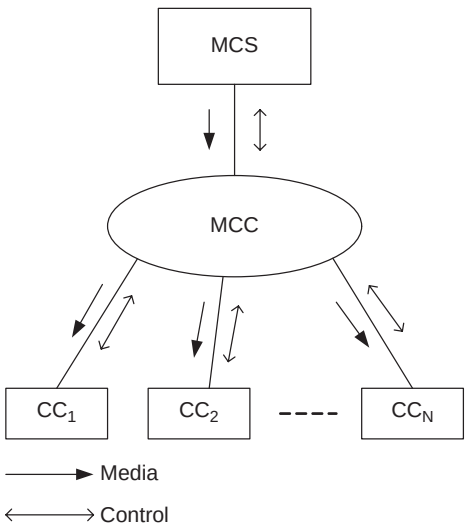



**Fig. 12.1** Reference Star-based abstract architecture

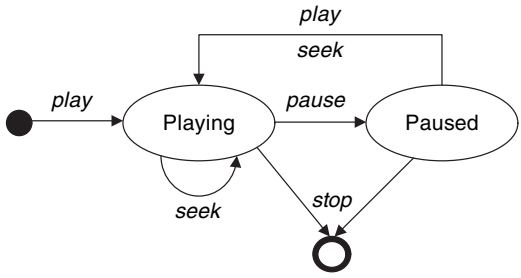



**Fig. 12.2** Automaton of the CPS playback status

CC also contains an image of such automaton which must be kept updated consistently with the automaton of the MCS. Thus, when the MCS changes the playback status automaton, it sends an update message to all the clients so that they can change their automaton accordingly. During this update the MCS will not consider any other incoming control command.

To exemplify the interaction among MCS and CCs we use the time sequence diagrams shown in Fig. 12.3, in which we assume that the coordination mechanism is random-based. In particular, three scenarios are considered: (a) *without contention*, in which only $CC_1$ issues a control command; (b) *without contention and with command discard*, in which $CC_1$ issues $command_1$ and $CC_2$ issues $command_2$ which arrives after $command_1$; and (c) *with contention*, in which $CC_1$ and $CC_2$ issue a control command quasi-simultaneously. In case (a), $CC_1$ issues a control command and the MCS, after receiving the command, processes it, changes the playback status, and transmits the update message to all CCs. In case (b), the MCS during the control command processing and the CPS state update, rejects any other control command. In case (c), $CC_1$ and $CC_2$ issue two control commands which arrive at the MCS at the same time. The MCS must take a decision about which control command to accept so that it can discard the rest. Afterwards, the MCS behaves in the same manner as in case (a).

To date few systems have been designed and implemented to provide CPSs. Their architecture can be classified as Star-based architecture (see above) or CDN-based architecture.

The MBone VCR on Demand [9], the MASH Rover [13] and the ViCRO$^C$ [4] systems rely on a Star-based architecture in which a single centralized server provides group organization, IP-multicast-based media streaming delivery, and streaming control. In particular, the media streaming delivery is based on IP-multicast for all systems. The media streaming control is based on IP-unicast for the MBone VCR on-Demand system, whereas IP-multicast is used for the other two systems. The streaming control protocols integrated in these systems use a random-based coordination mechanism, which resolves contentions by accepting the first incoming control command and discarding the others. The aforementioned systems experience two main issues: performance bottleneck represented by the centralized server and unfeasible deployment on the Internet due to the scarce availability of IP-multicast.

Furthermore a more recent streaming control protocol designed for a Star-based architecture is the COoperative COntrol Protocol (COCOP) [6]. COCOP relies on a

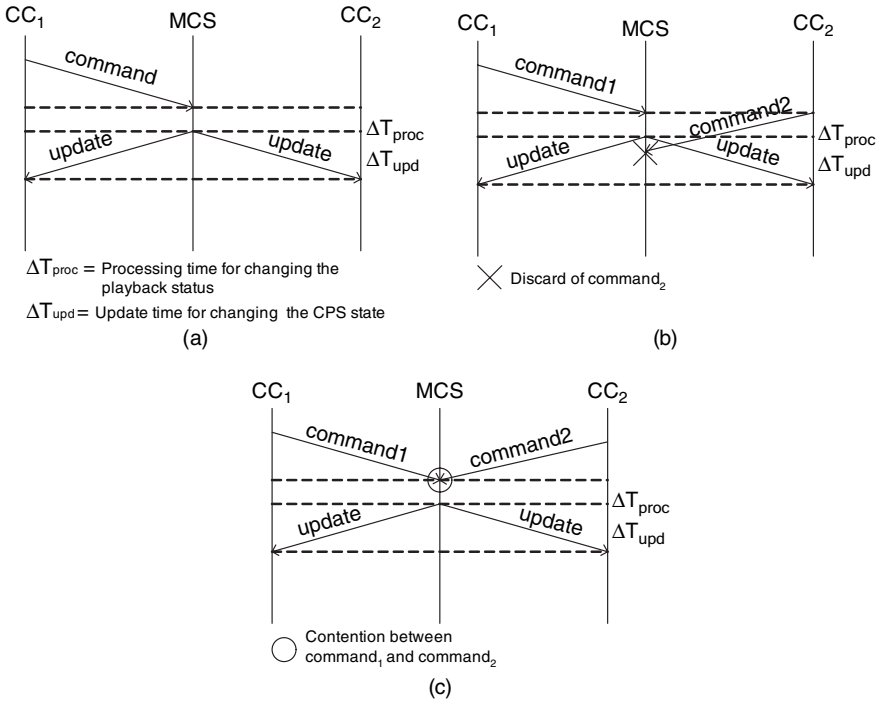



**Fig. 12.3** Time sequence diagrams of the interactions between the MCS and two CCs: (**a**) no contention; (**b**) no contention and command discard; (**c**) contention

random-based mechanism similar to that of the aforementioned systems but it also introduces a cooperation mechanism according to which a group member avoids to send any control command when it senses that another group member has already issued a control command. It is demonstrated by Fortino et al. [6] that Cooperation greatly improves performance.

The COMODIN system [7] provides the same functionalities of the Star-based systems but relies on a CDN-based architecture. This approach not only allows overcoming the issues of the aforementioned systems but also increases efficiency of the media streaming control with respect to such systems.

## 12.3 An Overview of the COMODIN System

In this section we provide a brief overview of the COMODIN system. The architecture of the COMODIN system is organized into two planes (Fig. 12.4): the *Base* plane, which consists of a streaming CDN (SCDN) providing on-demand media streaming services, and the *Collaborative* plane, which provides the collaborative playback service.

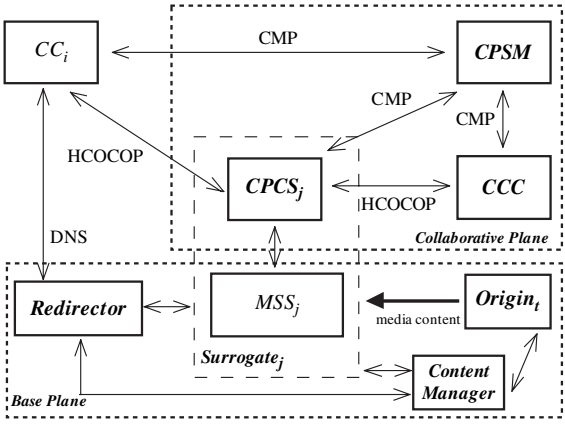



**Fig. 12.4** The COMODIN architecture

The *Base* plane is composed of the following basic network components:

- The Origin, which archives the media objects to be distributed by the CDN.
- The Surrogate, which is a partial replica of the Origin with the additional ability to temporarily store content and deliver it to clients through the access network by using the Media Streaming Server (MSS) component.
- The Client, which is a multimedia application requesting specific media content made available through the CDN.
- The Redirector, which selects the most adequate Surrogate for each different client request on the basis of a redirection algorithm [10].
- The Content Manager, which coordinates the storage of media content between Surrogates and Origin servers.

The *Collaborative* plane consists of the following additional components to provide the collaborative playback service:

- The Collaborative Playback Session Manager (CPSM), which provides the group formation and management core service which is based on collaborative playback session management protocol (CMP). In particular, the CPSM allows for the formation, (un)subscription, initiation, joining/leaving, and termination of collaborative playback sessions (CPSs).
- The Collaborative Playback Control Server (CPCS), which is integrated with the MSS of the *Base* plane and supports the remote control of the media streaming shared among the members of a CPS.
- The CPCS Coordination Channel (CCC), which coordinates distributed CPCSs serving the same CPS through the coordination channel protocol (CCP).
- The Collaborative Client (CC), which is an enhancement of the Client component of the Base plane which interfaces the user with the collaborative playback service.

A CPS supported by the COMODIN architecture can be set up and run according to the following phases:

(1) *Organization*. An organizer CC connects to CPSM and requests the organization of a CPS.
(2) *Invitation*. The organizer CC invites other CCs to subscribe to the organized CPS by means of direct messaging.
(3) *Subscription*. Invited CCs connect to CPSM and subscribe to the CPS.
(4) *Initiation*. The organizer CC connects to CPSM, requests the initiation of the CPS, and the message is then redirected to a CPCS.
(5) *Join*. The CCs become CPS members through subscription and the message are then redirected to their respective CPCSs.
(6) *Execution*. The CPS is started by any member who issues the Play control request. A CPS's state changes by a sequence of successive control requests (Pause, Play, Seek). This phase, from the control point of view, is enabled by HCOCOP which is defined in the next section.
(7) *Termination*. The CPS can be terminated by its organizer CC by means of a voting mechanism.

An example CPS scenario featured by the COMODIN system and consisting of a group of four clients organized into two subgroups (A and B) of two clients attached to two different CPCSs (CPCS A and CPCS B), is shown in Fig. 12.5.

The numbers (1)–(7) identify the interaction scenarios (or message sequences exchanged between the active components) carried out in the aforementioned corresponding phases:

(1) The client belonging to the subgroup A ($CC_1{}^A$) organizes a CPS (hereafter called $CPS_K$).
(2) $CC_1{}^A$ invites three other clients ($CC_2{}^A$, $CC_1{}^B$, and $CC_2{}^B$) to subscribe to $CPS_K$.
(3) $CC_2{}^A$, $CC_1{}^B$, and $CC_2{}^B$ subscribe to $CPS_K$.
(4) $CC_1{}^A$ initiates $CPS_K$.

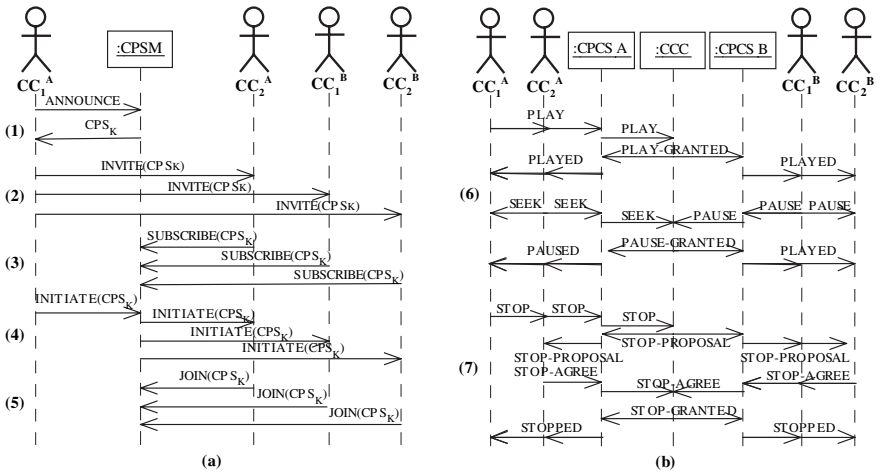



**Fig. 12.5** A CPS scenario: (**a**) CPS set-up; (**b**) a running CPS

(5) $CC_2^A$, $CC_1^B$, and $CC_2^B$ join $CPS_K$.
(6) $CC_1^A$ starts the media playback. At a given time, $CC_1^B$ requests a PAUSE and, quasi-simultaneously, $CC_2^A$ requests a Seek; $CC_1^B$ wins the competition because its command arrives before the other command.
(7) $CC_1^A$ triggers a voting procedure to tear down $CPS_K$ and $CC_2^A$, $CC_1^B$, and $CC_2^B$ agree.

## 12.4 HCOCOP

The HCOCOP is an extension of the COCOP protocol [6] for CDN-based architectures. HCOCOP relies on the following characteristics:

- *Random-based mechanism for transmitting control commands*. A control command can be sent by any group member when he/she wishes to. The avoidance of explicit synchronization mechanisms (e.g. floor-based coordination) among group members increases interactivity even though contentions among issued control commands can arise.
- *FCFS policy for contention resolution*. If two or more control commands are quasi-simultaneously issued by different group members, the control command which will drive the CPS state change is chosen on the basis of an FCFS policy and the others are all discarded.
- *Cooperation-based mechanism to reduce the transmission rate of likely unsuccessful control commands*. A group member avoids to send a control command if it detects a control command issued by another group member. This mechanism lowers the number of contentions that can arise.
- *Soft state-based management of the CPS state*. Once a control command changes the playback status, the CPS state is updated by messages and timers without managing hard states.

HCOCOP is mapped onto the hierarchical control architecture of a CPS (hereafter called $CPS_K$) as shown in Fig. 12.6. It also shows where the automata (which define the protocol behavior) are located. The control structure components are derived from the architectural control components of the COMODIN collaborative plane when a CPS is executed: $CCC_K$, is the front-end of the CCC component for the $CPS_K$; $CPCS^K{}_i$ is the front-end of the i-th CPCS for the $CPS_K$; $C^{K,i}{}_x$ is the x-th collaborative client of the $CPS_K$ served by the i-th CPCS front-end.

HCOCOP basically works as follows: if a client $C^{K,i}{}_x$ sends a control command (ClReq), its reference $CPCS^K{}_i$, before accepting it, forwards such ClReq to $CCC_K$ to resolve possible conflicts which can be generated if clients attached to other $CPCS^K{}_w$ (with $w \neq i$) send a ClReq quasi-simultaneously. $CCC_K$ accepts the first incoming ClReq, replies to all CPCSs, and discard other client requests for a given amount of time to regulate client interactivity and avoid session deadlocks. Possible conflicts generated by clients attached to the same $CPCS^K{}_i$ are instead resolved by $CPCS^K{}_i$ which adopts the same policy as the policy adopted by the $CCC_K$.

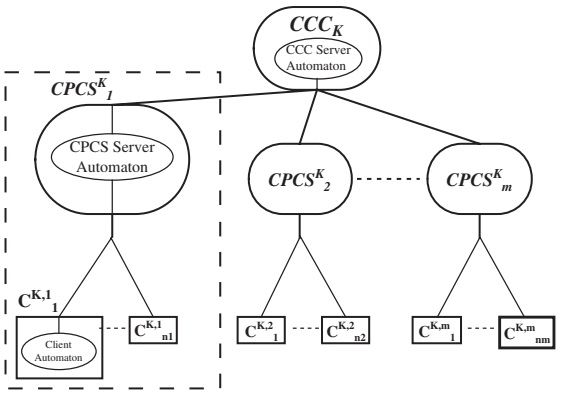



**Fig. 12.6** The CDN-based control architecture of $CPS_K$

HCOCOP can operate under three cooperation modes:

- Global cooperation (*GlobalCoop*): the ClReq is forwarded downwards by the $CPCS^K_i$ to all its attached clients and by the $CCC_K$ to all $CPCS^K_w$ ($w \neq i$) and then to all the attached clients. Such mechanism allows a client to detect a ClReq sent by other clients so as to refrain itself to send a ClReq which would be probably discarded.
- Local cooperation (*LocalCoop*): the ClReq is only forwarded downwards by the $CPCS^K_i$ to all its attached clients.
- No cooperation (*NoCoop*): the ClReq is not forwarded to any other client.

The automata defining the HCOCOP behavior are shown in Fig. 12.7. The *Client Automaton* (see Fig. 12.7a) of the client $C^{K,i}_x$ generates a client request (ClReq) when the user issues a control command (UsrReq) and enters into a Ready state. Then the request is sent to the $CPCS^K_i$, and it enters into the RequestDone state. This state is also entered when the client $C^{K,i}_x$ in the Ready state senses ClReqs sent by other clients attached to the same $CPCS^K_i$ (if *LocalCoop* is enabled) and also by other clients attached to $CPCS^K_w$ with $w \neq i$, if *GlobalCoop* is enabled. In the RequestDone state (in which the automaton remains until a Reply is received) additional ClReqs sent by other clients are ignored and the client $C^{K,i}_x$ is disabled from generating new control requests to limit the session load. It is processed after a Reply arrives. To control the interactivity degree of the session, new user control commands are blocked until a given time $T_{CC}$ elapses.

The *CPCS Automaton* (see Fig. 12.7b) of the $CPCS^K_i$ can receive a ClReq while it is in the Ready state. Reception of a ClReq makes it enter into the Synchro state. If the ClReq comes from its attached clients, such ClReq (or *upward* ClReq) is forwarded to the CCC Server Automaton and, if local or global cooperation is enabled, it is also forwarded to its other attached clients. If the ClReq comes from the CCC (i.e. a ClReq originated by clients attached to other CPCS servers), such ClReq (or *downward* ClReq) is forwarded to all the attached clients. In Synchro or Ready states, upon receiving a Reply from the CCC Server Automaton, the CPCS Automaton processes the Reply and forwards it to all its attached clients. Afterwards

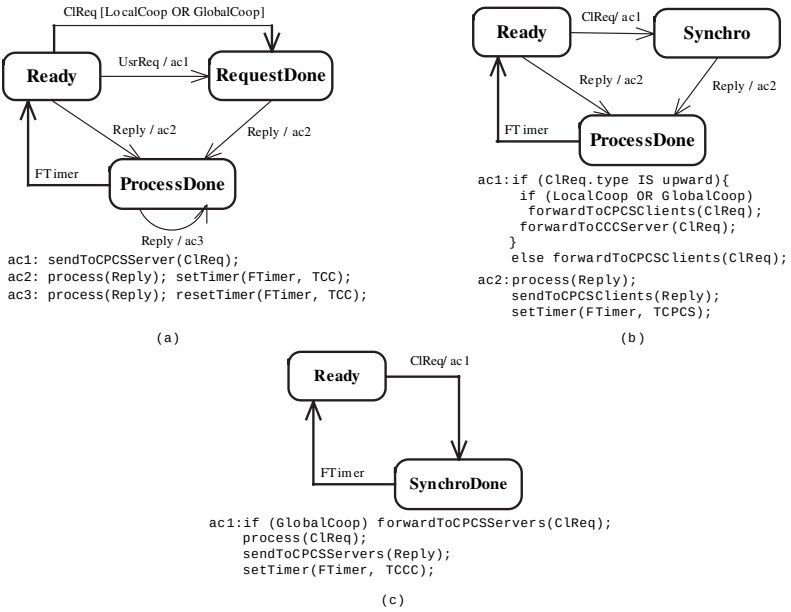



**Fig. 12.7** Automata of the HCOCOP protocol: (**a**) client automaton; (**b**) CPCS automaton; (**c**) CCC server automaton

it enters the ProcessDone state wherein it rests until a given time $T_{CPCS}$ elapses. Such delay is introduced both to make the clients aware of all changes in the session state, thus exploiting a soft-state like paradigm [12], and to regulate the group interactivity.

The *CCC Server Automaton* (see Fig. 12.7c), when receives a ClReq sent by the CPCS Automaton of $CPCS^{K}_{i}$ in the Ready state, accepts such ClReq and forwards it to all the other CPCS automata, if global cooperation is enabled. A Reply is then sent to all the CPCS automata and the CCC Server Automaton passes into the SynchroDone state wherein it rests until a given time $T_{CCC}$ elapses. Such delay is introduced to assure the consistency of HCOCOP.

## 12.5 Simulation-Based Analysis of HCOCOP

This section is focused on the analysis of the HCOCOP performance in order to demonstrate the major benefits provided by the cooperation approach in a CDN-based architecture. In this regard, an object-oriented discrete event simulation framework [6] is exploited to implement HCOCOP and evaluate its performance in CDN-based architectures having different numbers of clients and different topologies. The HCOCOP performance is also compared to the performance of non-cooperative and cooperative protocols in a Star-based architecture to show that the

use of a CDN can actually improve streaming control efficiency with respect to Star-based architectures.

### 12.5.1 Performance Indices

A cooperative playback control protocol must assure the consistency of the cooperative playback session (CPS) and, at the same time, must give users the ability to change the playback status. The definition of the performance indices, that refer to the relevant features which characterize a cooperative playback control protocol, takes into account (1) the handling of a user request for issuing a control command and (2) the handling of an issued control command; in particular, as discussed in Sect. 12.4:

(1) The Client Automaton enables or disables user requests that can therefore be forwarded as ClReq or blocked. In the ProcessDone state user requests are blocked to assure consistency of the CPS whereas in the RequestDone state user requests are blocked according to the cooperation mechanism to give priority to other already issued user requests.
(2) A non blocked user request is first forwarded as ClReq by the Client Automaton, to its reference CPCS server and, if accepted, it is then forwarded by this CPCS server to the CCC server which could accept it or not.

On this basis two performance indices are defined: the *blocking probability* ($P_{BLK}$) and the *denial probability* ($P_{DEN}$). The former is defined according to point (1) as the probability that a user request is *blocked* by the client process. The latter is defined according to point (2) since there is probability that a ClReq can be discarded (or *denied*) by the CDN. In particular, different denial probabilities are defined: (i) $P_{DEN}(CPCS)$, which is defined as the probability that a ClReq is discarded by the reference CPCS server; (ii) $P_{DEN}(CCC)$, which is defined as the probability that a ClReq is discarded by the CCC server; (iii) $P_{DEN}(CDN)$, which is also referred as the overall denial probability, is defined as the probability that a client request is discarded at either the *CPCS* Server or the *CCC* Server of the CDN, and is calculated as $P_{DEN}(CPCS) + (1 - P_{DEN}(CPCS))P_{DEN}(CCC)$.

The denial probability should be as low as possible since the server rejection of a client request is always considered a very unpleasant event for the user who generated the control command. In fact, although a user is completely aware that he/she is not always able to control the server, when a request is forwarded to the network, it is very likely that the user will expect to get his/her request accepted. The blocking probability should also be acceptably low since it characterizes the user inability to issue a control request. However, the denial probability is more critical than the blocking probability since users are generally more tolerant of the inability to send a control request than the rejection of a forwarded control request.

### *12.5.2 The Simulation Parameters*

In this section we describe the parameters of the simulation framework and their setting in order to define a realistic simulation scenario which enables the evaluation of HCOCOP on CDN-based control architectures (Sect. 12.4).

Firstly, we consider more general aspects such as the duration of each simulation session and the degree of user activity; then, we focus on those parameters that allow us to characterize CDN-based control architectures (link delays, processing delays, and timers) and Star-based architectures used for comparison purposes.

#### 12.5.2.1 General Aspect Parameters

For each simulation run the duration of the simulation session $T_{SESSION}$ is set to an amount of time that allows for deriving performance values of a pre-determined statistical relevance (i.e. with at least a 0.95 probability that the statistical error is below 5%).

The average inter-arrival time between two successive requests issued by the same user (User Activity) is characterized by the *Mean Request Interarrival Time* (MRIT) which is modeled according to a statistical model based on the *Gamma* probability distribution function [11]. In particular, User Activity is classified as very low (MRIT>=15m), low (10m<=MRIT<15m), medium (5m<=MRIT<10m), high (120s<=MRIT<5m) and very high (MRIT<120s). To enable the complete evaluation of HCOCOP in sessions with high to very high user activity, the value of MRIT was varied within the range $\{10\,\text{s}, 180\,\text{s}\}$.

#### 12.5.2.2 CDN Parameters

The *delay* between two adjacent nodes ($\delta$) is defined according to the following link delay model:

$$\delta_i = K_f\delta_m + N(K_v\delta_m, \sqrt{K_v\delta_m})$$
$$K_f + K_v = 1 \qquad K_f, K_v \geq 0$$

where $\delta_m$ is the mean delay and $\delta_i$ is the instantaneous delay for a given message. $\delta_i$ is the sum of a fixed part and a variable part, and the values of $K_f$ and $K_v$ are the relative weighs of the two parts, with $K_f$ set to 0.7. The variable part of $\delta_i$ is generated by a normal random variable whose mean and variance are set to $K_v\delta_m$. The distribution of the normal variable is truncated at $-K_f\delta_m$ in order to assure that $\delta_i$ cannot assume negative values. The normal distribution is chosen according to the considerations presented by Gibbon et al. [8]. The parameters of the delay model are set according to the values measured in a CDN testbed established across Italy and Spain [7]. In particular, $\delta_m$ is set to 3 ms for the links between a client and its reference CPCS server, and to 61 ms for the links between a CPCS server and the

CCC server. For a fair comparison, $\delta_m$ between clients and the server is set to 64 ms in the considered Star-based architecture.

The *server processing delay* ($T_{PROC}$) is the amount of time taken by a CDN server (CPCS or CCC) or the Star server to serve an accepted request and accordingly change the state of the CPS. $T_{PROC}$ is set to 200 ms.

The *server timers* ($T_{CCC}$, $T_{CPCS}$, $T_{CC}$) are used to control the reactivity of servers ($T_{CCC}$ and $T_{CPCS}$) and the overall degree of system interactivity. They are both set to 3.0s, as is the client timer $T_{CC}$; this setting avoids deadlock situations, as shown by Fortino et al. [6].

### 12.5.3 Operational Modes of HCOCOP

In this section the *NoCoop*, *LocalCoop* and *GlobalCoop* operational modes defined in Sect. 12.4 are analyzed and compared. Moreover, the performances are compared with those obtainable with a Star-based architecture which exploits the COCOP protocol [6]. The Star-based architecture employed, hereby referred to as "Star", is representative of existing collaborative playback architectures which have a centralized nature, as control messages are processed by a single server entity (see Sect. 12.2). The COCOP protocol also operates in two different modes, *cooperative* (*Coop*) and *non-cooperative* (*NoCoop*), and is defined by two automata: the automaton of the COCOP client process, which is similar to the Client Automaton of HCOCOP (see Sect. 12.4), and the automaton of the COCOP server process which resembles the CPCS Automaton of HCOCOP but does not have the Synchro state since there is no need to synchronize with other servers. Moreover, the control protocols employed by the Star-based systems (MASH Rover and ViCRO$^C$, see Sect. 12.3) are similar to COCOP operating in the NoCoop mode that can be considered an archetypal implementation of those protocols and can be effectively used for comparison purposes.

### 12.5.4 Performance Evaluation

The simulation phase aims at evaluating the performance of the HCOCOP protocol in a simple CDN-based architecture with two subgroups and 12 clients, which is a quite large number for a cooperative playback session, since such sessions are mainly intended for small/medium sized groups of users [13].

We first present results achieved in a CDN-based architecture with a symmetric topology; then, we examine the behavior of HCOCOP in an asymmetric topology and in an adaptive scenario in which a client is dynamically redirected from one CPCS server to the other.

#### 12.5.4.1 CDN with Symmetric Topology

A first set of simulations have been carried out in a symmetric CDN with 12 clients and 2 CPCS servers. Due the symmetry of this topology, 6 clients are assigned to each CPCS, as shown in Fig. 12.8.

Figure 12.9 shows the denial probability at the CPCS server, $P_{DEN(CPCS)}$. From the figure the benefits of the cooperation modes are evident. As described in Sect. 12.4, the *LocalCoop* mode disables users to issue control commands when the client process senses a request issued by another client attached to the same CPCS server. The use of this mode significantly decreases the denial probability with respect to the *NoCoop* mode. Benefits of cooperation are further enhanced under the *GlobalCoop* mode, since clients attached to a given CPCS server are also able to detect a request issued by clients attached to other CPCS servers.

Figure 12.10 shows that the denial probability at the CCC server, $P_{DEN(CCC)}$, is not appreciably modified by the cooperation approach. However, due to the improvement at the local group level, the values of overall denial probability, $P_{DEN(CDN)}$, are much lower when the global cooperation mode is exploited (Fig. 12.11). The denial probabilities experienced in the CDN and centralized (or Star) architectures are also compared in Fig. 12.11. Denial probabilities obtained in the CDN with *LocalCoop* and *NoCoop* modes are comparable with the denial probabilities achieved in the Star with the corresponding *Coop* and *NoCoop* modes. However, the denial probabilities obtained in the CDN with *GlobalCoop* are far lower than all other cases. Therefore, the use of CDN architectures, combined with cooperation mechanisms, can actually lead to a remarkable improvement in the ability of a client to control the server.

Even if the denial probability is the main performance index, it is important to verify if this improvement is obtained at the expense of the blocking probability. Figure 12.12 shows that the blocking probability is not significantly affected either by the cooperation mode (no cooperation, local or global cooperation) or by the type of architecture (Star or CDN).

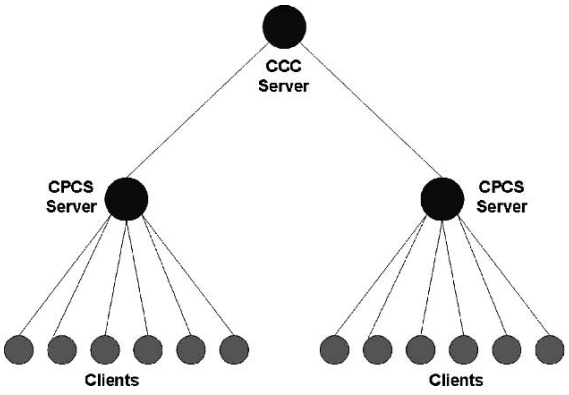



**Fig. 12.8** Symmetric CDN architecture with 2 subgroups and 6 clients per subgroup

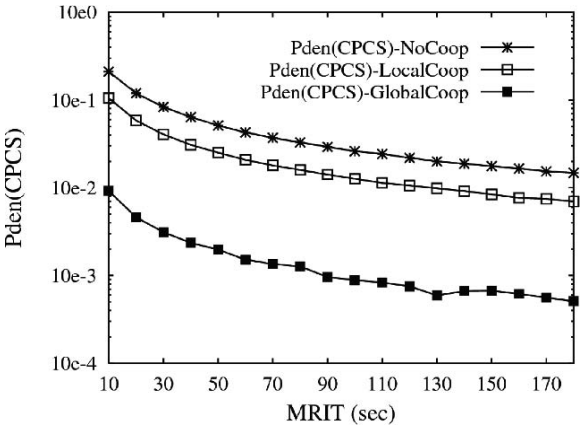



**Fig. 12.9** Denial probability at the CPCS servers: comparison between NoCoop, LocalCoop and GlobalCoop operational modes

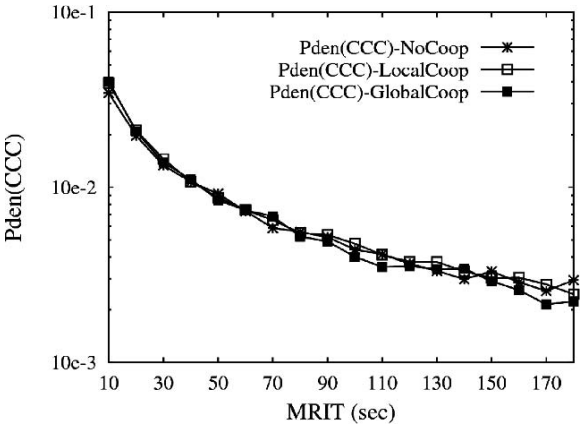



**Fig. 12.10** Denial probability at the CCC server: comparison among NoCoop, LocalCoop and GlobalCoop operational modes

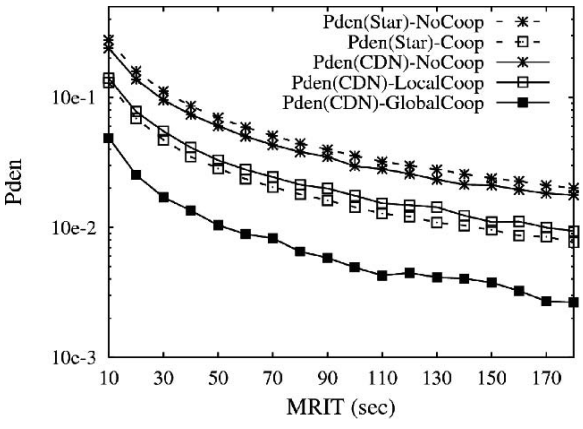



**Fig. 12.11** Overall denial probability: comparison between CDN and Star-based architectures under cooperative and non cooperative operational modes

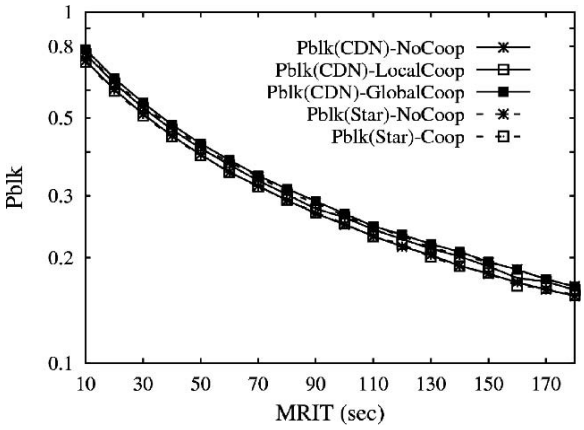



**Fig. 12.12** Blocking probability: comparison between CDN and Star-based architectures under cooperative and non cooperative operational modes

### 12.5.4.2 Asymmetric CDN Topologies and Dynamic Client Redirection

A further set of simulation runs have been carried out to investigate the HCOCOP performance in a CDN architecture in which 12 clients are asymmetrically distributed among 2 CPCS servers (see Fig. 12.13). In particular, 7 clients are allocated to one server and 5 to the other. This topology may be obtained starting from the previously examined symmetric topology, in case that one of the clients is moved (or "redirected") from one server to the other.

To better understand this phenomenon, it must be recalled that in a CDN a *request-routing* algorithm is employed to route a client request to an appropriate surrogate, which in our case corresponds to assigning the client to a specific CPCS server. In case of *adaptive request routing* [15], the surrogate can be dynamically changed according to CDN conditions, which in our case is under examination. The implications of this event are discussed in the following.

Figure 12.14 reports the overall denial probability experienced by the clients belonging to the two subgroups under cooperative and non cooperative modes.

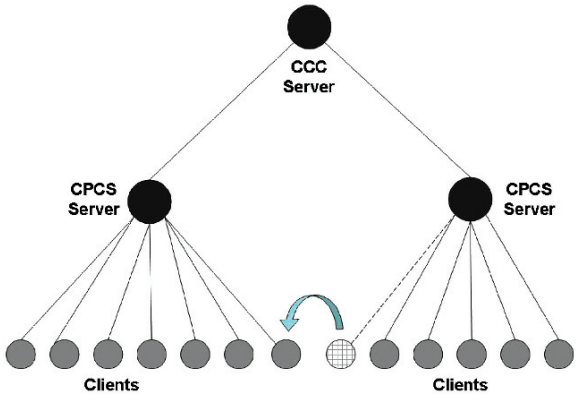



**Fig. 12.13** Asymmetric CDN architecture with 2 subgroups, one with 7 clients and the other with 5 clients

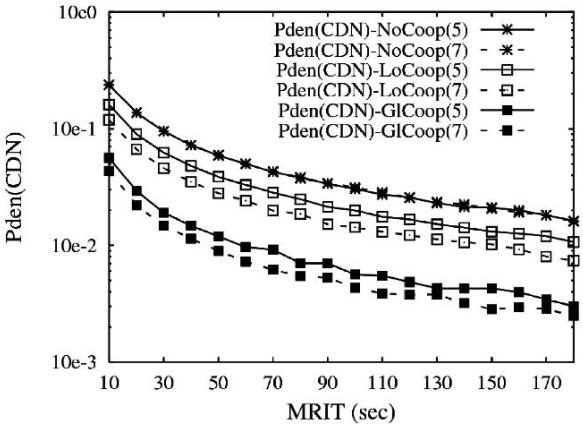



**Fig. 12.14** Overall denial probability in an asymmetric CDN architecture: comparison among NoCoop, LocalCoop and GlobalCoop operational modes

Comparison shows that, with *NoCoop*, no difference in denial probability is found between the two subgroups. On the other hand, under cooperative modes, *LocalCoop* and *GlobalCoop*, the clients that belong to the most numerous subgroups have more chances to control the session state. This phenomenon can be considered a beneficial outcome of the cooperation mechanism; indeed the aggregation of clients in the same subgroup can improve the performance of all the participants of the subgroup. In particular, this phenomenon can be explained as follows. At the local level, the cooperative mechanism allows clients to perceive the requests that are generated by other clients. Therefore, as the number of clients in the same subgroup increases, it becomes easier to avoid issuing the requests that will be probably discarded at the local CPCS server. This benefit balances the drawback that comes from the fact that the level of local concurrency increases with the number of clients. On the other hand, once a client belonging to the larger subgroup gains the control of the local CPCS server, it has a higher chance of controlling the CCC server than a client belonging to the other subgroup. In fact the larger subgroup forwards a higher number of requests to the CCC server; therefore, these requests undergo a lower level of concurrency at the upper CDN level than the requests forwarded by the smaller subgroup. According to this outcome, clients can be profitably redirected to existing subgroups whereas isolated clients or clients belonging to very small subgroups can be penalized.

Moreover, no remarkable differences have been noticed between the blocking probabilities experienced by clients of the 2 subgroups.

Figure 12.15 focuses on the effect of the dynamic redirection of one client from a subgroup to the other, thus passing from a symmetric topology, with 6 clients per subgroup, to an asymmetric one, with 7 and 5 clients per subgroup. As a confirmation of the results shown in Fig. 12.14, the overall denial probability decreases in the subgroup to which the client is redirected and increases in the other subgroup, whereas the denial probability related to the symmetric topology is in the middle. This can also be seen the other way round: if the initial configuration is the asymmetric one, the redirection of a client can be performed to achieve a symmetric topology and this way obtain a better fairness among clients.

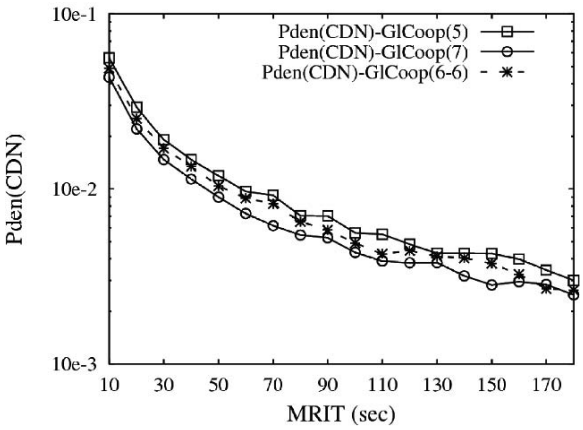



**Fig. 12.15** Effect of client redirection on the overall denial probability: comparison between a symmetric CDN architecture and an asymmetric one resulting after a client redirection from one CPCS server to the other

As opposed to the denial probability, the blocking probability is hardly affected by client redirection.

In conclusion, the purpose of improving the fairness properties of the CDN architecture, with respect to denial probability, can be one of the rationales that drive the request routing algorithm, along with other usual parameters such as network proximity, client-server latency, and load of surrogates. The combination of such parameters is currently investigated with the purpose of defining a routing algorithm that improves not only data delivery, but also the effectiveness of the session control protocols.

## 12.6 Visionary Thoughts for Practitioners

The actual development and deployment of CPSs supported by CDN-based architectures provides the possibility to offer collaborative playback services on the current Internet infrastructure. It can also be enabled for several important application domains ranging from e-Learning to e-Entertainment.

A CDN-based CPS can efficiently support the Collaborative Learning on-Demand (CLoD) e-Learning paradigm [4], a virtual collaborative learning method which enables a self-tutored and interactive learning process where a small group of remotely dislocated students requests, watches, and controls a playback of a lecture and exchanges questions. CLoD borrows some of the ideas of the Tutored Video Instruction (TVI) and Distributed Tutored Video Instruction (DTVI) learning methodologies and tools [14] in which a small group of students driven by a tutor goes over a videotape of a lecture. DTVI is a fully virtual version of TVI, in which each student has a networked computer equipped with audio (microphone and headset) and video (camera) facilities to communicate within a group. TVI and DTVI have proven real effectiveness in that the students involved in their experimentation have been shown to outperform students who physically attended the lectures. The main

difference between CLoD and DTVI is that CLoD does not assume the presence of a tutor which guides students to construct knowledge. In fact, while in DTVI only the tutor has control of the videoconference recorder (VCR), in CLoD each participant of the playback session uses a shared VCR remote controller in a sort of group-supervised collective tutoring.

CDN-based CPSs can also feature e-Entertainment applications such as the Virtual Theaters which are distributed virtual environments where people avatars (virtual alter egos of people) meet and plan to cooperatively watch and control a movie by exchanging comments or chatting with each others.

## 12.7 Future Research Directions

The CDN-based architecture proposed in this chapter is currently being enhanced to increase service effectiveness and efficiency. In particular, the defined HCOCOP currently does not differentiate among control commands; however associating different handling policies to different control commands can result in a more effective control of a cooperative playback session. A multi-policy playback control protocol for Star-based architectures has been proposed by Fortino et al. [5] where the authors have defined three policies (random-based, token-based and voting-based) and respectively associated them to the control commands Pause, Play/Seek and Stop according to their semantics. The handling of the Pause control command requires being highly interactive so that it can be effectively supported by the provided random-based policy of HCOCOP. The handling of the Play/Seek control commands can be supported by a token-based mechanism which allows the token holder to issue the control command. Finally, the handling of the Stop control command, as its acceptance would cause the CPS to be terminated, should be done according a majority criterion so that a voting-based policy can be effectively exploited.

The COMODIN system provides a best-effort media streaming synchronization among the group members of a CPS. Currently synchronization mechanisms at the CDN or at the client site are not offered, which would guarantee a synchronized view of the multimedia session to all clients of the group. Research efforts are under way to define a synchronization mechanism driven by the CDN which will provide more than best effort synchronization of the multimedia playback view without burdening the clients.

## 12.8 Conclusions

This chapter has presented a novel CDN-based architecture that supports collaborative media streaming services and allows an explicitly-formed synchronous group of users to select, watch, and cooperatively control a multimedia session. The control of the playback session is enabled by HCOCOP whose performance was evaluated

through discrete event simulation. Results have shown that the hierarchical CDN-based approach is highly efficient, when compared with the usually adopted Start-based architecture, as denial probability is reduced while blocking probability is not significantly affected. Another interesting outcome is that in asymmetric topologies the clients that are assigned to more numerous groups are better served than isolated clients or clients belonging to very small subgroups. This phenomenon, if combined with other parameters such as network proximity, client-server latency, and load of surrogates can be exploited to tune the request routing algorithm, which is one of the major components of a CDN.

## References


1. Cranor, C. D., Green, M., Kalmanek, C., Shur, D., Sibal, S., Sreenan, C. J., Van der Merwe, J. E. (2001) Enhanced Streaming Services in a Content Distribution Network. *IEEE Internet Computing*, 5(4):66–75.
2. Crowcroft, J., Handley, M., Wakeman, I. (1999) *Internetworking Multimedia*. Morgan Kaufmann Pub, San Francisco, USA.
3. Dommel, H. P., Garcia-Luna-Aceves, J. J. (1999) Group Coordination Support for synchronous Internet Collaboration. *IEEE Internet Computing*, 3(2):74–80.
4. Fortino, G., Nigro, L. (2003) Collaborative Learning on-Demand on the Internet MBone. In: Ghaoui C (ed) *Usability Evaluation of Online Learning Programs*. Idea Group Publishing, Hershey (PA), USA, pp 40–68.
5. Fortino, G., Mastroianni, C., Russo, W. (2004) A Multi-Policy, Cooperative Playback Control Protocol. In *Proc. of the 3rd IEEE Int'l Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, pp 297–302.
6. Fortino, G., Mastroianni, C., Russo, W. (2005) Cooperative Control of Multicast-based Streaming On-Demand Systems. *Future Generation Computer Systems, The International Journal of Grid Computing: Theory, Methods and Applications* 21(5):823–839.
7. Fortino, G., Russo, W., Mastroianni, C., Palau, C., Esteve, M. (2007) CDN-supported Collaborative Media Streaming Control. *IEEE Multimedia*, 14(2):60–71.
8. Gibbon, J. F., Little, T. D. C. (1996) Use of Network Delay Estimation for Multimedia Data Retrieval. *IEEE Journal on Selected Areas in Communications*, 14(7):1376–1387.
9. Holfelder, W. (1998) Interactive remote recording and playback of multicast videoconferences. *Computer Communications* 21(15):1285–1294.
10. Molina, B., Palau C. E., Esteve, M., Alonso, I., Ruiz, V. (2006) On Content Delivery Network Implementation. *Computer Communications*, 29(12):2396–2412.
11. Padhye, J., Kurose, J. (1999) Continuous Media Courseware Server: a Study of Client Interactions. *IEEE Internet Computing*, 3(2):65–72.
12. Raman, S., McCanne, S. (1999) A model, analysis, and protocol framework for soft state-based communication. *ACM SIGCOMM Computer Communication Review*, 29(4):15–25.
13. Schuett, A., Raman, S., Chawathe, Y., McCanne, S., Katz, R. (1998) A Soft State Protocol for Accessing Multimedia Archives. In *Proc. of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, pp. 29–39.
14. Sipusic, M. J., Pannoni, R. L., Smith, R.B., Dutra, J., Gibbons, J. F., Sutherland, W.R. (1999) Virtual collaborative learning: a comparison between face-to-face Tutored Video Instruction (TVI) and Distributed Tutored Video Instruction (DTVI). (Technical Report N. SMLI TR-99-72 by Sun Microsystems Laboratories, Palo Alto, CA, USA).
15. Wang, L., Pai, V., Petersen, L., (2002) The effectiveness of request redirection on CDN robustness. *ACM SIGOPS Operating Systems Review*, 36:345–360.

# Chapter 13
# CDN for Live and On-Demand Video Services over IP

Mirosław Czyrnek, Ewa Kuśmierek, Cezary Mazurek, Maciej Stroiński, and Jan Węglarz

## 13.1 Introduction

Nowadays services such as Video-on-Demand and live TV programming available over broadband IP networks become reality. Users want to have access to high quality video at any time, wherever they are, on a device that is available at the moment and in an interactive way. Traditional TV distribution platforms cannot satisfy these requirements. Delivery over broadband IP networks on the other hand, allows providers to offer value added services with an opportunity for truly interactive content access. The main challenge in the design of a large-scale multimedia delivery system over IP is the aggregate volume of data to be delivered and the high magnitude of the aggregate transmission rate. Progress in signal processing allows for high quality signal to be delivered to the end users without imposing excessive bandwidth requirements. 1 Mbps is considered sufficient to obtain a reasonable quality. However, providing service to thousands of users at the same time poses a challenge.

Content Delivery Network (CDN) is a solution that has been successfully used in systems such as World Wide Web. Advantages of CDNs for rich multime-

Mirosław Czyrnek
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61-704 Poznan, Poland, e-mail: majrek@man.poznan.pl

Ewa Kuśmierek
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61-704 Poznan, Poland, e-mail: kusmiere@man.poznan.pl

Cezary Mazurek
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61-704 Poznan, Poland, e-mail: mazurek@man.poznan.pl

Maciej Stroiński
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61-704 Poznan, Poland, e-mail: stroins@man.poznan.pl

Jan Węglarz
Poznan Supercomputing and Networking Center, ul. Z. Noskowskiego 12/14, 61-704 Poznan, Poland, e-mail: weglarz@man.poznan.pl

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

dia accessible over broadband IP networks seem obvious. However, there are differences between a CDN used for traditional Web content and a CDN designed specifically for multimedia content. CDNs for Web content typically support only straightforward delivery of low-quality streams. The multimedia CDNs on the other hand, aim at delivery of content with quality that can compete with traditional broadcast media, and support for sophisticated services [6, 7]. The special consideration required for delivery of digital video and audio content is due to the specific characteristics of this type of content.

In this chapter, we address the key aspects of the multimedia CDN design based on the presentation of iTVP, a platform which is built for IP-based delivery of multimedia content on a country-wide scale to a large number of concurrent users. The platform services include access to live TV programming, video-on-demand and audio-on-demand, time shifting, Electronic Program Guide (EPG), and Personal Video Recorder (PVR). Given the intended range of operation and the nature of the content offered, efficient content delivery is necessary for the successful operation of the platform. Therefore, CDN is one of its key components.

The key characteristic of our CDN is the hierarchical nature of the system where replica servers are placed at the higher level and the lower level nodes are located in the last mile operators' networks. The entire CDN is divided into autonomous but cooperating regions. We make use of content replication integrated with content caching, and take advantage of cooperative replication at the higher CDN level. Content delivery is performed in a file download mode to lower level nodes and in streaming mode from lower level nodes to the end users. Such an approach allows us to adopt a relatively simple traffic model, as compared to models constructed for a VBR-encoded video, and hence simplifies bandwidth management. We use central content directory within each region as a solution for content location. CDN monitoring subsystem provides necessary information to select nodes best suited to provide services to a given user.

We present iTVP content delivery architecture, its functional structure, and principles of operation. We explain the rules of replica server placement, content allocation and distribution, and user request routing. The CDN is characterized in terms of mechanisms ensuring efficient resource usage, scalability, and reliability. We address requirements for resources such as bandwidth and storage, especially important for high quality multimedia streaming and examine the resulting user perceived Quality of Service (QoS).

The chapter is organized as follows. In Sect. 13.2 we present the background information and the related work on multimedia CDNs. The introduction to iTVP with the general description of the entire platform is presented in Sect. 13.3. The following three Sects.: 13.4, 13.5, and 13.6, concentrate on the key aspects of CDN functionality, i.e. replica server placement, content allocation and distribution, and user request routing. The CDN description is followed by presentation of operational data collected in the system and illustrating its performance in Sect. 13.7. We describe the directions of future research in Sect. 13.8 and share our thoughts on the future of multimedia CDN in Sect. 13.9. We conclude the chapter in Sect. 13.10.

## 13.2 Background and Related Work

CDNs for multimedia content differ from traditional CDNs which are mainly used for delivery of Web objects. Multimedia content has characteristics that are significantly different from other types of content and the nature of multimedia content affects a number of design decisions such as CDN topology, number and locations of replica servers, content allocation and distribution. In this section we present a brief description of research concentrating on various aspects of multimedia CDN design and operation. The multimedia features that are relevant in the context of a CDN can be roughly divided into two groups: characteristics of multimedia content objects such as typical data volume and encoding techniques, and delivery and content access modes.

### *13.2.1 Multimedia Characteristic Influence*

We start with the multimedia characteristic's influence on the CDN total cost computation, more specifically with the influence on the bandwidth related distribution cost and storage cost. Multimedia objects are usually much larger than Web objects. The video size depends on a number of parameters such as resolution and type of encoding, but typically an hour long video size is on the order of several hundred MBs to several GBs. Furthermore, there is no one target encoding rate which would be suitable for all users. The higher the rate the better the video quality. Thus, the rate should be maximized subject to the available bandwidth. Video is typically encoded in such a way that there are multiple quality versions to choose from, depending on the available bandwidth. Such an approach further increases the video object size. Consequently, replica servers can no longer store all content accessed by users. Users may have to obtain various objects from various replica servers, as opposed to one closest server. Storage must be considered as a resource that is limited. It becomes an important component of the total cost when formulating replica server number and placement problems. Furthermore, large objects incur higher delivery costs in terms of bandwidth usage. Thus, cost of distribution within CDN, i.e. from the origin server to replica servers and from one CDN node to another, should be considered in addition to cost of content delivery from CDN nodes directly to the end users. Yang et al. [22] analyze the influence of the number of replica servers on such two-step delivery cost in multimedia CDNs. They concluded that too many replicas cause the distribution cost within CDN to be shared by fewer users and hence, the excessive number of servers increases the cost beyond the optimal value. Therefore, it is important to carefully choose the number of replica servers.

Another feature of multimedia content that is important in the context of CDN is related to video encoding schemes. Multimedia content does not change frequently as it is the case for Web pages. Therefore, the problem of cache consistency is not of great importance. However, since multimedia content delivery has certain resource requirements, mostly with respect to the available bandwidth in case of

content streaming, there may be multiple quality versions of the same content to be made available through CDN. Specifically, content may be coded in a layered fashion, giving users an option of receiving as many layers, and as high video quality, as bandwidth availability allows. The layers are ordered in such a way that each given layer increases video quality but only when combined with all previous layers. Therefore, the layers cannot be treated as separate objects. A decision has to be made on replication of each layer. Su et al. [20] deal with a replication method for layered multimedia that targets reduction of content access time and storage costs.

### *13.2.2 Multimedia Delivery and Access Modes Influence*

Multimedia content is typically delivered in streaming mode that allows users to play content concurrently with its reception after only a short delay, as opposed to content download. Using streaming as a delivery mode has important consequences for CDN operation. There are a number of techniques that are used to improve scalability of content streaming systems [1, 10, 13, 14, 15, 19]. They are generally based on multicast transmission mode and take advantage of the server bandwidth usage reduction when one stream is delivered to a number of users. The use of multicast techniques, be it at the network or application level, changes the character of the delivery cost dependency on the number of replica servers. In a unicast environment, the total *server bandwidth* does not dependent on the number of replica servers or their locations and only on the number of concurrent users, since there is one stream per user. In a multicast environment, the total server bandwidth usage increases with the number of replicas since requests are spread over a larger number of servers and one stream is generally directed to fewer users.

The total *network bandwidth* is distinguished from the total server bandwidth in that it accounts not only for the aggregate transmission rate but also for the distance over which content is transmitted. Then, in the unicast case, the total network bandwidth decreases with an increase in the number of replica servers when shortest path routing is used. In the multicast case network bandwidth also depends on the number of users receiving each stream. A longer path but shared by a larger number of users can result in lower network bandwidth usage.

The problem of minimization of the total delivery cost, including the total network and total server bandwidth, under the assumption that multicast streaming is used for content delivery from replica servers to users, is considered by Alemida et al. [2, 3] who show that conventional unicast content delivery systems yield costs much higher than the optimal value. They address the problem of finding an optimal number of replicas, replica placement and request routing that result in near optimal cost in a multicast enabled system. Consideration of the number of replica servers and their location influence on the total network, and server bandwidth usage in the system utilizing proxy-assisted periodic broadcast is presented by Kusmierek et al. [18]. The authors show that server bandwidth usage is minimized with just one replica server and generally increases as the number of replica servers increases. On

the contrary, network bandwidth requirement decreases with an increase in the number of replica servers.

Application of scalable streaming techniques has influence not only on the number of replica servers but affects also content allocation that yields minimal cost. The problem of content allocation in hierarchical CDNs where mechanisms that apply multicast, partial delivery, and use out-of-order delivery of movie segments, is considered by Griwodz [11]. The author determined that the cost-optimal placement decisions, i.e. decisions that efficiently use multicast, may affect QoS provided to users by placing popular movies further away from the end users than less popular ones. Such a negative influence on the QoS can be limited with a modified cost computation and proper selection of the stream merging mechanism parameters.

The resource usage in multimedia CDNs exhibits variability on various time scales. Specifically, TV broadcast based on CDNs show diurnal changes since typically there are fewer users in the morning than in the evening hours. Given the typical sizes of multimedia objects, the magnitude of changes resulting from variability in the number of concurrent users is much higher than in the case of Web objects. Cahill et al. [5, 6] concentrate on dynamically changing resource requirements in such a system under the assumption that CDN operates over a shared rather than dedicated private network. Hence, resources used for content delivery can be leased from service provider and released as needed. Consequently the authors present a CDN model that does not include start-up costs but is based on the assumption that the number of replica servers needed for content delivery can change as needed, and defines dynamically changing storage space and bandwidth requirements.

Finally, multimedia content is often accessed in a non-sequential way, as users perform VCR-like operations. Such type of content access may influence content allocation. If it is not possible to replicate the entire content repository at each replica server site due to storage space limitations, it may be beneficial to store certain video segments close to the end users. It has been shown that such an approach applied to video prefixes reduces access delay and limits network status influence on content playback. Enabling VCR-like operations may require replication of video segments that are spread over the entire content [12] allowing user to jump to another video part without long access delay. Such an approach affects content allocation and user request routing rules.

## 13.3 iTVP Platform

We describe our approach to multimedia CDN design by presenting iTVP platform. Before we move on to its CDN related features, we provide a functional and organizational description of the platform.

iTVP is a multimedia delivery platform that was designed for operation on a large country-wide scale with the potential of serving selected content to any Internet user. The system is built for delivery of the following services: transmissions of live TV programming, time shifting of live transmissions, video-on-demand,

audio-on-demand, and accompanying services such as Electronic Program Guide and Personal Video Recorder. The platform is designed to deliver content from multiple independent content providers and with cooperation of last mile network operators. Content can be made available in various encoding formats and delivered to a variety of end devices including PCs, Set-Top-Boxes, and handheld devices.

Content delivery system is one the key components of the platform which includes also content provider systems, a number of interactive portals serving as primary access points for the end users and license management system (Fig. 13.1). Content provider system functions as a CDN origin server and is responsible not only for storing all contents of a given provider but also for content encoding, description with metadata, verification, licensing, and publishing. License management system is designed to support various licensing scenarios and various payment methods in cooperation with interactive portals.

Functionally, the iTVP CDN consists of the following subsystems: distribution, monitoring, management, and reporting subsystem. The distribution subsystem is responsible for distribution and delivery of multimedia assets. It performs content caching and live stream relaying as well as resource management and access control. To work effectively, distribution subsystem uses system nodes parameters gathered on-line and processed by monitoring subsystem. Thus, the monitoring subsystem is mainly responsible for delivering up-to-date information about various aspects of system operation including hardware performance counters, network interface parameters, application level load, service performance indicators, and nodes availability. Collecting these parameters enables global system load balancing, failure detection, and evaluation of system performance. The management subsystems task

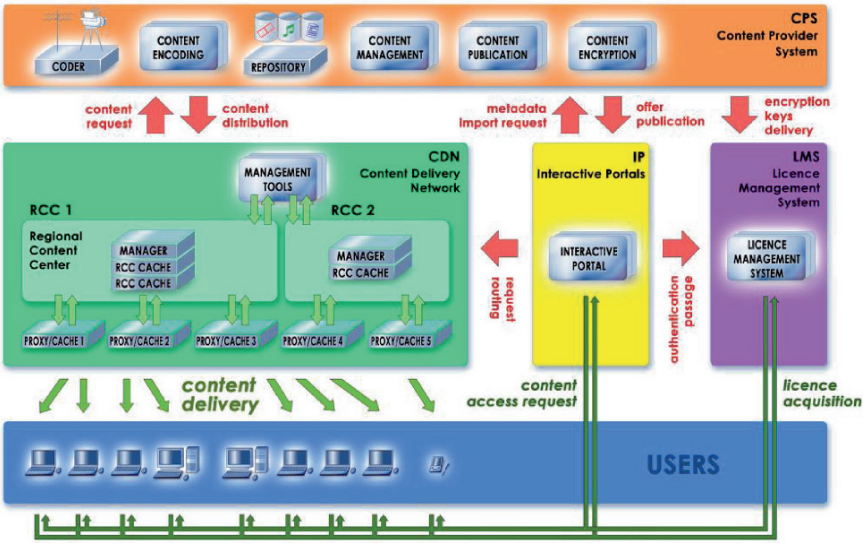



**Fig. 13.1** iTVP platform components

is to store and provide information about CDN nodes configuration, internal services access points, network level configuration, and external services provided by other platform components such as content provider systems. The reporting subsystem collects real-time monitoring parameters gathered by monitoring subsystem and stores the events occurring during system operation for system performance analysis. The iTVP CDN subsystems are distributed and deployed in all system nodes and cooperate with each other to provide system services.

Organizationally, there are three types of entities involved in the content delivery. First, there can be a number of independent content providers that constitute the primary sources of multimedia content. The second entity is the CDN operator that oversees the system functioning. This role is performed by Poznan Supercomputing and Networking Center which is also an operator of a country-wide all optical network backbone in Poland, called PIONIER. And third, there are a number of independent network providers (ISPs) whose participation ensures that their user access to the platform services.

The iTVP platform prototype has been deployed at the beginning of 2006 [8, 17]. The fully operational phase, in terms of functionality, has been reached at the beginning of 2007. The platform is steadily increasing its programming and service offer which results in a steady increase in the number of users. Initially the content repository offered several thousands of content items. This number has increased by almost an order of magnitude toward the end of 2007. The number of unique IP addresses to which content is delivered monthly, grew from around a hundred thousand in the first half of 2006 to more than half of a million toward the end of 2007. Currently, there are two content providers: public national TV (TVP) and public radio. The programming offer includes live transmissions, which are either conducted concurrently with or prior to on-air transmissions, or prepared exclusively for the Internet platform, and on-demand content including archived versions of live transmissions. Hence, the content repository is expanding by including newly produced content. It also will be growing due to digitization of the archival TV programs.

## 13.4 iTVP CDN Architecture

Typically, the problem of selecting the number and locations of replica servers in a CDN is solved to minimize the cost of CDN operation reflecting the resource requirements. Another goal to consider is the optimization of user-experienced QoS. Better QoS requires potentially more resources to be utilized in delivering content, hence, these are two conflicting goals. In case of multimedia content, QoS is defined differently than for other types of content, and ensuring certain QoS plays a very important role in multimedia CDN planning. Access time is just one QoS parameter. Other important factors include the continuity of multimedia playback and multimedia quality determined by video resolution and audio quality. User may be willing to wait longer to receive a Web page, but most users will not tolerate video playback that is interrupted one time after another. Therefore, many iTVP CDN de-

sign decisions were made to balance QoS offered to user and the delivery costs, but under the assumption that certain minimum QoS must be ensured.

### *13.4.1 Two-Level Hierarchical Design*

The first of the CDN design choices we present is the replica server number and placement problem. Our solution to this problem is determined by a number of factors. The most important of these factors are the intended range of operation, the types of content and services to be provided, and the underlying transport network topology. We now comment on each of these factors.

The iTVP platform was designed to provide services to users on a country-wide scale. As mentioned in the previous section, selected content is accessible to any Internet user; however, the content delivery system is designed primarily for users located within the country. Under this assumption, it is clear that the replica servers should be placed in the areas of high concentration of the end users, namely large metropolitan areas, and that the server geographic distribution should match demographic distribution subject to the topology of the network used for content delivery.

The character of content offered by iTVP, namely multimedia live and on-demand content, determines the delivery cost and the storage cost. Transmission of large multimedia files requires a lot of bandwidth between origin server and replica servers as well as between the replica servers and the end users. Placing replica servers close to the users lowers the bandwidth related delivery cost considerably; however, such an approach increases the number of replicas and consequently increase distribution cost within CDN, as well as the storage cost. Therefore, the distance between origin servers and replica servers should be such that the total cost is minimized and well balanced between distribution bandwidth, delivery bandwidth, and storage costs.

The on-demand content is delivered to users in a streaming mode, unless file download mode is available for a given content and is selected by a user. Content streaming reduces access time and storage requirements at the end systems but has stringent network requirements, such as available bandwidth and delay jitter. Hence, edge delivery, i.e. delivery by servers placed at the edge of the backbone network, with suitable QoS is difficult technically and expensive economically. An optimal way to avoid problems with content streaming is to exclude WAN segments from the delivery path, i.e. to use servers placed in access (local) networks to stream content directly to the end users. Such an approach can be implemented with the cooperation of last mile operators, i.e. ISPs. The incentive for an ISP to participate in content delivery is to ensure better QoS for its users and to reduce bandwidth cost for incoming traffic.

The above argument led to a two-level hierarchical architecture of a CDN for iTVP platform. The components of the CDN architecture shown in Fig. 13.2 are gradually introduced throughout this section. The upper level nodes are the replica

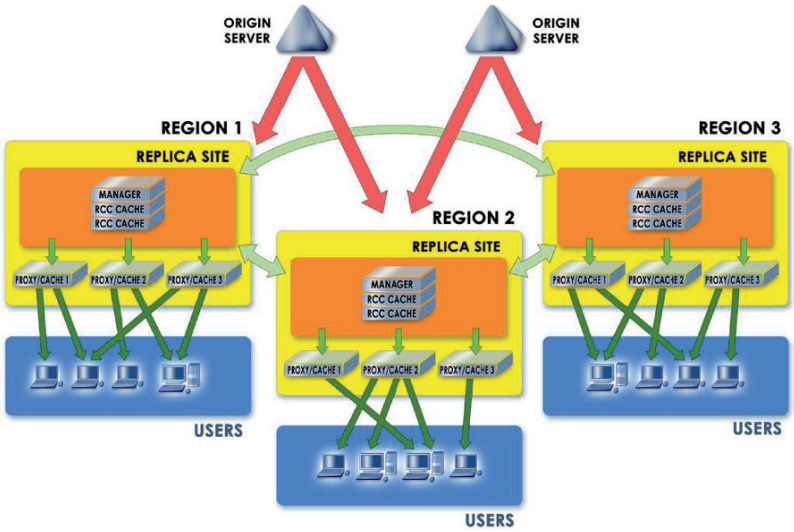



**Fig. 13.2** iTVP CDN architecture

servers, which we refer to as *caches*, placed at strategic places all over the country. The lower level nodes, *proxy/caches*, are located in various ISPs' networks and provide services directly to the end users to ensure proper QoS. The introduction of upper level nodes allows us to reduce bandwidth related distribution costs and the origin server load. Given the two-level architecture, the content is typically obtained from the content provider repository, i.e. the origin server, transferred to a cache node, from cache transferred to a proxy/cache and then delivered to the end users. As we will shortly see, there are also other delivery paths possible. In order to distinguish various phases of the delivery process, we refer to content transfer through the CDN all the way to proxy/caches as *content distribution*. The last phase of delivery from a proxy/cache to the end user is referred to as simply *content delivery*. Consequently, we distinguish distribution and delivery cost.

Content is delivered to users typically in streaming mode. However, streaming takes place only for content delivery from proxy/caches to end user, excluding live transmissions which are streamed along the entire distribution path. Distribution of on-demand content within CDN to proxy/caches is carried out in a file download mode. We justify this approach in the next section while describing content allocation and distribution rules. Consequently, the scalable content streaming techniques are not utilized and cost dependency on the number of servers determined for unicast distribution model applies. Specifically, network bandwidth related distribution cost decreases with an increase in the number of replica servers. Server bandwidth needed for content delivery does not depend on the number of servers. However, storage cost increases with the number of servers.

### 13.4.2 CDN Node Placement

Metropolitan Area Networks (MANs) are good potential locations for hosting replica servers due to their connectivity and bandwidth availability. A single replica site can host a number of caches. They are all seen as a single higher level cache pool by proxy/caches connected to this replica site and by nodes in other replica server locations. Hence, from now on we use the term replica server, or Regional Content Center (RCC) in iTVP terminology, to refer to a replica site that is a cluster of caches. Placing multiple caches in one replica server location increases available resources, mainly server bandwidth and storage, and improves service reliability.

The number of proxy/caches depends on the number of ISPs that cooperate with the CDN. Each ISP can have a number of proxy/caches depending on the size of the network (number of potential users). For reliability purposes it is recommended that there are at least two proxy/caches available regardless of the network size. Proxy/caches are connected to the nearest, in terms of core network topology, replica server to minimize distribution costs at this stage.

One replica site and all proxy/caches connected to this replica constitute a region. The operation of regions is not coordinated by any central entity. Each region has a regional manager node and a certain degree of autonomy. However, regional managers are aware of the existence of other regions and cooperate in performing content distribution tasks. Content can be distributed from one region to another but only within the higher level of the hierarchy, i.e. between caches. Cooperation between lower level nodes has justification only for nodes within one ISP's network. In addition, proxy/caches obtain content only from the replica server within their own region.

As the system grows, new sites are added to serve the country regions more effectively, balance the internal CDN network traffic and increase total caching space. The location of a new replica server is determined by the network topology, analysis of the network traffic and CDN operation parameters. Depending on the new replica server location, some of the proxy/caches may be reallocated to the new region and new ones may be added. An increase in the number of replica sites decreases network bandwidth related distribution cost as distance between proxy/caches and caches decreases. Bandwidth requirements for the origin server does not increase due to the cooperation among various replica sites. One replica server can obtain content from another replica instead of the origin server. The locations of the new replica sites are determined partly by the network topology.

### 13.4.3 Network Level Configuration

Content distribution is carried over PIONIER [4], which is country-wide Polish all-optical network based on DWDM technology. PIONIER connects 21 Metropolitan Area Networks (MAN), which are potential sites for hosting replica servers




**Fig. 13.3** PIONIER topology

(Fig. 13.3). The set of caches contains nodes connected with Gigabit Ethernet technology to PIONIER infrastructure with 1 Gbps channels dedicated to CDN traffic. The proxy/caches located in ISPs' networks are connected to replica servers with 100–400 Mbps dedicated channels. For the internal CDN communication and content distribution, layer two links are configured, forming a virtual network dedicated for the CDN traffic. Virtual Local Area Networks (VLANs) enable separation of the CDN-related traffic from other applications operating in the network, provide better QoS and high level of security for content distribution, which is very important for the multimedia industry.

## 13.5 Content Allocation and Distribution

Performance of a CDN depends not only on the number and location of replica servers, i.e. CDN configuration, but also on the content allocation. Determination of how content is stored in a CDN, in how many of available locations, has impact on the total cost through the storage space component, and on the user perceived quality. We now explain our CDN design choices for content allocation and distribution, keeping in mind iTVP CDN topology presented in the previous section.

### *13.5.1 Content Distribution Modes*

In iTVP CDN content allocation rules are partly determined by the fact that users acquire content from one of proxy/caches located within their ISP's network. Since only proxy/caches provide content directly to the end users, origin servers as well as replica servers are not visible, every requested object must be stored by at least one of these proxy/caches. The rule is that users obtain an entire content from one node. A switch to another proxy/cache during an ongoing content playback is done only in a rare case of a node failure. A set of objects stored by proxy/caches within an ISP's network is therefore determined mainly by users' requests as in a standard caching system. Consequently, the number of content replicas and their location at the ISP network level within CDN, depends directly on user requests. Typically, each content is stored at a predefined number of proxy/caches in an ISP's network. Currently, this number is set to two nodes.

Content distribution in the iTVP CDN can be characterized as content caching integrated with content replication. Content is distributed to a proxy/cache in a user-initiated *pull* mode in case of a cache miss, i.e. reception of a request for content that is not available at any proxy/cache within given ISP's network. In addition, a *push* mode can be initiated by a content provider or the CDN operator and is used when a new content is published and a high popularity is anticipated, or when content popularity is expected to increase due to a diurnal change in the access pattern. Distribution in a push mode, or replication, is intended to decrease cache miss ratio and consequently, reduce content access time. Distribution in a pull mode, adjusts content availability to match content popularity. Very popular content is stored by a large number of nodes, while less popular content may be available only at a few selected proxy/caches. Each proxy/cache uses one storage space for both content that is cached and content that is replicated. Such a solution to distribution allows the CDN to take advantage of the combined approach.

Given the character of the content offered in iTVP, proxy/caches cannot store all contents requested or replicated over a longer period of time. The storage space has to be managed to ensure high hit ratio and to eliminate cases of service denial due to lack of storage space. The strategy adopted for cache replacement can be characterized as a modified LRU. The modification takes into account 'content size', since a popular but small object can be preferred for replacement over a less popular but larger object. Consideration for content size reduces distribution costs in terms of bandwidth used, which is especially important given large object sizes. The mode of distribution used for a given content, i.e. caching or replication, is not taken into account in the content selection for replacement. Since the replacement procedure ties a lot of system resources, it is performed periodically and it frees storage space up to a predefined level. Selection of the replacement frequency and the level to which storage space is freed has significant impact on the CDN performance. Freeing too much space increases distribution costs and cache miss ratio, freeing not enough space may result in service denial.

Proxy/caches acquire content from higher level nodes in their region. Content distributed to a given replica site is stored in at least one cache node. In order to

minimize storage requirements, each content can be stored at only one cache within a region. This is the strategy currently implemented in iTVP CDN. Having more than one content replica at a higher level within a region increases service reliability and server bandwidth available for distribution to proxy/caches, but it also increases storage cost. Hence, the choice of the number of content replicas stored at caches is made to balance storage and bandwidth costs. Content is distributed to caches either from the origin server or from cache in another replica server. CDN regions cooperate on content distribution in both push and pull mode, thus implement ocooperative content replication and caching. Such cooperation further reduces origin servers load and distribution costs. Cooperation between proxy/caches is limited to a set of nodes within one ISP's network. There is no incentive for one ISP to provide content to another ISP's network. Within one ISP's network content could be distributed from one proxy/cache to another proxy/cache but such transmission would increase proxy/cache load and could impact QoS provided to the end users. Storage space management at replica server sites is performed in a way similar to proxy/cache space management except for the fact that the cache replacement procedure for caches is performed not periodically but when free space is needed.

### *13.5.2 Content Transmission Modes*

On-demand content is distributed in a file mode typically with a speed much higher than playback rate from the origin server all the way to proxy/caches. Streaming takes place only for content delivery from a proxy/cache to an end user. Live content, such as live TV programs, are streamed by content provider's servers to replica server and relayed by replica server to other replica servers. Server replicas in each region relay stream to proxy/caches, which in turn provide stream to the end users. Hence, live content distribution can be viewed as application layer multicast with CDN nodes forming an overlay network. File mode distribution for on-demand content has several advantages. With high transmission speed the entire content is available sooner for streaming, thus allowing user to perform VCR-like operations. With streaming taking place only within ISP's network, the QoS provided to user does not depend on the network conditions within CDN. The traffic model for file mode distribution is much simpler than for streaming and greatly simplifies network bandwidth management. Two mechanisms are employed to further reduce the content access time. First, distribution through the CDN hierarchy is done in a cut-through or pipelined manner, i.e. transmission from an origin server to a replica server (or between two replica servers) is done in parallel to the transmission between replica server and proxy/cache. A cut-through distribution limits the influence of the distribution path length on the time needed for distribution. Second, streaming from proxy/cache to a user can be performed in so called QuickStart mode, in which video file is not transmitted sequentially but in a way that allows streaming to start when the minimum amount of information is available, sometimes long before the entire file transfer is completed. The QuickStart mode makes the time needed for content

distribution from the user's standpoint, i.e. time needed to start content streaming practically independent of the content size.

### *13.5.3 Flash Crowd Handling*

One of the important aspects of CDN operation is its ability to handle flash crowds [16]. Publication of a content that is of interest to a large group of users can easily cause a sharp rise in the number of user requests received over a short period of time and consequently a sharp rise in the number of concurrent users. Such a situation is common for live transmission of interesting events. Typically, such transmissions are scheduled in advance and the distribution is performed also in advance of the event start. If such a sharp rise in the number of requests was not anticipated, then the first few requests cause content distribution to a large percentage of proxy/caches. The distribution tree rooted in the origin server is quickly established. Due to pipelined content transfer and QuickStart mode, content streaming from proxy/caches can start after a relatively short period of time. The subsequent user requests are spread over the available proxy/caches, where the content is already available. Regional managers have high request processing throughput since all information needed for routing is available locally.

## 13.6 User Request Routing

User request routing is a CDN function that complements CDN topology and content allocation decisions. Rules for selecting a node which should provide requested content to a user, is partly determined by the CDN topology and content availability. Typically, there is a set of nodes that provide service to a given user. Multimedia content again distinguishes itself from other types of content, with respect to the request routing approach. Given the network requirements of streaming delivery, the delivery path parameters are one of the most important factors in the node selection process, next to the server load. These parameters include not only the distance to be traversed but also the available bandwidth and delay jitter. In this section we list iTVP CDN node selection criteria and describe the mechanism used for user request routing.

### *13.6.1 Node Selection Criteria*

Selection of a node to provide service to a given user identified by an IP address is performed based on several criteria. These criteria include user's location with respect to proxy/caches, content availability, and nodes' load. The strategy is to

create a set of nodes that can provide services to a given user based on the primary criterion and then narrow this list down to a predefined number of nodes based on the secondary criteria.

The initial set of nodes that can provide services to a given user is determined based on the user's IP address and mapping established between IP subnetworks and proxy/caches. The mapping reflects user location in some ISP's network. The mapping between IP subnetworks and proxy/caches is created based on the list of ISPs cooperating with the CDN. In addition to proxy/caches placed in various ISPs' networks, there are also proxy/caches that can potentially provide services to an arbitrary Internet user. These proxy/caches are connected to either the backbone or to a MAN with 1 Gbps channels. They are referred to as *default proxy/caches* as opposed to *dedicated* nodes in ISPs' networks. The default proxy/caches are intended to provide access to iTVP services to users that are not clients of any ISP associated with iTVP platform, and to provide access to the content that is not available on any of the proxy/caches in the ISP's network of a user. The latter role is to reduce the access time in case of a cache miss. Recall that a cache miss triggers content distribution to proxy/caches. If the requested content is available at any of the default proxy/caches, the user request may be routed to one of them while content distribution is started to dedicated proxy/caches. Therefore, the set of nodes selected based on a user's IP address, typically includes dedicated and default proxy/caches.

Given that there may be, and typically there is, more than one proxy/cache in an ISP's network, the next criterion for node selection is the requested content availability. Such an approach minimizes content access time and reduces distribution costs. Content availability or content location is determined based on the information stored in a content directory maintained in each region. Hence, any content can be quickly located by a regional directory look-up. Each regional manager maintains such a directory and also acts as a directory server for other regions. The price paid for a quick content look-up is mostly related to directory update overhead. However, the regional manager supervises operation of all nodes, including content distribution. Thus, most content location updates are performed as tasks accompanying content distribution and as such impose no communication overhead. Updates necessary due to content removal from a node resulting from cache replacement procedure, do not constitute a significant overhead. A central content directory, or a central manager, usually constitutes a single point of failure, which is considered to be the most serious drawback of such a solution. In order to ensure reliability, there is more than one cache in each CDN replica server site, which is capable of taking over the manager and directory server tasks. Thus, it is ensured that data vital to managing the region is not lost in a case of the manager failure.

Out of all nodes that can provide services to the user, the nodes where the requested content is available are selected. If there are still more than one node satisfying these criteria, proxy/cache with the smallest load, i.e. the smallest number of concurrent users, is selected with priority given to dedicated proxy/caches. Given that static path properties between the end users and each of the proxy/caches located in a given ISP's network can be considered similar; and dynamic properties depend on the load, the number of concurrent users reflects well both server load

and available bandwidth. More precisely, for reliability purposes, the list of nodes is narrowed down to more than one proxy/caches, typically two, and user receives two addresses where the requested content can be accessed. If there are no nodes that have the requested content, including default proxy/caches, a miss occurs and content distribution is started.

### *13.6.2 Request Redirection Mechanism*

The node selection procedure and user request routing is performed by regional managers. However, users do not interact with the managers directly. A number of independent portals ensure access to iTVP services for the end users. A user request is directed by a portal to a regional manager for further processing, i.e. for proxy/cache selection. The mapping between IP subnetworks and proxy/caches translates into mapping between IP subnetworks and CDN regions. Hence, for a given user's IP address there is one region that should provide service to this user and regional manager receiving the request can forward it to an appropriate region. Since portal acts as a middle man between end user and CDN in obtaining access to content, the regional manager's response to the request is sent back through the portal. It is either a response containing proxy/cache addresses or a 'wait' response in case of cache miss.

## 13.7 iTVP CDN Performance Evaluation

Based on the data collected by the CDN reporting subsystem we compute a number of metrics to evaluate the CDN performance and user perceived quality. First, we describe the system configuration, main content provider repository and the CDN load observed during the period of time for which data was collected. Next, we evaluate distribution cost by examining number of objects distributed between various CDN levels and the corresponding data volume, the delivery cost given by bandwidth usage for lower level nodes, and storage cost given by cache space utilized by the CDN nodes. CDN performance also is evaluated through user perceived quality which we measure with request hit ratio.

### *13.7.1 iTVP CDN Configuration*

In the current configuration there are two replica servers operating in Poznan and in Krakow with a number of proxy/caches located in several cities: Warszawa, Poznan, Gdansk and Szczecin connected to Poznan region, Krakow, Lodz and Katowice connected to Krakow region. The origin server with iTVP content repository

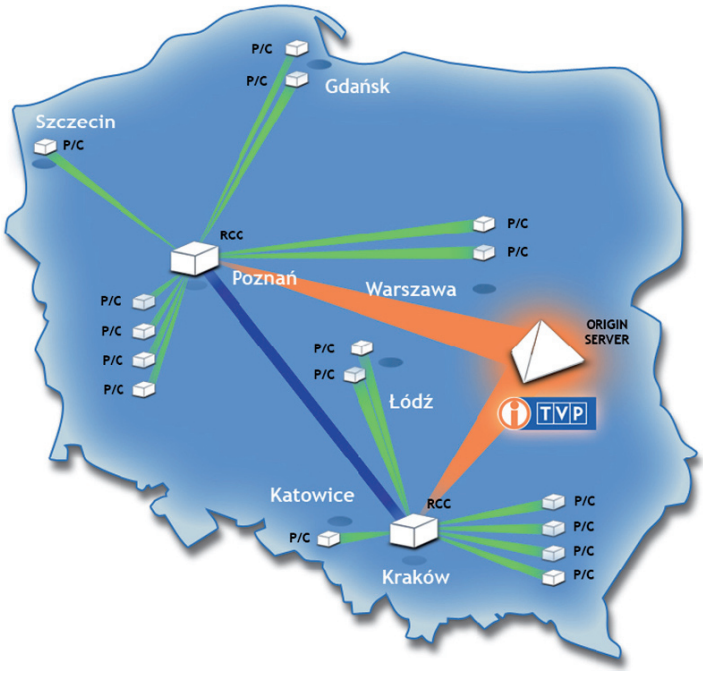



**Fig. 13.4** Current iTVP CDN configuration

is located in Warszawa and connected to both replica servers as shown in Fig. 13.4. Cache nodes and default proxy/caches are equipped with cache space of size on the order of TBs. Dedicated proxy/caches have typically 400 GBs of cache space.

### *13.7.2 Content Repository Characteristics*

Content is encoded with MPEG4-complaint codecs (Windows Media). Most objects are available in two different quality formats, one encoded for transmission speed ranging from 28 to 128 kbps destined mostly for handheld devices, and the second one encoded for speed ranging from 160 kbps to 700 kbps. Selected content is encoded for higher speeds with the target value of 1.5 Mbps. The content intended for downloads is also encoded with the quality corresponding to the speed of 1.5 Mbps. Each format contains multiple streams for various levels of available bandwidth within the range defined for a given format (multi-rate CBR). A lower quality format contains 3 to 4 video streams with rates ranging from 16 to 91 kbps, and 1 to 3 audio streams with rates ranging from 8 to 20 kbps. A higher quality content format usually contains 4 to 5 video streams with rates ranging from 121 to 563 kbps, and 3 to 4 audio streams with rates ranging from 20 to 128 kbps. Figure 13.5 presents

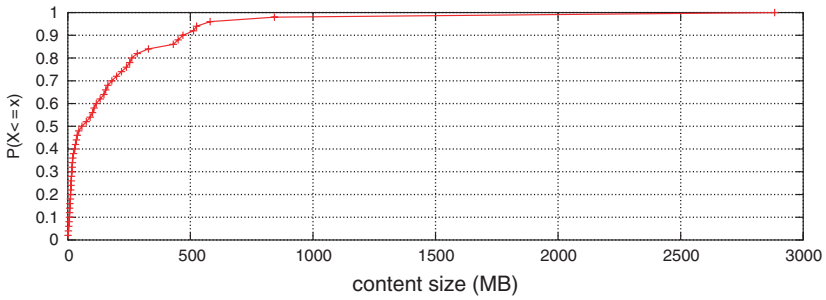



**Fig. 13.5** Content size CDF

content size Cumulative Distribution Fucntion (CDF) for on-demand content. Most files contain several hundred MBs of data. More precisely, 46% of objects have sizes larger than 100 MBs.

The number of objects published daily is typically on the order of several tens of items. There are several news programs that are broadcast several times a day and the archived versions of these programs are published for on-demand access soon after their live transmissions. Such content is usually accessed during the following couple of weeks only, hence, the publication rate of such programs does not affect the total number of content objects in the provider's repository that are accessed by the users.

### *13.7.3 CDN Load*

The load of the multimedia CDN may be characterized by different quantities, e.g. number of users, user sessions, total and average time of the content playout as well as volume of data sent to the end users. We define number of users as the number of unique IP addresses from which requests for content were received. This number should be interpreted as a lower bound on the actual number of users, however, we are working on more accurate measures to achieve better resolution of this parameter. We observe rapid growth of the number of users, exceeding hundreds of thousands per month, which is proportional to the improvement of services offered. The number of user sessions is also rapidly growing and is on the order of millions per month, which means that most users are the returning ones. The total time of the content playout during one month is on the order of hundreds of playout years, with the average playout time on the order of tens of minutes per session. Most iTVP content have considerable playout duration, which is currently rather rare in the Internet. The volume of data sent to the end users depends mainly on the content encoding quality and the available bandwidth. Since these parameters are steadily improving, we count hundreds of TBs of data sent to the end users monthly and this number is growing.

The load of the CDN is directly influenced by changes in the number of concurrent users who play the content. The number of concurrent users varies significantly over time showing diurnal and weekly patterns. There are several different patterns depending on the type of content. A different pattern can be observed between TV and radio content. The peak number of users for TV content is usually reached in the evening hours, while radio content is accessed mostly in the morning and early afternoon hours. The differences in access patterns also show that on a weekly scale, the radio content popularity is the highest on work days. For TV content, popularity distribution over time is different for weekends than for week days. On weekends there are more user requests in the early afternoon hours than during week days. Figure 13.6 presents the number of concurrent users relative to the maximum value over a period of one week for two types of content and the aggregate number for all contents. Access patterns also vary on longer time scale. Consequently, the resource requirements also exhibit variations on different time scales.

Figure 13.6 also shows different characteristics of user request frequency for radio and TV programs. The radio content is accessed and played rather continuously compared to the program-dependent access to the TV content. The difference in access pattern for each type of content is visible when we compare the user request distribution with the distribution of data volume transmitted to the end users over time. Figure 13.7 shows the cumulative volume of data transmitted, relative to the

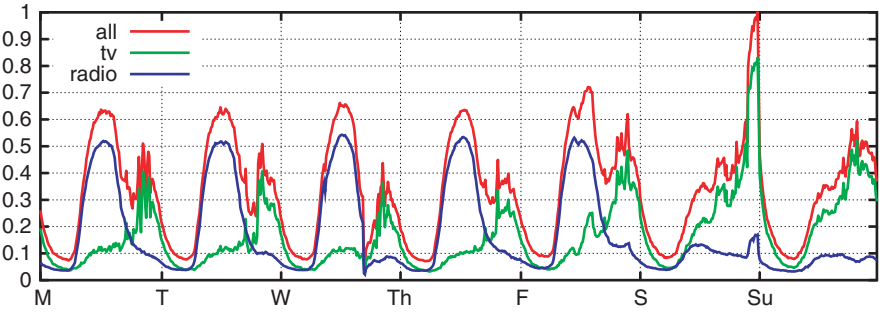



**Fig. 13.6** Number of concurrent users variability

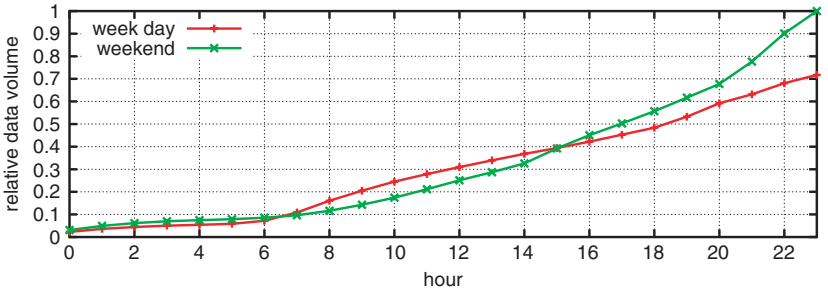



**Fig. 13.7** Cumulative volume of data transmitted over time

maximum value for a period of one day. We observe that in the morning hours the increase in cumulative data volume is much more gradual than later during the day, since initially most users receive radio content. In the evening hours on the other hand, most users receive TV content which requires more data to transmit. The change in the slope occurs earlier on weekends, since the number of users receiving TV content starts increasing earlier.

### *13.7.4 Content Distribution Performance*

CDN performance is evaluated based on two groups of quantities. The first group characterizes resource usage related costs and includes distribution bandwidth, delivery bandwidth, and storage usage. The second group characterizes user experienced QoS.

We evaluate CDN effectiveness at various levels of hierarchy by comparing data volume transferred to nodes at a given level with data volume transmitted by these nodes to the level below. Since data flow takes place not only between adjacent levels in the CDN hierarchy but also horizontally within the higher level, we first evaluate the advantage of replica server cooperation by computing the actual origin server load and the estimated origin server load without replica cooperation. Figure 13.8 presents the ratio of data volume obtained daily from the origin server to the total volume of data transferred to both replica servers either from the origin server or from another replica, i.e. volume of data that would have to be obtained from the origin server without replica cooperation. The average computed over a period of several months is equal to 0.55, showing that data volume transferred from the origin server and exchanged between replica servers are comparable. Occasionally, the ratio reaches values close to 1 indicating that majority of content were obtained from the origin server. Typically, these are the cases, when a small number of less popular objects are obtained from the origin server by one replica, and due to lack of user interest they are not transferred to another replica. Based on this fact, we estimate that without replica server cooperation, the origin server load would

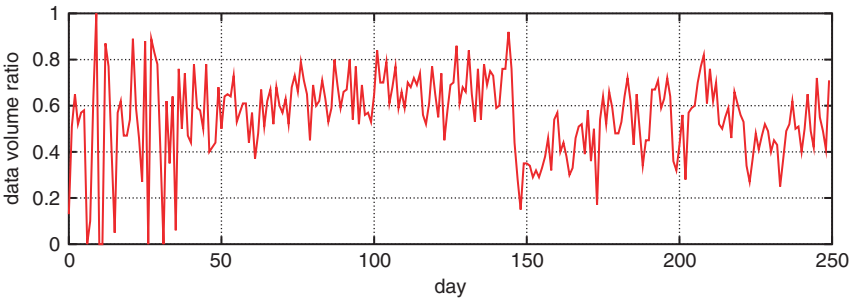



**Fig. 13.8** Server replica cooperation influence on the origin server load

be roughly doubled. This number justifies cooperation between replica servers. Furthermore, we conjecture that an increase in the number of replica servers should not affect the origin server load considerably provided that all replicas are connected in a full mesh manner and that load within each region is kept at a reasonable level. By reasonable level we mean that proxy/caches within each ISP's network are able to cache all content requested by its users over a period of several hours, i.e. the cache content is fairly stable over such a period of time. In a system with $n$ replica servers, the ratio of the origin server load with replica cooperation to the load without replica cooperation should be roughly 1 to $n-1$. The scalability is ensured at the cost of increased bandwidth usage between replica servers.

Next, we examine data volume transferred to and from replica servers (summed over all caches in a given replica) on a daily basis. Figure 13.9 presents ratio of data volume obtained daily by each replica server to the total volume of data transferred by a given replica server to all proxy/caches in its region. The average computed over a period of several months is equal to around 0.13 for Poznan replica server and around 0.12 for Krakow replica. Both values are on the same order, the difference can be attributed to the slightly higher load observed for Poznan replica. This in-out volume ratio allows us to estimate the origin server load reduction due to the existence of the higher level of the CDN hierarchy. If proxy/caches were to obtain content directly from the origin server, the load experienced by this server would be roughly 8 times higher. Hence, we can conclude that higher CDN level nodes are quite efficient in reducing the origin server load. The replica server load depends linearly on the number of proxy/caches within its region. Thus, an increase in the number of users in a given region, resulting in an increase in the number of proxy/caches, may prompt the decision to add another replica server and to split the existing region.

A similar comparison between the volume of data transferred to a node and the volume of data transmitted by a node is performed for proxy/caches at the lower level of the CDN hierarchy. However, in this case we have to take into account the fact that users often do not watch the entire content. Only part of the content can be transmitted to a user while the entire content is obtained and stored by the proxy/cache. Therefore, in addition to the data volume transferred by a proxy/cache

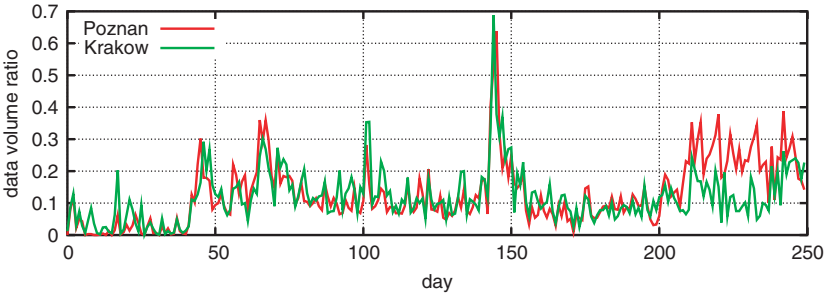



**Fig. 13.9** Data volume transferred *to* replica server vs data volume transferred *from* replica server

to its users we also compute a theoretical upper bound by summing up the sizes of all objects accessed by users regardless of what part is actually played by a user. The ratio of data volume obtained by a proxy/cache to the volume of data transmitted to users daily is around 0.04, meaning that proxy/cache transmits roughly 25 times more data than it obtains. The ratio of data volume obtained by a proxy/caches to the theoretical upper bound on the data volume transmitted is around ten times lower. Proxy/cache load depends linearly on the number of users, since content is delivered in unicast mode to users. We also examine volume of data transmitted by a proxy/cache in the context of data volume transmitted by the origin server to the CDN, in order to estimate the origin server load we would observe without a CDN. It appears that directing users of one proxy/cache only to the origin server would require the server to transmit as much as 50 times more data than it currently transmits to replica servers.

In order to evaluate the effectiveness of content caching we also compute the average number of acquisition per content at various levels of the CDN on various time scales. A high value of this factor could result from low resource, namely storage space, availability, or from ineffective storage space management, e.g. content replacement strategy. The number of acquisition from the origin server averaged over a period of several months is 1.02 per content, meaning that vast majority of objects are obtained from the origin server only once. Similar per content number of acquisitions is obtained for each replica server. For proxy/caches this factor has higher values, typically around 1.2, when computed over a several months long period of time. As we decrease the length of time period over which the number of acquisitions is averaged, the value of this factor quickly approaches 1 indicating that even if a content is acquired from replica server more than once, the transmissions are usually days or weeks apart. We conclude that the CDN is equipped with sufficient storage space for the load currently experienced by it, and that the available space is managed efficiently.

The reduction of the origin server load and the network bandwidth usage we have shown so far comes at the cost of increased storage space requirements. We evaluate the storage related costs of the CDN operation by comparing the total volume of data stored by a node with the total volume of data transmitted by that node over a given period of time. The storage usage is computed as the aggregate size of all objects stored by a node over a given period of time. For replica servers the ratio of stored and transmitted data volume is very similar to the ratio of obtained and transmitted data volume, since the number of acquisitions per content is only slightly larger than 1. Thus, we estimate that each replica server transfers roughly 8 times more data than it stores on a daily basis. The ratio of data volume stored by a proxy/cache to the volume of data transmitted to users daily is around 0.4. If users were to play the entire content every time they access this content, the volume of data transmitted by a proxy/cache could be as many as 8 times higher, resulting in much lower ratio of volume stored to volume transmitted. These numbers illustrate the gain from the increased amount of storage space available due to the existence of a number of CDN nodes organized in a two-level hierarchy.

### *13.7.5 User Perceived Quality*

User perceived QoS is characterized by the hit ratio, i.e. the number of user requests for which content is available from at least one of the CDN nodes that could provide services to the users. Hit ratio is directly related to the CDN performance in terms of content distribution. Quality of content playback has a great impact on user perceived quality; however, it depends mostly on the properties of the network path between proxy/cache and user. Playback quality is much more difficult to quantify in an objective way. Therefore, we concentrate on the content access time to evaluate the effectiveness of the CDN. The access time depends on whether the request is a hit or a miss. In the former cases the time needed to obtain the response depends more on the properties of network path between the manager node and the user than on the CDN efficiency in general and regional manager's load in particular. In the latter case that cut-through distribution and QuickStart mode make distribution time practically independent of the distribution path length and content size, respectively. Hence, we evaluate content access time based on the request classification as a hit or a miss. This hit ratio is defined at the user level and is typically higher than hit ratio computed for proxy/caches since user may access content from a default proxy/cache while the content distribution to a dedicated proxy/cache is started in response to the request. Typically, the hit ratio observed in the CDN is above 0.9. We conclude that the vast majority of users experience a very short access delay.

## 13.8 Future Research Directions

Despite the fact that iTVP CDN is fully operational and utilized on a large scale, there are several directions in which the future research is planned. Some of them are directly related to content delivery while others concentrate on the set of services provided to users and the consequences they will have for the iTVP CDN.

One challenge will be to monitor the system growth and expand the CDN as needed. The observed resource usage and user perceived quality will be used to verify decision making process to add new replica servers, choose their locations, and to split the existing regions. In other words, the challenge will be to monitor the system and expand it following the rules determined by the system architecture which was designed to ensure high scalability of the CDN.

Currently content distribution in iTVP CDN is performed mostly in pull mode. Cache content at each node depends directly on users' requests and changes in response to these requests. The push mode distribution mechanism is used typically prior to live transmissions or after publication of content with anticipated high popularity. We plan to investigate the effect of using push mode to complement the pull mode to a larger degree than it is done currently based on the content access patterns observed on a variety of time scales. We have shown that the CDN load changes with time of the day and day of the week. Consequently, resource usage also fluctuates over these time scales. We plan to utilize unused resources for content distribution

to improve user perceived quality, namely schedule distribution of content prior to its predicted access time during high resource availability.

iTVP CDN was designed to operate as a platform utilized by a number of independent content providers. We expect that in the future not only the number of content providers will increase but that an end user will be able to perform both content consumer and provider roles. Thus, there will be a need for content exchange between groups of users potentially on a large scale. These changes will determine the directions of future research. We plan to investigate whether implementing a hybrid solution, where CDN is combined with P2P, such as proposed by Dong et al. [9] and Xu et al. [21], would be possible and advantageous (e.g. locally within an ISP network). However, applying P2P-like distribution mechanism to multimedia streaming is challenging due to a number of problems; asymmetry of bandwidth availability for the end users being one of the major ones.

One of the main goals in building iTVP platform is to provide interactive access to its users. Interactive services are gradually introduced. Due to their nature, services such as time shifting of live transmissions or PVR, are provided to users on individual basis and may have significant influence on the CDN resource usage. Therefore, we plan to investigate CDN design issues with respect to the interactive access to the offered content. In addition to new services, we also plan to extend the set of end devices with which users can access iTVP service. More specifically we will include various types of handheld devices such as palmtops or cell phones. Devices of this type require not only adjusting the content format but also introduce a new dimension to content distribution, namely user mobility. Consequently, we will also consider portability that will allow users to switch from one end devices, for example, from a cell phone display to another higher quality device when available.

## 13.9 Visionary Thoughts for Practitioners

We believe that the multimedia CDN development will pave the way for network-aware services, which will be delivered seamlessly to the end users over broadband IP networks. Many network-demanding applications, to be provided on a large-scale to the consumers, must be distributed over the network and need careful planning of the underlying infrastructure to achieve the desired performance and quality provisioning. We foresee that the next step in the multimedia CDNs development will focus on the change from *content delivery* to advanced *service delivery* on a network level. Such a change will lead to the transparent provisioning of the application services as the main goal of the next generation CDNs. The application services will use the content assets as well as the end user input provided to the CDNs as elements to be processed in dynamic, highly distributed, multi-function workflows, and as a result will deliver the personalized and interactive multimedia experience. Such an approach will enable multi-channel delivery and integration of different multimedia environments and consequently enable next generation multimedia applications that will profile content accordingly to the end user terminal capabilities.

This development scenario is very challenging in many research areas, e.g. resource management, content distribution, workflow construction and scheduling, task and dependency management, service discovery, service monitoring, and network level integration. We expect the need for such environments to arise in the near future, driven by the market trends (e.g. HD-quality videos, networked interactive games or virtual presence/work environments) as well as the end users' expectations to have access to services anytime and anywhere.

## 13.10 Conclusions

In this chapter, we present the iTVP platform built for large scale delivery of live and on-demand video services over broadband IP networks. A key component of the iTVP platform is a CDN. Designing a CDN for multimedia content over IP requires consideration for the characteristics of the content itself and for its delivery mode, namely content streaming. We have described factors that affect a number of design decisions and presented the resulting CDN architecture along with the CDN node placements rules, content allocation, and user request routing mechanisms. Since iTVP CDN is a real system which has been deployed and functions in the real environment, we have a rare opportunity to verify our design decisions. We have analyzed the advantages and the costs of the two level hierarchical architecture. We find that iTVP CDN is quite effective in reducing the origin server load, in efficient use of network bandwidth and storage space, and in ensuring short access time to content for the end users. We have also examined the scalability issues since the iTVP is a system that is growing very dynamically. As the size of the content provider repository is expanded and the attractiveness of the programming offer increases, the platform gains new users. Hence, we keep monitoring the resource usage and the user experienced QoS. Data collected from the system enables a detailed analysis of the content access patterns that can provide insight into CDN functioning and allow for more efficient resource usage. We have presented several future research direction aimed at this goal.

## References

1. Aggarwal, C., Wolf, J.L., Yu, P.S.: On optimal batching policies for video-on-demand storage servers. In: IEEE Conference on Multimedia Systems, pp. 253–258 (1996)
2. Almeida, J.M., Eager, D.L., Ferris, M., Vernon, M.K.: Provisioning content distribution networks for streaming media. In: IEEE INFOCOM, Vol. 3, pp. 1746–1755 (2002)
3. Almeida, J.M., Eager, D.L., Vernon, M.K., Wright, S.J.: Minimizing delivery cost in scalable streaming content distribution systems. IEEE Transactions on Multimedia **6**(2), 356–365 (2004)
4. Binczewski, A., Meyer, N., Nabrzyski, J., Starzak, S., Stroinski, M., Weglarz, J.: First experiences with the Polish Optical Internet. Computer Networks **37**(6), 747–760 (2001)

5. Cahill, A.J., Sreenan, C.J.: An efficient cdn placement algorithm for the delivery of high-quality tv content. In: 12th annual ACM international conference on Multimedia, pp. 975–976 (2004)
6. Cahill, A.J., Sreenan, C.J.: An efficient resource management system for a streaming media distribution network. Interactive Technology and Smart Education **3**(1), 31–44 (2006)
7. Cranor, C.D., Green, M., Kalmanek, C., Shur, D., Sibal, S., der Merwe, J.E.V., Sreenan, C.J.: Enhanced streaming services in a content distribution network. IEEE Internet Computing **5**(4), 66–75 (2001)
8. Czyrnek, M., Kusmierek, E., Mazurek, C., Stroinski, M.: Large-scale multimedia content delivery over optical networks for interactive TV services. Future Generation Computer Systems **22**, 1018–1024 (2006)
9. Dong, Y., Kusmierek, E., Duan, Z., Du, D.H.: A hybrid client-assisted streaming architecture: Modelling and analysis. In: The 8th IASTED International Conference on Internet Multimedia Systems and Applications (2004)
10. Gao, L., Zhang, Z.L., Towsley, D.F.: Catching and selective catching: efficient latency reduction techniques for delivering continuous multimedia streams. In: ACM Multimedia, Vol. 1, pp. 203–206 (1999)
11. Griwodz, C.: Movie placement in a hierarchical CDN with stream merging mechanisms. In: N. Venkatasubramanian (ed.) SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), pp. 1–15. SPIE (2004)
12. Guo, L., Chen, S., Xiao, Z., Zhang, X.: Disc: Dynamic interleaved segment caching for interactive streaming. In: 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), pp. 763–772. IEEE Computer Society (2005)
13. Guo, Y., Sen, S., Towsley, D.: Prefix caching assisted periodic broadcast: Framework and techinques to support streaming for popular videos. In: IEEE International Conference on Communications, Vol. 4, pp. 2607–2612 (2002)
14. Hu, A.: Video-on-demand broadcasting protocols: A comprehensive study. In: IEEE INFOCOM, Vol. 1, pp. 508–517 (2001)
15. Hua, K.A., Cai, Y., Sheu, S.: Patching : A multicast technique for true video-on-demand services. In: ACM Multimedia, pp. 191–200 (1998)
16. Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: International World Wide Web Conference, pp. 252–262 (2002)
17. Kusmierek, E., Czyrnek, M., Mazurek, C., Stroinski, M.: iTVP: Large-scale content distribution for live and on-demand video services. In: R. Zimmermann, C. Griwodz (eds.) Multimedia Computing and Networking SPIE-IS&T Electronic Imaging, Vol. 6504. SPIE (2007). Article CID 6504-8
18. Kusmierek, E., Du, D.H.C.: Proxy-assisted periodic broadcast for video streaming with multiple servers. Multimedia Tools and Applications, Online First (2004)
19. Ramesh, S., Rhee, I., Guo, K.: Multicast with cache (Mcache): An adaptive zero delay video-on-demand service. In: IEEE INFOCOM, Vol. 1, pp. 85–94 (2001)
20. Su, Z., Katto, J., Yasuda, Y.: Dynamic replication of scalable streaming media over content delivery networks. In: Communication and Computer Networks (2004)
21. Xu, D., Kulkarni, S.S., Rosenberg, C., Chai, H.K.: Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution. Multimedia Systems **11**(4), 383–399 (2006)
22. Yang, M., Fei, Z.: A model for replica placement in content distribution networks for multimedia applications. In: IEEE International Conference on Communications, Vol. 1, pp. 557–561 (2003)

# Chapter 14
# Information Dissemination in Mobile CDNs

Nicholas Loulloudes, George Pallis, and Marios D. Dikaiakos

## 14.1 Introduction

With the recent development of technologies in wireless access and mobile devices, the mobile network has become a key component of today's Internet vision [1, 46]. Current mobile networks, which are being deployed worldwide, enable mobility features to new applications and also extend existing wired Web applications to mobile terminals. The mobile wireless network offers a rich assortment of dynamic and interactive services, such as GPS navigation information, mobile TV, vehicular traffic information, and location-oriented services. The provision of such services requires techniques to disseminate data as efficiently as possible in order to minimize the total network traffic and to improve the mean response time to mobile users.

In the wired Web, network performance can be substantially enhanced by using additional bandwidth, which is often available at low cost. However, this approach is impractical for mobile wireless network infrastructures. Most of these networks have fixed spectrum and achievable data rate is fundamentally limited by interference [46]. This problem is likely to get more serious when more mobile users start using bandwidth-intensive services such as streaming media. In this context, caching and prefetching might be a solution. Specifically, these approaches have been extensively used in the wired Web to optimize the amount of bandwidth consumption by shifting the traffic away from overloaded content providers and closer to the content customers [43]. Although these methods offer several benefits (i.e. conservation of network resources and reduced latency), the dissemination of dynamic content and resource-hungry applications (e.g. multimedia applications) remain a challenge.

Nicholas Loulloudes
Department of Computer Science, University of Cyprus, 75 Kallipoleos str. 1678, Nicosia, Cyprus, e-mail: loulloudes.n@cs.ucy.ac.cy

George Pallis
Department of Computer Science, University of Cyprus, 75 Kallipoleos str. 1678, Nicosia, Cyprus, e-mail: gpallis@cs.ucy.ac.cy

Marios D. Dikaiakos
Department of Computer Science, University of Cyprus, 75 Kallipoleos str. 1678, Nicosia, Cyprus, e-mail: mdd@cs.ucy.ac.cy

R. Buyya et al. (eds.), *Content Delivery Networks*,

© Springer-Verlag Berlin Heidelberg 2008

Content Delivery Networks (CDNs) promise to address these challenges by moving the content to the "edge" of the Internet, and thus closer to the end-user [45]. An introduction to CDNs can be found in the first chapters of this book. Although there has been much work on wired CDNs [27, 32, 40], content dissemination on mobile environments has received little attention so far [1, 49]. This is due to the limited Internet access capabilities of most mobile terminals in the recent past. However, this situation seems to be changing with the advent of innovative cellular (e.g. 3G) and wireless (e.g. WiFi) services which allow mobile terminals to access Internet and other data services at speeds comparable to traditional wired access [46]. Previous research [6, 48] shows that cooperative caching in mobile environment improves the network performance and information dissemination. In this context, we believe that the infrastructure of CDNs may provide a scalable and cost-effective mechanism for accelerating the information dissemination in the mobile wireless environment [45]. However, the mobile wireless network infrastructure represents a fundamentally different information medium from the traditional Web in terms of access devices used, content availability, bandwidth, and cost to the end-user. Thus, the typical CDNs cannot be enhanced by mobile wireless networks, since CDN architecture does not take the distinguished characteristics of these networks into account. In this context, we define *mobile CDNs as overlay networks of surrogate servers which deliver content in the mobile wireless network infrastructures*. Specifically, CDNs may offer an exciting playground for exploiting the emerging technological advances of mobile computing.

The purpose of this chapter is to present the challenges and the current status of mobile CDNs, discuss the recent evolution of the mobile wireless networking infrastructure, as well as to investigate how information dissemination can be improved by the emerging mobile CDN practices.

The rest of this chapter is structured as follows. Section 14.2 presents the need for mobile CDNs. Section 14.3 introduces the mobile CDNs. Section 14.4 presents the wireless network infrastructure of mobile CDNs. Section 14.5 presents our vision about how the existing intermediaries in mobile environments can be adopted in mobile CDNs. Section 14.6 provides some implementation and experimentation perspectives for mobile CDNs. Section14.7 presents the future research directions over these networks. Finally, Sect. 14.8 concludes the chapter.

## 14.2 Motivation

The mobile Internet, defined as wireless access to the digitized contents of the Internet via mobile devices, has significantly advanced in terms of user population. Recent studies have shown that in Japan the number of people using the mobile Internet already exceed those using the stationary Internet [4]. In general, the mobile Internet goes where the users go; users demand Web access when and where they need it, using their mobile devices.

Nowadays, an increasing number of content providers is investing in mobile Internet. The automotive industry has already introduced such mobile technologies in vehicles that provide accurate navigational and traffic aids (traffic conditions such as accidents, congestion, road constructions, diversions) to their drivers. Also, vehicles can be equipped with devices that alert their drivers for emergency situations (e.g. fire, earthquake damages, terrorist attack damages, etc) using multimedia data. Although in most occasions, a simple text message is sufficient, multimedia data, such as images and videos of an accident (or a dangerous situation further ahead), provide drivers with more precise and convincing information in order to take any necessary actions. Furthermore, the banking industry has identified business opportunities in mobile Internet including automated banking services. Nowadays, many mobile phone users readily access such services from their handsets. In addition, mobile Internet has opened new opportunities to the entertainment industry by selling online music, books, films, and games. For instance, travelers/commuters, who are waiting at terminals, can use their mobile devices (e.g. PSP) to play games or interact with their friends who are in other geographic locations.

Implementation of the above examples requires advances both in wireless network technologies and supporting devices, as well as the development of a scalable and resilient internetwork infrastructure that would support efficient information dissemination techniques to mobile users. From a technological viewpoint, the recent advances in wireless networking (e.g. 3G, GPRS, Dedicated Short Range Communications - DSRC[1]) guarantee that mobile devices can access Internet and other data services at speeds comparable to traditional wired access.

The infrastructure of CDNs [45] provides a scalable and cost-effective mechanism for accelerating information dissemination in the wired Web. However, the architecture of typical CDNs is inadequate to enhance the features of mobile Internet. since it does not take mobility into account. The variations in mobile user requests are caused not only by changes in content popularity but also by user mobility. Each user request is characterized by the requested content, the time of the request, and the location of the user. In order to support mobile users, surrogate servers should be located "close" to the base stations of the wireless network providers.

Furthermore, the CDNs architecture should be reconsidered in order to meet the new challenges of mobile users needs. One characteristic of mobile networks is the *scarcity of resources*; due to the small sizes of portable devices, there are implicit restrictions with respect to the availability of storage, computation capacity and energy. For instance, consider a father who shoots with a Wi-Fi camera a digital video of his three-year-old child while playing at the beach. His thought is to upload the video to a server in order to free up his camera memory and thus increase its capacity for more pictures/videos. So far, typical CDNs do not support the uploading of user content to the surrogate servers. In practice, the CDN distributor module is responsible to decide which content would be replicated by surrogate servers. In light of the above, surrogate servers in mobile CDNs should provide user-oriented

[1] Dedicated Short Range Communications:
http://www.leearmstrong.com/DSRC/DSRCHomeset.htm

services. This can be implemented by allocating a portion of their cache to be used by mobile users for direct content uploads.

Another parameter that mobile CDNs should take into account is the *navigational behavior of mobile users*. A significant factor which affects the users navigational behavior is the actual devices being employed. Mobile devices have limited input capabilities; for instance, the numeric keypads of mobile phones allow only "minimal" text entry compared to the keyboard entry on PCs. Moreover, mobile users must also contend with slow download times and incremental billing costs. Consequently, these characteristics have led to differences between the way users access information on the mobile Internet and the way they access information on the wired Web.

The mobility of users urges the development of state-of-the-art *geo-location oriented services* in mobile CDNs. Consider a mobile user who uses a CDN-supported application. While the mobile user is moving, the application should be replicated by another surrogate server so as to be always "close" to the mobile user. Moreover, geo-location services may also be used to detect mobile user Internet connection speed. This is crucial for Web site owners that would like to demonstrate multimedia applications (e.g. ads) to prospective customers.

In addition, typical CDNs do not provide any mechanism that monitors in real time the status of users (who interact with the CDN) and the underlying network infrastructure. However, such a *monitoring mechanism* can be considered as a key component in the support of content dissemination in mobile CDNs considering the inherent limitations of the underlying mobile wireless network infrastructure. These limitations are briefly explained below:

- *Frequent network disconnections*: The random or even organized mobility of users can severely influence their connectivity with the rest of the network. This is mainly due to: (1) small connection periods with base stations or other nearby mobile devices, (2) the presence of obstacles such as high buildings, trees or cars, which significantly degrade the wireless signal quality and (3) the possibility that users have temporarily gone out of radio coverage. The above factors can lead to bandwidth degradation or even total loss of connectivity, which can ultimately cause loss of information.
- *Network fragmentation(s)*: As a result of the frequent disconnections mentioned above, the mobile wireless network is vulnerable to fragmentation(s). These prevent end-to-end connectivity between mobile nodes and consequently minimize the availability of information.
- *Mobile nodes constraints*: The majority of mobile devices face significant constraints in terms of processing power, storage capacity and most importantly uptime duration. These limitations are imposed primarily from the fact that such devices run on battery power and secondary due to their small size which prevents increased processing and storage capacity.

Due to the above limitations of mobile wireless networks, it is crucial for the CDN to know the status of mobile users so as to minimize the overall traffic in the network. For instance, consider a mobile user who requests to download a podcast

to his/her MP3 player, but during the downloading process the device goes offline due to battery drain. In such a case, the CDN should be aware of the user status and block any further data transmission destined to him/her.

All the above issues conclude that disseminating information to mobile users in an efficient and cost-effective manner is a challenging problem, especially, under the increasing requirements emerging nowadays from a variety of applications (e.g. streaming media, dynamic content) and the inherent limitations of the mobile wireless environment. *Mobile CDN* infrastructure may meet these challenges. The next section presents our insight for mobile CDNs.

## 14.3 Mobile Content Delivery Networks

Contrary to wired CDNs, mobile CDNs are deployed within the range of a wireless network (e.g. cellular network, WiFi) and offer high quality services for delivering dynamic data and rich multimedia content to mobile devices. Specifically, the network infrastructure in mobile CDNs is de-composed in the two following components: (a) the wired network infrastructure and (b) the wireless network infrastructure. The former is an infrastructure responsible for the wired environment of the CDN; it provides the communication links which connect origin servers with surrogate servers and surrogate servers with network elements (e.g. switches, routers, 3G/GSM enabled base stations (BS), Wi-Fi enabled access points (AP)). On the other hand, the wireless network infrastructure is responsible for enabling communication and information dissemination among static and mobile users in the wireless environment of the mobile CDN. Therefore, the client-server communication is replaced by three communication flows: (1) between the client and the edge of the wireless network (AP or BS), (2) between the edge of the wireless network (BS or AP) and the surrogate server, and (3) between the surrogate server and the origin server. A typical mobile CDN is depicted in Fig. 14.1.

Considering that the surrogate servers should be placed "close" to BSs or APs of wireless networks, the network topology of mobile CDN should be redeployed so as to address this issue. Therefore, the placement of surrogate servers should be reconsidered so as to provide a sufficient coverage to the mobile wireless infrastructure. Due to the large amount of base stations, mobile CDNs should incorporate in their infrastructures more surrogate servers than a typical CDN.

Regarding the architecture of surrogate servers, their cache should be segmented into two parts in order to support user-oriented services to mobile users. The one part is devoted for replicating the content of origin servers (which have been contracted by a CDN provider), and the second part is dedicated to mobile users for content upload. Furthermore, surrogate servers of a mobile CDN should also provide geolocation oriented services to mobile users. The deployment of such services requires the definition of a service discovery protocol to select a surrogate server based on the location context of the mobile user (the location context refers to the current geographical area or position of a mobile user), in order to migrate applications to the surrogate server closest to the mobile user.

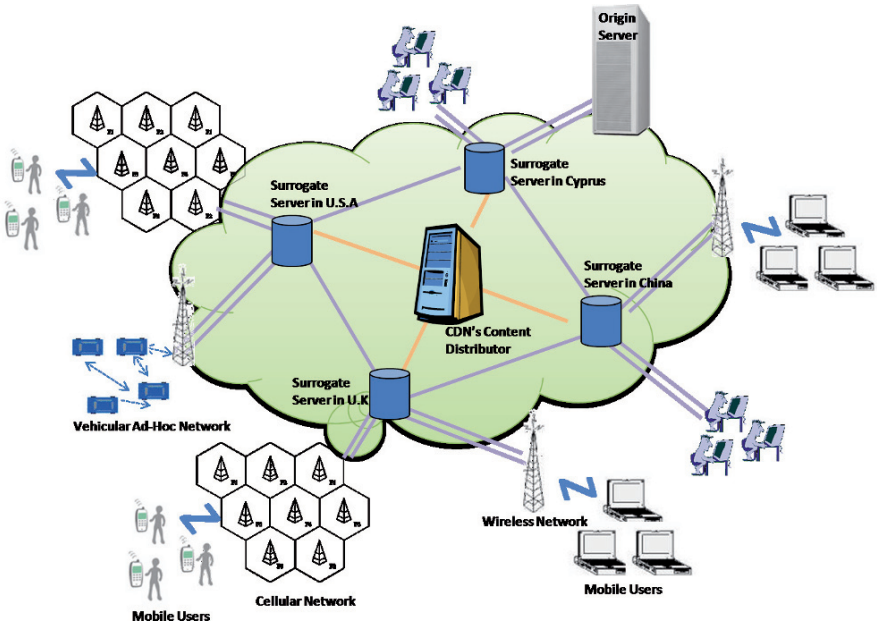



**Fig. 14.1** A typical Mobile Content Delivery Network

Apart from the networking issues involved in the establishment of the mobile CDN infrastructure, it is also important to determine which content outsourcing policy to follow (the content which would be replicated). Most CDN providers use either uncooperative or cooperative pull-based approaches [27]. The main characteristic of both approaches is that they are reactive; a data object is cached only when the user requests it and, consequently, these schemes impose large communication overhead (in terms of the number of messages exchanged) when the number of users is large. In addition, this mechanism does not offer high fidelity when the content changes rapidly or when the coherency requirements are stringent. Due to these limitations, the pull-based schemes are prohibitive in a mobile wireless environment. Mobile CDNs should enhance a cooperative push-based scheme. In contrast to the pull-based approaches (uncooperative and cooperative) which wait for the users to request information, the cooperative push-based approach lets the origin servers to proactively push the information into caches close to the mobile user, expecting a further reduction to the access latency. Indeed, Chen et al. [9] concluded that the cooperative push-based policy provides the best overall results, when compared to the other approaches. In such a scheme, the content is pushed (proactively) from the origin server to the surrogate servers. Upon a request, if the surrogate server has an object replica, it serves the request locally, otherwise, it forwards the request to the "closest" server that has the object replica and relays the response to the end-user. In case the requested object has not been replicated by some surrogate server, the request is served by the origin server. This scheme requires cooperation

among surrogate servers which incurs extra communication and management costs for its implementation. However, these costs are amortized by the fact that surrogate servers efficiently share the available bandwidth among them and also by the reduction of replication redundancy. In turn, the latter diminishes cache consistency maintenance costs.
Table 14.1 presents the main differences between a typical CDN and a mobile-specific CDN.

Therefore, the architecture of a mobile CDN, should consist of the following components:

- A set of surrogate servers (distributed around the world) which cache the origin servers content; the surrogate servers are not mobile and are located close to mobile base stations,
- A network infrastructure (wired and mobile wireless) which is responsible to deliver content requests to the optimal location and optimal surrogate server,
- A mechanism which monitors the network infrastructure in real-time for available bandwidth, latency and other sources of congestion,
- A cache manager, which manages efficiently the content that has been replicated to surrogate servers,
- A content location manager, which manages the content locations and schedules data prefetching,
- An accounting mechanism which provides logs and information to origin servers and CDN providers.

The above components interact with one another in order to deliver the requested content to mobile users. The content represents a collection of objects, which may change over time. For instance, content may represent a set of objects of a particular Web site, a streaming service, a Web service or any distributed application. When a mobile user requests an object, the user request is sent to the currently "closest"

**Table 14.1** Typical CDN vs. mobile CDN

| Features | Typical CDN | Mobile CDN |
|---|---|---|
| Content type | static; dynamic; streaming | static; dynamic; streaming |
| Users location | fixed | mobile |
| Surrogate servers location | fixed | fixed |
| Surrogate servers topology | "close" to Internet Service Providers | "close" to Base Stations |
| Replicas maintenance cost | medium | high |
| Services | application services | geo-location oriented application services; user-oriented services |
| Content outsourcing policy | cooperative/uncooperative pull-based scheme | cooperative push-based scheme |

surrogate server. If there is a cache hit (the surrogate server has an updated replica of the requested object), the request is served locally. Otherwise, the content location manager forwards the request to the closest surrogate server that has the requested object.

In response to the high variability in user demands and the dynamic nature of the requested content, mobile CDNs should integrate content management policies in surrogate server caches. There are different approaches related to which content to outsource [40], and which practice to use for outsourcing [19, 28, 44, 51]. The cache manager is responsible for the content which is replicated to surrogate servers as well as for keeping the surrogate server replicas up-to-date. Specifically, several issues related to cache consistency should also be considered [30]. Thus, the cache manager performs periodically checks in order to incorporate the spatial and temporal changes in user demand. On the other hand, the content location manager is responsible to replicate the outsourced content to surrogate servers. Finally, a mechanism monitors in real-time the user status since a user may be unavailable to receive the requested content due to its mobility.

Regarding the CDN market, although several CDN providers have been emerging in the market,[2] there is a lack of mobile CDNs. According to the authors knowledge, Ortiva Wireless[3] is the only mobile CDN provider, which is dedicated to delivering video to mobile users under highly variable wireless network conditions. By optimizing the delivery of mobile TV, video, audio, and Web content, Ortiva disseminates high quality information across any wireless network, expanding revenue opportunities, coverage, and network capacity without costly infrastructure modifications.

## 14.4 Wireless Network Infrastructures of Mobile CDNs

As we mentioned above, the network infrastructure in mobile CDNs is de-composed in the wired and the wireless network infrastructure. The wired infrastructure provides the resilience and fault-tolerance that CDNs require since a substantial part of it belongs to the Internet backbone which is provisioned with a high level of redundancy. In this section, we focus on the two variations of wireless infrastructure that are currently available [37] and discuss their suitability in mobile CDNs.

### *14.4.1 Mobile CDNs under Centralized Wireless Infrastructures*

Cellular and Wi-Fi networks are two indicative network types of centralized wireless infrastructures. From Fig. 14.2 it can be observed that in such infrastructures all

[2] A complete record of the existing CDNs' providers can be found in [29]

[3] Ortiva Wireless: http://www.ortivawireless.com/

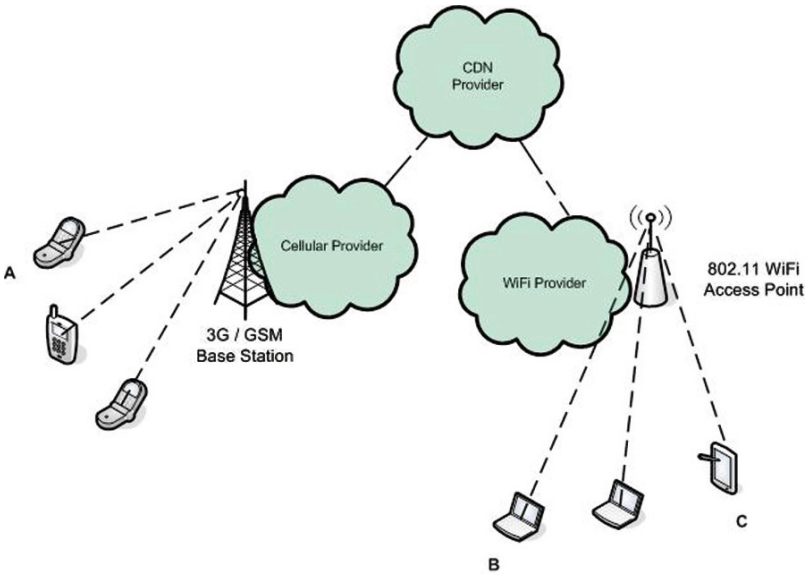



**Fig. 14.2** A centralized wireless network infrastructure

users communicate with a central authority. In the case of cellular networks this authority is usually a 3G and/or GSM enabled base station (BS), whereas in Wi-Fi networks this is an IEEE 802.11 enabled access point (AP). The BS/AP operates as a hardware bridge for all the wireless users [37] and is responsible for the allocation and control of the wireless channel. In addition, BSs and APs keep track of the changes in user mobility in order to prevent content dissemination to them when they are out of the radio coverage or offline.

The centralized wireless network infrastructure provides a good framework for mobile CDNs. For instance, consider user A in Fig. 14.2, who owns a cellular phone (registered with a cellular network) with an embedded video camera. This user is currently on vacation and makes a video recording of a sightseing s/he is visiting. At the same time, s/he wants to upload this video to a personal, public online blog, so that users B and C belonging to a Wi-Fi network have access to it. As long as the user is within the radio coverage of the cellular network, the phone transmits the recorded video to the responsible BS of the area. In turn, the BS forwards the video through the cellular network to the Web server hosting the blog. The mobile CDN provider is then responsible to push this content as close as possible to the WiFi network in which users B and C belong to.

However, this centralization in the wireless network can pose several problems. The existence of a central authority, which orchestrates the communication in the wireless environment, limits the scalability and resilience in mobile environments [10]. Therefore, careful and systematic provision of such infrastructures is necessary so as to provide the required communications coverage and Quality of Service (QoS) of mobile users.

### 14.4.2 Mobile CDNs under Ad-Hoc Wireless Infrastructures

Ad-hoc wireless infrastructures inherently provide several advantages over centralized ones for mobile environments. From a technical point of view, ad-hoc wireless networks are comprised from autonomous nodes that inter-communicate through wireless ad-hoc links. Communication takes place in a multi-hop fashion without relying on any sort of authority that overlooks channel allocation, topology formation or content dissemination. In contrast to centralized infrastructure networks, they provide cost-effective solutions in terms of network extension when the number of users increases [38, 50]. In summary, the main differences between centralized wireless infrastructures and ad-hoc wireless is depicted in Table 14.2.

A Mobile Ad-Hoc Network (MANET) is a wireless network with a dynamic arbitrary topology constructed in an ad-hoc, peer-to-peer (P2P) fashion from mobile nodes for information retrieval and dissemination. The basic characteristics of MANETs are summarized as follows:

- *Ad-hoc and Peer-to-Peer (P2P) connectivity*: such networks have the ability to be self-organized and self-configurable. The ad-hoc infrastructure is decentralized and independent of any fixed infrastructure. This inherent characteristic of decentralization highlights the fault tolerance of such networks, as no single point of failure exists. Furthermore, ad-hoc connectivity provides network scalability since there is no need of extra infrastructure setup. Their P2P connectivity emerges from the ad-hoc connectivity of the infrastructure and dictates that each participating node in the network is treated as an equal peer. This results from the fact that at any given time, a peer can function both as host and router [21].
- *Network topology*: dynamic due to the unpredictable random mobility patterns of participating nodes. These conditions are supplemented by the ability of mobile nodes to join and leave the wireless network at any given point in time without

**Table 14.2** Centralized wireless infrastructures vs. ad-hoc wireless infrastructures

| Features | Centralized Wireless Infrastructures | Ad-Hoc Wireless Infrastructures |
|---|---|---|
| Fault-Tolerance | No – Access Point is a single point of failure | Yes |
| Scalability when number of node increases | No | Yes |
| Self-Organization and Self-Configuration of Nodes | No | Yes |
| Maintenance and expansion costs | High; Extra Access Points are needed | Low |
| Central Control Authority | Access Point/Base Station | Does not exist |
| Bridging | Access Point or Base Station (hardware bridge) | Each node acts as a bridge for other nodes (software bridge) |

prior notification. Fixed nodes can peer with mobile nodes, providing gateway functionality to other fixed infrastructures or the Internet.

- *Mobile node constraints*: mobile nodes rely on rechargeable batteries for their operation meaning that uptime duration is substantially limited. In addition, their processing and storage capacities are constraint from their relative small size.

MANETs gained significant attention from the research and industry communities as prominent wireless ad-hoc infrastructures for information dissemination in fast-changing and uncertain mobile environments [15, 42]. For instance, consider the following example which is depicted in Fig. 14.3. User A wants to download a video from an online multimedia store (registered with a mobile CDN) to his/her laptop. However, in his/her existing location there is no wireless coverage from a WiFi or cellular provider. Thus, s/he switches to an ad-hoc mode and joins[4] an existing MANET comprised by users B,C,D and F. User E who is also part of the MANET provides Internet connectivity to the rest of the group. Hence, A's request will be forwarded through the MANET and E to the online multimedia store. The responsible mobile CDN will receive this request and will redirect it to a surrogate server which is "close" to E. Then, the aforementioned mobile CDN provides this video in a different encoding and compression so as to compensate for the inherent limitations of mobile nodes [2].

A subclass of MANETs are Vehicular Ad-hoc Networks (VANETs). These are formed among vehicles driven within road constraints and fixed-road side nodes that

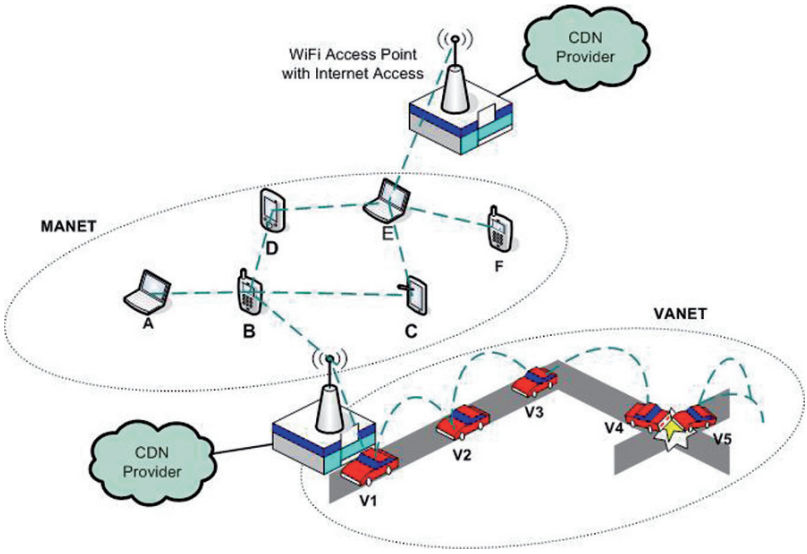



**Fig. 14.3** A typical ad-hoc wireless network infrastructure

[4] Security and how users trust each other are very crucial issues in wireless ad-hoc infrastructures, but these are out of scope in this chapter.

provide location services and Internet gateway functionality. Specifically VANETs were propelled by the need of implementing a suitable infrastructure for Intelligent Transportation Systems [2]. In general, VANETs are different from MANETs in terms of:

- *Network Topology*: even more dynamic than MANETs, due to the extremely high mobility of vehicles. As opposed to the latter, the movement of vehicular nodes in VANETs is quite predictable since most of the time vehicles are driven in a well organized fashion within the constraints of roads. As in MANETs, road-side nodes peer with vehicular nodes and act as gateway to other fixed infrastructures and the Internet. Also these road-side nodes broadcast location-oriented information to vehicles.
- *Node Constraints*: no constrains exist in terms of processing power and uptime duration since vehicular nodes are constantly fed with power from the vehicle's electrical installation.

The following scenario depicts the benefits if mobile CDNs are utilized in VANETs. Consider that vehicles V1...V5 have informally agreed to take part in a VANET. Suppose that a traffic accident occurs between V4 and V5 at the junction. Instantly, V4 and V5 broadcast an emergency alert and possibly some cabin photos or short videos to all their neighbor vehicles informing them about the accident. This alert is marked with high priority and is propagated throughout the VANET. As soon as V1 receives the alert, it utilizes its Internet connection from the nearby base station to post this alert to an online traffic monitoring system. The mobile CDN will place this alert and any accompanied photos or videos to the most appropriate surrogate server in terms of the accident location. This enables emergency response teams to receive an overall view of how the accidents looks like before going on site with the minimum latency. Furthermore, far-away vehicles heading towards the accident location will be informed in advance through the aforementioned traffic monitoring system such that necessary adjustments in their course are made so as to avoid congestion build-up. In general, mobile CDNs can be utilized in vehicular environments for:

- the provision of mobile Internet to vehicles with extensive support in dynamic and streaming content.
- dissemination of road traffic-related information to interested vehicles and authorities such as congestion, accidents and diversions.
- location-oriented services such as location and availability of various facilities in an area of interest (i.e. hotel, restaurants, gas stations, parking places).
- dissemination of environmental measurements gathered from vehicular sensors.
- support for distributed gaming.

Finally, Table 14.3 represents the key differences between MANETs and VANETs.

**Table 14.3** Mobile ad-hoc networks vs. vehicular ad-hoc Networks

| Features | MANET | VANET |
|---|---|---|
| Mobile Nodes | Laptops, Smartphones, PDAs | Vehicles |
| Node Movement | Random – Unpredictable | Organized – Predictable (within road constraints) |
| Node Constraints | Limited uptime due to battery constraints | Power is not an issue |
| Mobility | High | Very High |
| Network Topology | Dynamic | Dynamic |
| Supported Network Technologies | UMTS, GSM, WiFi, Bluetooth | DSRC, UMTS, GSM, WiFi |

## 14.5 Visionary Thoughts for Practitioners

Several projects from the research and industry communities exist in the literature which address the development and deployment of intermediary components between origin servers and mobiles nodes for seamless information dissemination [12]. Such intermediaries are deployed in wireless infrastructures optimizing intermittent network connections and bandwidth utilization. This section presents the existing work in the utilization of intermediaries in mobile environments and our vision of how these can be adopted in mobile CDNs.

*IBM's Web Express* system follows the client/intercept/server wireless computational model [18] for the optimization of Web access from mobile resource-limited nodes. This model aims to minimize latency over wireless links by utilizing an intercept technique that allows applications to optimize communications. In order to address this issue, WebExpress uses two key components: the Client Side Intercept (CSI) deployed on mobile nodes and the Server Side Intercept (SSI) deployed on the wired network. The CSI intercepts the mobile node request and co-operate with SSI so as to reduce data transmissions and improve content availability over the wireless link. Several optimization methods are used by WebExpress to meet these challenges such as caching of content both at the client (CSI) and server (SSI) side, computation of difference between cached based objects and new responses, multiplex of all requests over a single session, and HTTP header reduction.

In our view, WebExpress and more specifically the client/intercept model can be adopted in mobile CDNs by moving CSI from origin servers to surrogate servers. This relocation has the added benefits of further *minimizing the communication latency* during mobile users Web access, and *reducing the network traffic* over wireless links due to the optimization methods mentioned above.

Dikaiakos et al. [13] propose an independent information transfer protocol for the dissemination of information among vehicles and fixed road-side nodes participating in a VANET. The *Vehicular Information Transfer Protocol (VITP)* is an application layer, stateless protocol, that specifies the syntax and semantics of messages exchanged between vehicular peers, while at the same time it pertains its independency from the underlying VANET protocols. The authors proposed the deployment

of VITP on vehicles for the support of location-based, traffic-oriented, services to drivers. Queries in VITP are location-oriented. This means that queries are transmitted to a specific geographic area of interest in order to retrieve some desired content. Inside this destination (target) location, a dynamic collection of VITP peers, called Virtual Ad-Hoc Server (VAHS), is established on the fly in order to compute the reply to the location query. The VITP peers that participate in the VAHS co-operate to generate the final reply to the VITP request until a return condition is met. VITP queries follow a request-response methodology. In addition, the protocol includes support for data caching through cache-control headers which can be included in VITP queries. If there is a hit in a peer's cache for a VITP query, this peer can use the aforementioned headers to decide whether to respond based on its local cache or whether to retransmit the query towards to the target location.

In the context of mobile CDNs, fixed, road-side VITP peers can be used as surrogate servers. In contrast to vehicles, such peers have increased processing and storage capacity, higher wireless-radio coverage and continuous high-bandwidth Internet connection. Each road-side peer can be appointed as the responsible surrogate server for the area it covers. Therefore, these surrogate servers could cache the responses to frequently asked VITP messages and location information.

Furthermore, mobile CDNs may offer their services to several existing vehicular applications, such as CarNet [24] and TrafficView [26]. Specifically, CarNet was one of the first proposed applications that utilized efficient information dissemination techniques to provide vehicular services. On the other hand, TrafficView is a proposed framework, for inter-vehicle information exchange with regard to traffic and road conditions, route scheduling and emergency messages dissemination. TrafficView is able to display a real-time, dynamic view of the road traffic further down and overall act as a compliment of traditional GPS systems installed on many cars today.

Finally, Daly and Haahr [11] support that information dissemination in MANETs is improved by using intermediaries running the SimBet Routing algorithm. Specifically, SimBet uses a metric based on social networks analysis to locally determine the centrality of a mobile node within the network. For finding the central node, the notion of *betweenness centrality* is used. Therefore, if the destination node is not known to the sending node, then information is routed to the most central node. As far as mobile CDN is concerned, this scheme may be applied on such an infrastructure so as to forward the requested content to the most central mobile node of the wireless network; this is useful in case the user requesting the content is temporarily out of coverage (or offline).

## 14.6 Implementation and Experimentation Perspectives

The first step prior the actual design and implementation of new technologies, policies and applications to real-time, production, CDN infrastructures, are considered to be simulations. Simulations are usually employed during the initial development

phase of any of the above since it is quite difficult to experiment in real-time environments. From an implementation perspective, the following simulation methodologies may be used in order to evaluate the performance of a mobile CDN infrastructure:

- **Simulation testbeds:** Large scale simulation testbeds provide the necessary infrastructure for the implementation and evaluation of new CDN technologies, policies and applications. One such testbed is PlanetLab.[5] PlanetLab is a global research overlay network in which participating nodes are strategically located at sites around the world. PlanetLab forms a CDN testbed for creating and deploying planetary-scale services and massive applications that span a significant part of the globe. This CDN testbed consists of a network of high-performance proxy servers. Such proxy (*surrogate*) servers have been deployed on many PlanetLab nodes. Two academic CDNs have already been built on top of PlanetLab (CoDeeN[6] and Coral[7]). A detailed discussion of academic CDNs is given in Chap. 1.
- **Simulation software:** As a rule of thumb, in all scientific research areas, software simulators are a must-have tool during the design and testing phase of any product. CDN simulators pose no exception to the above rule and are highly valued among the CDN-oriented research and industry communities. Specifically, these tools provide the means for simulating CDN applications and infrastructures without the financial burden of acquiring, installing and configuring the actual underlying hardware. Simulation results are reproducible and easier to analyze since simulated environments are free from any other unpredictable and uncontrollable interfering factors (i.e. unwanted external traffic), which researchers may encounter while experimenting on real infrastructures. CDN simulators, simulate the behavior of a dedicate set of machines to reliable and efficiently distribute content to users on behalf of an origin server. Chapter 5 of this book presents an analytic simulation tool for CDNs called *CDNsim*.

However, the above CDN simulation software and testbeds are proposed having in mind fixed-wired infrastructures. None of the above frameworks inherently supports the simulation and evaluation of mobile CDNs due to the fact that user mobility, the wireless medium and other distinguished characteristics of mobile CDNs described in Sect. 14.3, are not taken under consideration. As per the view of the authors, in terms of CDN simulation software, practitioners should extend existing tools (such as CDNsim) through the development of new extensible add-on modules that will allow the support of mobile CDNs. Such modules could provide:

- **Realistic Mobility Traces:** Mobile nodes movement behavior can be described using a large set of mobility traces generated from well-known and accepted mobility models. For instance, a module that supports generation of mobility traces

[5] Planetlab: http://www.planet-lab.org

[6] A CDN for PlanetLab: http://codeen.cs.princeton.edu

[7] Coral CDN: http://www.coralcdn.org

could incorporate the functionality provided by SUMO [20]. SUMO is a well-known microscopic, highly portable traffic simulator which produces realistic mobility traces for simulating vehicular behavior in urban environments. Moreover, developers of such module could also consider CosMos, a communication and mobility scenario generator proposed by Günes and Siekermann [16]. Unlike other mobility generators that provide traces based on a single mobility model, CosMos integrates several mobility models thus providing more realistic mobility patterns for wireless network simulations.

- **Support for wireless environments:** As discussed previously in this chapter, the wireless environment exhibits different characteristics than the wired environment. For this reason, simulating the characteristics of the wireless environment is a crucial milestone towards the correct simulation and evaluation of mobile CDNs. New modules that provide support for wireless environments should be able to correctly and accurately simulate characteristics such as oscillating signal strength, bandwidth, intermittent, connections in both infrastructure and ad-hoc environments. Even more, such models can leap a step forward by possibly simulating phenomena associated with wave propagation such as multi-path, fading, diffraction etc.
- **Support for mobile resource-limited nodes:** Mobile CDN simulators should also take into account a key characteristic of mobile nodes: the limitation of available resources. More specifically, the majority of such nodes that make up a mobile CDN are limited in terms of energy availability. Hence, models that describe the power consumption behaviour of such nodes (changes between sleep, transmit, receive and idle modes) under different scenarios should be designed and implemented in new add-on modules. Ultimately, such modules will aid in the correct simulation of mobile nodes behaviour under the above limitations and will allow the experimentation on techniques that aim in energy conservation.

In terms of simulation testbeds, ORBIT[8] is a wireless network testbed that supports research in wireless technologies. It consists of large-scale wireless networks made up from 400 high-speed cellular (3G) and IEEE 802.11 enabled nodes interconnected in a grid layout. The obvious advantage of ORBIT over other large-scale testbeds is the support of mobility through well known mobility models such as the Brownian motion or the random waypoint model. This provides the ability to examine various wireless applications and technologies such as MANETs and location-oriented services.

From an experimentation perspective, we expect that mobile CDNs would improve the mean response time of mobile users as well as increase the byte hit ratio in the cache of surrogate servers. On one hand, the mean response time represents the users waiting time in order to serve their requests. On the other hand, the byte hit ratio provides an indication for the performance of the network. Moreover, we expect that surrogate servers in mobile CDNs would have low replica redundancy due to their high degree of cooperation. This is a critical issue if we take

[8] Open-Access Research Testbed for Next-Generation Wireless Networks(ORBIT): http://www.orbit-lab.org

under consideration that high data redundancy leads to waste of money for CDN providers [27]. Finally, it is expected that mobile CDNs can improve the QoS of mobile users by serving the majority of their requests. Without such an infrastructure, many mobile users might face denial of services.

## 14.7 Future Research Directions

Mobile wireless networks are characterized by the high variability in user demand as well as the high demand for dynamic content and media applications. For example, a driver wants to be alerted for any emergency situations during his/her journey using multimedia data. Another characteristic of mobile networks is the scarcity of resources. For instance, consider a person who wishes to upload a video to a server in order to free up his/her camera so as to accept more pictures/videos. Therefore, crucial issues should be addressed. In the following subsections, we discuss the future research directions for mobile CDNs in terms of content placement techniques, disseminating dynamic content and mobile streaming media.

### *14.7.1 Content Placement Techniques*

The content placement problem is to decide where content is to be replicated so that some objective function is optimized based on requests and resource constraints. This problem has been extensively studied for static user demands [19, 28, 44, 51]. However, due to mobility and resource constraints, the existing schemes are not applicable to a mobile wireless environment. Specifically, new approaches should be investigated, which deal with the high demand for dynamic content, media applications and user mobility.

In this context, the content placement approaches for dynamic user demands are of interest [8, 25, 31, 33]. An algorithm that dynamically places replicas and organizes them into an application-level multicast tree with limited knowledge of the network topology was presented by Chen et al. [8]. This algorithm aims at satisfying both user-perceived latency and server capacity. In [25] the authors presented another framework for dynamic content placement. In this approach, the problem of optimal dynamic content placement has been described as a semi-Markov decision process, where the user requests are assumed to follow a Markovian model. Presti et al. [31] address the dynamic content replication by using a non-linear integer programming formulation. Specifically, the decision on how the system should evolve is the outcome of a nonlinear integer programming formulation of the problem. Rabinovich et al. [33], presented an application CDN (called ACDN) which is dedicated to deliver dynamic content. They proposed a heuristic algorithm which dynamically places replicas based on past observed demand.

However, none of the above content-placement techniques has been evaluated under a wireless network infrastructure. Motivated by this fact, the authors in [1] presented an online heuristic algorithm for dynamic placement of content replicas in a mobile CDN. The proposed algorithm (called online MDCDN) is based on a statistical forecasting method, called Double Exponential Smoothing (DES). Taking user demand variations into account, this method predicts the future demand at each surrogate server. These predictions are used to decide whether to add a new content replica or remove an existing one in order to minimize the total traffic over the backbone. For the experiments, the authors used a mobility simulator [22] and modeled a 20-km radial city, divided into area zones based on population density and natural limits (e.g. rivers, highways, etc.).

A distributed algorithm, called DbC, was proposed by Miranda et al. [23] in order to place the content as evenly as possible among all the servers that belong to a wireless ad-hoc network. Thus, the replicas of the data items are sufficiently distant from each other to prevent excessive redundancy. On the other hand, the replicas remain close enough to each end-user to improve information dissemination. Simulation results showed that DbC improves the dissemination of items throughout the network. Three efficient content placement techniques for a wireless ad hoc network infrastructure have also been proposed by [17]. These methods place the replicas on mobile hosts taking into account either the PT values[9] of objects (E-SAF - Extended Static Access Frequency) or the PT values of objects and the neighboring mobile hosts (E-DAFN - Extended Dynamic Access Frequency and Neighborhood), or the PT values of objects and the whole network topology (E-DCG - Extended Dynamic Connectivity based Grouping). Experiments performed in [17] have shown that E-DCG gives the highest accessibility and the E-SAF method gives the lowest traffic.

### *14.7.2 Disseminating Dynamic Content*

The applications that disseminate dynamic content have high sensitivity to delays. For instance, a driver wants to know which road is better to follow in order to avoid any delays due to a traffic jam. In such a case, a delay - even a few seconds - may be intolerable. Thus, the efficient dissemination of dynamic content in mobile environments is a critical issue. To this end, a wide range of proposed approaches and techniques have been proposed under the CDN infrastructure in order to accelerate the generation and dissemination of dynamic content to mobile users [1, 5, 34]. Many of the proposed approaches are implemented in commercial systems (Websphere

[9] PT value is defined as the product of the popularity of the object and the time remaining until the object is updated next.

Edge services of IBM[10], EdgeSuite network of Akamai[11]) proving in this way the importance and applicability of dynamic content technology.

Fragment-based policies have received considerable attention from the research community in recent years [5, 34], since experiments have shown that the typical overhead of composing a Web page from fragments is minor to the overhead of constructing the whole page from scratch [5]. Akamai has also enhanced fragment-based policies using the Edge Side Includes technology [14]. A novel scheme to automatically detect and flag "interesting" fragments in dynamically generated Web pages that are cost-effective cache units, is proposed by Ramaswamy et al. [34]. A fragment is considered to be interesting if it is shared among multiple pages or if it has distinct lifetime or personalization characteristics.

Instead of caching fragments of Web pages, another common approach for disseminating dynamic content is to cache the means which are used in order to generate the pages over the surrogate servers. This approach is based on the fact that generation of content on demand needs more time than simply fetching any other dynamic content, since it requires to issue one or more queries to a database. Thus, a simple idea is to cache the application code at the CDN surrogate servers, and keep the data centralized. This technique is the basic module of the *Edge Computing* product from Akamai and ACDN [33]. A disadvantage of this approach is that all users requests should be served by a central database. This centralized architecture leads to performance bottlenecks. In order to address this issue, another approach is to create a partial copy of the database in each surrogate server. The systems that have been implemented this approach are known as Content-Aware Caching systems (CAC). Finally, another approach is to cache the result of database queries as they are issued by the application code. This approach is known as Content-Blind query Caching (CBC) [35]. When a query is issued, it is checked whether the query result is cached or not. Experiments results presented by Sivasubramanian et al. [41], have shown that the CBC presents the best performance when the query workload exhibits high locality. On the other hand, the Edge Computing and the CAC present better performance than the CBC when the query workload exhibits low locality.

### *14.7.3 Disseminating Mobile Streaming Media*

The increased bandwidth of next-generation mobile systems makes streaming media a critical component of future content delivery services. Thus, the efficient dissemination of streaming media in mobile environments is a challenging issue. Typically, streaming media content can be categorized as follows:

- *Live content:* the content is disseminated "instantly" from the encoder to the media server and then onto the end-user.

[10] IBM WebSphere Application Server:
http://www-306.ibm.com/software/webservers/appserv/was/

[11] Akamai: http://www.akamai.com/

- *On-demand content:* the content is encoded and then stored as streaming media files on media servers. The content is then available for request by the end-users.

The media server is a specialized one which consists of a software that runs on a general-purpose server. Its task is to serve the digitized and encoded content to end-users. When a user requests a certain content, the media server responds to the query with the specific video or audio stream. Design requirements of such servers under the context of CDNs are discussed by Roy et al. [36]. However, the dissemination of media content to a large number of mobile users creates serious problems due to the strict requirements of streaming media, the inherent limitations of wireless networks and the mobility of users.

In this context, a mobile CDN may address these limitations by distributing the high demands of mobile users for streaming media to its surrogate servers. Furthermore, CDN surrogate servers improve the dissemination of streaming content by employing state-of-the-art compression techniques (caching, encoding). Specifically, a mobile streaming media CDN should consider the following issues:

- Surrogate server selection: When a user requests an object, his/her request should be directed to a surrogate server for serving the requested content. To achieve this, Yoshimura et al. [49], proposed a mobile streaming media CDN architecture in which the surrogate server selection is determined by a SMIL[12] file modification. The SMIL file is stored in the streaming media server and contains information about the status of surrogate servers. In this architecture, the mobile users read the SMIL file in order to select the best surrogate server in CDN.
- Media Caching: Earlier research efforts have shown that CDN performance is improved when caching techniques are integrated in a CDN. The problem is to determine what media streams should be cached in surrogate servers disks. A solution would be to store all the media streams. However, such a solution is not feasible since media streams require huge amounts of storage. Therefore, efficient data management schemes should be considered in the context of mobile streaming CDNs. In such a scheme, the cache manager manages efficiently the content that has been replicated to each surrogate server. A simple idea is to partition the objects into segments. In particular, the media objects are divided into smaller units so that only partial units are cached. This leads to efficient storage and network utilization. In the literature, several variants have been proposed for caching segmentations [7]. Prefix caching [39], and variable sized segmentation [47] are some indicative techniques for caching segmentation.
- Managing Session Handoffs: In streaming media, the user sessions are usually long-lived. However, the long-lived nature of streaming sessions in a mobility environment has raised the issue of managing in an efficient way the session handoffs among surrogate servers. During a handoff, no frames should be lost, and the data stream to the video player should be kept as smooth as possible. In [49], a SMIL file is used to control the session handoffs.

[12] The SMIL (Synchronized Multimedia Integration Language) is a W3C Recommended XML markup language for describing multimedia presentations. It defines markup for timing, layout, animations, visual transitions, and media embedding among other things.

A mobile streaming media CDN (MSM-CDN) architecture, which enables media delivery over next-generation mobile networks has been described by Apostolopoulos et al. [3]. The MSM-CDN architecture has been designed to be modular and interoperable with other systems and underlying networks. In this scheme, the overlay servers are the basic components of the MSM-CDN; their task is to cache media streams. Here, an overlay server can be considered as surrogate server. The delivery of media is done through streaming and data-transfer interfaces. Specifically, the interfaces in MSM-CDN facilitate the delivery of media streams to mobile users. Another mobile streaming media CDN architecture has been proposed in [49]. The originality of this architecture is that all the technologies related to CDN are enabled by SMIL modification.

## 14.8 Conclusion

The recent advances in mobile content networking (e.g. GSM/3G, WiFi, etc.) enable the wireless network infrastructures to support bandwidth-intensive services such as streaming media, mobile TV etc. The information which is usually requested by mobile users is either dynamic content or media applications. Taking into account that mobile user appetites for information is expected to keep growing, we need innovative techniques that can improve information dissemination. However, the emergence of typical CDNs cannot meet these challenges in a mobile wireless network due to the inherent limitations of such an infrastructure as well as the distinguished characteristics of mobile devices. Traditional CDNs do not take the mobility of users into account. In this context, *mobile CDNs* may address these issues by accelerating the information dissemination in mobile wireless network infrastructures. A mobile CDN differentiates from typical CDNs in terms of the topology of surrogate servers, content outsourcing policy, and application services.

In this chapter, we presented a pathway for mobile CDNs, in order to understand their role in the recent evolution of the mobile networking infrastructure, as well as to investigate how the information dissemination can be improved by the emerging mobile CDN practices. In this context, the main characteristics of mobile CDNs were given. Next, we presented the most popular methodologies and implementations of mobile CDNs in terms of disseminating dynamic content, content placement techniques and mobile streaming media. Finally, we presented the network infrastructures of mobile CDNs and explored existing information dissemination techniques.

In summary, information dissemination in a mobile wireless environment is an interesting, useful, and challenging problem. The emergence of mobile CDNs has opened new perspectives in the research community. Although several research works exist, there is a lot of room for improvement in both theoretical and practical applications, since technology in the areas of mobile computing and mobile networking is still evolving.

## References


1. Aioffi, W.M., Mateus, G.R., Almeida, J.M., Loureiro, A.A.F.: Dynamic content distribution for mobile enterprise networks. IEEE Journal on Selected Areas on Communication **23**(10) (2005)
2. Anda, J., LeBrum, J., Ghosal, D., Chuah, C.N., Zhang, M.: Vgrid: Vehicular ad-hoc networking and computing grid for intelligent traffic control. In: Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st, Vol. 5, pp. 2905–2909 (2005)
3. Apostolopoulos, J.G., Wee, S., tian Tan, W.: Performance of a multiple description streaming media content delivery network. In: Proceedings of the 2002 International Conference on Image Processing (ICIP 2002), pp. 189–192. Rochester, New York, USA (2002)
4. Chae, M., Kim, J.: What's so different about the mobile internet? Commun. ACM **46**(12), 240–247 (2003)
5. Challenger, J., Dantzig, P., Iyengar, A., Witting, K.: A fragment-based approach for efficiently creating dynamic web content. ACM Trans. Inter. Tech. **5**(2), 359–389 (2005)
6. Chand, N., Joshi, R.C., Misra, M.: Cooperative caching in mobile ad hoc networks based on data utility. Mobile Information Systems **3**(1), 19–37 (2007)
7. Chen, S., Wang, H., Zhang, X., Shen, B., Wee, S.: Segment-based proxy caching for internet streaming media delivery. IEEE MultiMedia **12**(3), 59–67 (2005)
8. Chen, Y., Katz, R.H., Kubiatowicz, J.: Dynamic replica placement for scalable content delivery. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS), pp. 306–318. Cambridge, USA (2002)
9. Chen, Y., Qiu, L., Chen, W., Nguyen, L., Katz, R.: Efficient and adaptive web replication using content clustering. IEEE Journal on Selected Areas in Communications **21**(6), 979–994 (2003)
10. Chisalita, I., Shahmehri, N.: A peer-to-peer approach to vehicular communication for the support of traffic safety applications. In: Proceedings of IEEE 5th International Conference on Intelligent Transportation Systems, pp. 336–341 (2002)
11. Daly, E.M., Haahr, M.: Social network analysis for routing in disconnected delay-tolerant manets. In: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, pp. 32–40 (2007)
12. Dikaiakos, M.D.: Intermediary infrastructures for the world wide web. Computer Networks **45**(4), 421–447 (2004)
13. Dikaiakos, M.D., Florides, A., Nadeem, T., Iftode, L.: Location-aware services over vehicular ad-hoc networks using car-to-car communication. IEEE Journal on Selected Areas in Communications **25**(8) (2007)
14. Dilley, J., Maggs, B.M., Parikh, J., Prokop, H., Sitaraman, R.K., Weihl, W.E.: Globally distributed content delivery. IEEE Internet Computing **6**(5), 50–58 (2002)
15. Dow, C.R., Lin, P.J., Chen, S.C., Lin, J.H., Hwang, S.F.: A study of recent research trends and experimental guidelines in mobile ad hoc networks. In: AINA '05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications, pp. 72–77 (2005)
16. Günes, M., Siekermann, J.: Cosmos - communication scenario and mobility scenario generator for mobile ad-hoc networks. In: Proceedings of the 2nd Int. Workshop on MANETs and Interoperability Issues (MANETII'05) (2005)
17. Hara, T.: Replica allocation methods in ad-hoc networks with data update. Mobile Networks and Applications (MONET) **8**(4), 343–354 (2003)
18. Housel, B.C., Samaras, G., Lindquist, D.B.: Webexpress: a client/intercept based system for optimizing web browsing in a wireless environment. Mobile Networking and Applications **3**(4), 419–431 (1998)
19. Kangasharju, J., Roberts, J.W., Ross, K.W.: Object replication strategies in content distribution networks. Computer Communications **25**(4), 376–383 (2002)

20. Krajzewicz, D., Hertkorn, G., Rossel, C., Wagner, P.: Sumo (simulation of urban mobility); an open-source traffic simulation. In: 4th Middle East Symposium on Simulation and Modelling (MESM2002), pp. 183–187. SCS European Publishing House (2002). Http://sumo.sourceforge.net/ (last accessed January 2007)
21. Mahdy, A.M., Deogun, J.S., Wang, J.: Mobile ad hoc networks: a hybrid approach for the selection of super peers. In: Proceedings of the 2nd IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2005), pp. 280–284 (2005)
22. Mateus, G.R., Goussevskaia, O., Loureiro, A.A.F.: Simulating demand-driven server and service location in third generation mobile networks. In: Proceedings of the 9th International Euro-Par Conference, pp. 1118–1128. Klagenfurt, Austria (2003)
23. Miranda, H., Leggio, S., Rodrigues, L., Raatikainen, K.E.E.: An algorithm for dissemination and retrieval of information in wireless ad-hoc networks. In: Proceedings of the 13th International Euro-Par Conference, pp. 891–900. Rennes, France (2007)
24. Morris, R., Jannotti, J., Kaashoek, F., Li, J., Couto, D.D.: Carnet: A scalable ad hoc wireless network system. In: Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System, pp. 61–65. Kolding, Denmark (2000)
25. N. Bartolini, F.L.P., Petrioli, C.: Optimal dynamic replica placement in content delivery networks. In: 11th IEEE International Conference on Networks (ICON 2003), pp. 125–130. Sydney, Australia (2003)
26. Nadeem, T., Dashtinezhad, S., Liao, C., Iftode, L.: Trafficview: traffic data dissemination using car-to-car communication. SIGMOBILE Mob. Comput. Commun. Rev. **8**(3), 6–19 (2004)
27. Pallis, G., Vakali, A.: Insight and perspectives for content delivery networks. Commun. ACM **49**(1), 101–106 (2006)
28. Pallis, G., Vakali, A., Stamos, K., Sidiropoulos, A., Katsaros, D., Manolopoulos, Y.: A latency-based object placement approach in content distribution networks. In: Third Latin American Web Congress (LA-Web 2005), pp. 140–147. Buenos Aires, Argentina (2005)
29. Pathan, A.M.K., Buyya, R.: A taxonomy and survey of content delivery networks. Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne (2007)
30. Pitoura, E., Chrysanthis, P.K.: Caching and replication in mobile data management. IEEE Data Eng. Bull. **30**(3), 13–20 (2007)
31. Presti, F.L., Petrioli, C., Vicari, C.: Dynamic replica placement in content delivery networks. In: Proceedings of the 13th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005), pp. 357–360. Atlanta, GA, USA (2005)
32. Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley (2002)
33. Rabinovich, M., Xiao, Z., Douglis, F., Kalmanek, C.R.: Moving edge-side includes to the real edge - the clients. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 12–26. Seattle, Washington, USA (2003)
34. Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F.: Automatic fragment detection in dynamic web pages and its impact on caching. IEEE Trans. Knowl. Data Eng. **17**(6), 859–874 (2005)
35. Rilling, L., Sivasubramanian, S., Pierre, G.: High availability and scalability support for web applications. In: SAINT '07: Proceedings of the 2007 International Symposium on Applications and the Internet. IEEE Computer Society, Washington, DC, USA (2007)
36. Roy, S., Ankcorn, J., Wee, S.: Architecture of a modular streaming media server for content delivery networks. In: Proceedings of the 2003 International Conference on Multimedia and Expo (ICME '03), Vol. 3, pp. 569–572. IEEE Computer Society, Washington, DC, USA (2003)
37. Royer, E., Toh, C.: A review of current routing protocols for ad-hoc mobile wireless networks. Personal Communications, IEEE **6**(2), 46–55 (1999)
38. Sailhan, F., Issarny, V.: Energy-aware web caching for mobile terminals. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 820–825. IEEE Computer Society (2002)
39. Sen, S., Rexford, J., Towsley, D.F.: Proxy prefix caching for multimedia streams. In: Proceedings of the IEEE INFOCOM, pp. 1310–1319. New York, USA (1999)

40. Sidiropoulos, A., Pallis, G., Katsaros, D., Stamos, K., Vakali, A., Manolopoulos, Y.: Prefetching in content distribution networks via web communities identification and outsourcing. World Wide Web **11**(1), 39–70 (2008)
41. Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: Analysis of caching and replication strategies for web applications. IEEE Internet Computing **11**(1), 60–66 (2007)
42. T Nadeem, P.S., Iftode., L.: A comparative study of data dissemination models for vanets. Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems (MOBIQUITOUS) (2006)
43. Teng, W.G., Chang, C.Y., Chen, M.S.: Integrating web caching and web prefetching in client-side proxies. IEEE Trans. Parallel Distrib. Syst. **16**(5), 444–455 (2005)
44. Tse, S.S.H.: Approximate algorithms for document placement in distributed web servers. IEEE Trans. Parallel Distrib. Syst. **16**(6), 489–496 (2005)
45. Vakali, A., Pallis, G.: Content delivery networks: status and trends. IEEE Internet Computing **7**(6), 68–74 (2003)
46. Wu, T., Dixit, S.: The content driven mobile internet. Wireless Personal Communications: An International Journal **26**(2–3), 135–147 (2003)
47. Xu, Z., Guo, X., Wang, Z., Pang, Y.: The dynamic cache algorithm of proxy for streaming media. In: Proceedings of the International Conference on Intelligent Computing (ICIC 2005), pp. 1065–1074. Hefei, China (2005)
48. Yin, L., Cao, G.: Supporting cooperative caching in ad-hoc networks. IEEE Trans. Mob. Comput. **5**(1), 77–89 (2006)
49. Yoshimura, T., Yonemoto, Y., Ohya, T., Etoh, M., Wee, S.: Mobile streaming media cdn enabled by dynamic smil. In: Proceedings of the 11th International World Wide Web Conference, WWW2002, pp. 651–661. Honolulu, Hawaii, USA (2002)
50. Zemlianov, A., de Veciana, G.: Capacity of ad hoc wireless networks with infrastructure support. IEEE Journal on Selected Areas in Communications **23**(3), 657–667 (2005)
51. Zhuo, L., Wang, C.L., Lau, F.C.M.: Load balancing in distributed web server systems with partial document replication. In: Proceedings of the 31st International Conference on Parallel Processing (ICPP), p. 305. Vancouver, Canada (2002)

# Chapter 15
# Infrastructures for Community Networks

Thomas Plagemann, Roberto Canonico, Jordi Domingo-Pascual,
Carmen Guerrero, and Andreas Mauthe

## 15.1 Introduction

Content delivery has undergone a sea of changes in recent years. While even only ten years back the major delivery channels were television and radio broadcast, nowadays content is delivered digitally via the Internet or other electronic delivery channels. The engineering problem of delivering multimedia content through the Internet has received much attention by the research community. However, the delivery of content to heterogeneous mobile terminals in a community context still poses many problems. Early Internet based content delivery systems were designed as centralized systems where content is provided from a central server to a large population of the end users (see Fig. 15.1-(a)). This trend is now shifting towards decentralized systems, such as Peer-to-Peer (P2P) systems, in which the role of the content provider and producer is no longer restricted to a few professional content creators. Thus, the content delivery paths is not anymore only from a central server through backbone networks to the end users, but also from one end user to other end user(s) (see Fig. 15.1-(b)). This has been triggered by the emergence of relatively cheap consumer electronics enabling everybody to become a content producer; and

Thomas Plagemann
Department of Informatics, University of Oslo, Norway, e-mail: plageman@ifi.uio.no

Roberto Canonico
Consorzio Interuniversitario Nazionale per l'Informatica CINI - Laboratorio Nazionale per l'Informatica e la Telematica Multimediali ITEM at University of Napoli, Italy, e-mail: roberto.canonico@unina.it

Jordi Domingo-Pascual
Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya, Jordi Girona, 1–3. Campus Nord. Barcelona 08034, Spain, e-mail: jordi.domingo@ac.upc.es

Carmen Guerrero
Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, Spain,
e-mail: carmen.guerrero@uc3m.es

Andreas Mauthe
InfoLab 21, Computing Department, Lancaster University, Lancaster LA1 4WA, UK,
e-mail: andreas@comp.lancs.ac.uk

R. Buyya et al. (eds.), *Content Delivery Networks*,
© Springer-Verlag Berlin Heidelberg 2008

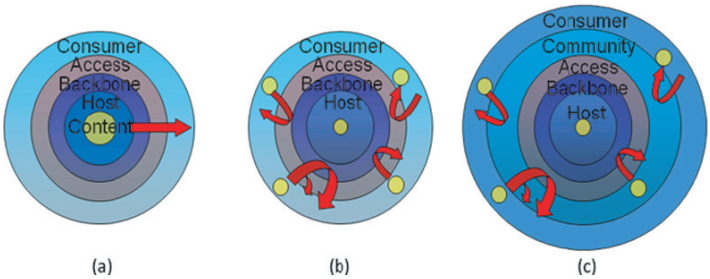



**Fig. 15.1** Content delivery paths: (**a**) traditional, (**b**) today's P2P, (**c**) future community networks to the home (based on [44])

the high penetration of high-speed network access (e.g. xDSL networks) and P2P technologies turning computer users into content providers.

The European *Network-of-Excellence* CONTENT [65] studies future developments in this area with a specific focus on the resulting research challenges related to content delivery for and within community networks. Here, the end users not only consume content but also produce it and provide core elements of the network infrastructure, i.e. the physical community network. Thus, the content delivery path in community networks does not necessarily use any infrastructure provided by Internet Service Providers (ISPs) (see Fig. 15.1-(c)).

While the term "community network" is intuitively well understood it is worthwhile to analyze the concept of community networks. Rosson et al. define [54] community networks as follows:

"*A network community is a group of people whose communication and collaboration over networks strengthens and facilitates their shared identity and goals. The emergence of network communities is a striking example of what might be called grassroots technology development[..] A community network is a special case of a network community in which a physical community coextends with the network community.*"

According to this definition, the community is not only formed by people collaborating through the network, but also by people contributing with their own resources (like in civic and neighborhood networks). Community members mainly provide the access network in the form of several kinds of wireless network technologies, which are connected to the Internet via one or several ISPs. Since a (substantial) part of the content delivery in community networks can be done within the physical community networks without any ISP involvement, there is no evidence that communities might be a larger threat to the Internet than classical Content Delivery Network (CDN) and P2P users, quite the contrary.

With respect to the content delivery, the most important insight is that the "grassroots technology development" in community networks is driven by "people", i.e. the average end users, which might not have any particular education and skills in computer and network administration, software development etc. Thus, decentralization of content delivery must be combined with self-configuring, self-organizing,

self-managing, and self-adapting solutions at all technical layers to minimize the need for human intervention.

Furthermore, Cowan et al. [20] identified in 1998 that content services play a central role:

*"In fact, communities are repositories of large amounts of heterogeneous information that need to be searched, read, explored, acted upon, updated, and that offer opportunities for collaboration and other forms of two-way communication."*

In 1998, multimedia content was not central to this insight. However, we argue that the technological developments in consumer electronics and Information Communication Technologies enable the easy use of multimedia content, and by this create a strong demand for various kinds of content services in community networks. Community members do not only want to consume content, but they want to share it, to search for particular content, to combine artifacts, and to edit complex multimedia objects.

Thus, content delivery and usage is special in the context of community networks for two major reasons: first, autonomic network and overlay solutions are needed to establish and maintain proper CDNs over physical community networks; and second, arbitrary and complex content services (e.g. content adaptation, transcoding, indexing, storage) are needed that go far beyond the simple transfer and consumption of content.

In order to describe the current state and short and long term research challenges, the reminder of this chapter is structured as follows: the following section gives background information on community networks, including a simple architectural framework and related work. The description of industrial challenges and long term research challenges follows this architectural framework. In the conclusions, the most important aspects of content delivery and content service for community networks are summarized.

## 15.2 Background and Related Work

An interesting phenomenon of the last few years is the creation of a number of Wireless Community Networks (WCNs) that provide Internet access in urban areas to community members. These networks were created either by the spontaneous collaboration of people who shared their own xDSL home connection to the Internet, or by the initiative of local institutions. For example, councils and universities have started to offer wireless access to Internet services to user communities (e.g. students) in limited areas (e.g. neighbourhoods, campuses, commercial halls) or public buildings. An example of "institutional" WCN is the Wireless Mesh Networks provided by the Town of Amherst [4] to its citizens.

The most popular "spontaneous" WCN is created by the so called "FON community" [25]. FON members (i.e. *Foneros*) share some of their home xDSL Internet connection and get free access to the Community's FON Spots worldwide. The FON community has also created a business, selling Internet access to those who decide

not to share any connection with the rest of community. Up to now, FON is just acting as a WiFi ISP with just a peculiar business model. Some commercial Internet Service Providers in Europe have already raised concerns about legal issues related to the sharing of residential Internet access [41]. In the future, content services might be provided to the community and thereby increasing its business value. The idea of providing services to the community is already supported by the Ninux.org, an Italian community [43] that provides dynamic-DNS and a SIP-based PBX service to its members.

Interestingly enough, the spontaneous community network model has also proven to be successful in less developed countries, in particular to provide Internet connectivity in rural areas. The Dharamsala Wireless-Mesh community network came to life in February 2005, following the deregulation for outdoor use of WiFi in India. Now the network provides broadband Internet services to a few thousands users. Apart from Internet access, community members use the network for file-sharing applications, off-site backups, playback of high quality video from remote archives and extensive VoIP telephony.

To meet today's and future challenges of content delivery and usage in community networks, it is not sufficient to address individual sub-systems only, like only the CDN. Instead, the entire system, comprising IP based networks, CDNs, content services, and end users must be covered.

### *15.2.1 Architectural Framework*

In the architectural framework depicted in Fig. 15.2, community networks are expected to play a central role in the intermediate future since they provide basic connectivity. In this context, physical community networks are the sum of all the networks that interconnect devices within home environments, neighborhoods, and their combination into multi-hop and mesh networks. Comparable to social networks the primary aim of community networks is to support the local community. Since multimedia content is usually distributed over such networks, several new appealing research issues come up, as for example, mobility, nomadicity, resource assignment, user required/perceived Quality of Service (QoS) and Quality of Experience (QoE), topological robustness, resilience, and network protection.

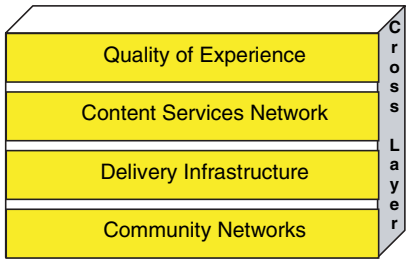



**Fig. 15.2** High level architectural framework

Typically, overlay network solutions are used to implement CDNs, which is captured in our architectural framework by the delivery infrastructure level. They include more and more end users as well, visible as peers and overlay nodes that provide certain resources and services to the network. Overlay networks provide an abstraction that hides the irksome details in the underlying physical networks, e.g. of a wireless mesh network that forms a community network. However, overlay network solutions must also be aware of the basic properties of the underlying community networks to fulfill the non-functional requirements of services, such as resilience and performance. Typical functional aspects of overlays are caching and request routing. They can be solved through networks of proxy caches or distributed hash tables that interconnect peers directly.

On top of the delivery infrastructures, content services networks consist of a set of services for handling multimedia content. These services should support the entire life-cycle of audiovisual content and should also be able to interoperate to create complex services through the combination of several simpler ones. Typical examples of content services are automatic analysis and indexing services for content classification and content abstraction, transcoding services for format adaptation, as well as search services providing sophisticate support for content search.

Finally, on top of the architectural framework is the QoE level, which reflects the actual experience of the end user. In general, QoS is defined as a set of technical parameters capturing mainly quantitative aspects such as throughput, error rate, delay, jitter, frames per second, and bits per pixel. The lower levels of the architectural framework cover networking aware QoS parameters. However, these QoS parameters do not actually reflect the user experience, which depends not only on technical parameters, but also on the effects that failures and faults have on the actual perceived quality. Although QoE is a function of different QoS parameters at network, system and application level, there is not a direct translation between QoS parameters and QoE. Therefore, it has to be established which kind of degradation actually lowers the user experience the least.

Orthogonal to these basic four levels of the architectural framework there are several cross layer issues which are relevant for the scenario of content delivery in community networks. One important class of these cross layer issues is related to QoS parameters at different levels and how they relate and correlate to each other. Another class of cross layer issues is related to the problem that functions at different layers might impact each other, which could in the case of self-adapting solutions lead to cascading effects or unstable system behavior.

### *15.2.2 Community Networks*

Community networks are generally networking infrastructures not owned by ISPs but by individual users or groups of users sharing resources distributed in a relatively small geographical area, like a neighborhood. Providing connectivity to community networks is a challenging task since nodes use a diversity of access technologies and can display a degree of mobility.

Current technologies which may be used for community network infrastructure are: xDSL, Powerline, FTTH for fixed nodes connected to an ISP; WiMAX, MBWA, 3G/UMTS/HSDPA for nodes with wireless access to an ISP; WiFi and Bluetooth for mobile nodes and home networks. Due to the availability of such a large variety of networking technologies, community networks may include nodes not only acting as user terminals, but also as routers, relays, or gateways. Fixed nodes, for instance, may behave as hot-spots, whilst visiting nodes, i.e. devices traveling through the area of the community network, may behave as a mobile gateway, router or terminal.

Within the concept of community networking, multiple networking technologies come together such as mobility with Mobile IPv4 and IPv6, multihoming, network mobility (NEMO), mobile ad-hoc networks (MANETs), wireless mesh networks (WMNs), and even wireless sensor networks (WSNs) and wireless multimedia sensor networks (WMSN). Usually, this interworking of different networking technologies is not pre-planned nor is it managed by operators. Hence, self-configuration capabilities as addressed by autonomic networks are required. In summary, community networks exploit a wide range of network technologies and techniques resulting in a challenging research environment.

The community networking scenario adds extra complexity to the handover process since in addition to handover within the same technology (i.e. *horizontal handover*) the handover between different networking technologies (i.e. *vertical handover*) also has to be supported. In order to efficiently manage such heterogeneous environments, the IEEE 802.21 standard is currently being developed within IEEE. This standard aims at enabling handover and interoperability between heterogeneous network types, including 802 and non-802 networks. The 802.21 standard defines an abstraction layer, providing Media Independent Handover (MIH) functions with the goal of simplifying the management of handovers to and from different access technologies.

As the distribution of multimedia content includes real-time delivery, QoS becomes a key aspect in community networks. QoS provision is still an open issue in wired networks, but it is even more complex in wireless environments. In this context, the evolution of the IEEE 802.11 extensions to provide QoS is crucial for the deployment of Multimedia Wireless. Also, contributions for QoS in MANETs and WMNs are of utmost importance for content delivery in community networks.

### *15.2.3 Delivery Infrastructures*

Delivery infrastructure in the context of this chapter refers to a logical infrastructure created on top of a community network with the specific purpose of enabling access to content services. Most of today's delivery infrastructures mainly aim at the efficient delivery of content to community members. To overcome the limitations of the traditional client/server approach, the P2P paradigm is becoming more and more popular. P2P infrastructures usually implement some form of overlay network

to deploy services that cannot be directly embedded in the underlying network, e.g. multicast routing object location, and event propagation.

Typical supporting services implemented by means of overlays are for instance request routing and actual content delivery. These services can either be implemented with the collaboration of end systems alone, or with support of specialized proxies.

A common theme in the research of delivery infrastructures for community networks is *autonomicity*. Community networks are largely based on collaborating individuals that provide resources to the community network. Therefore, P2P technologies are very important not only in the case of downloading and streaming of stored content, but also for live streaming and every other aspect of resource sharing.

Considering the key building blocks of the widely deployed P2P based CDNs, three basic elements can be distinguished, viz. the *peer-to-peer overlay network*, a specific *content delivery strategy*, and a *caching strategy*. The overlay network is responsible for connecting the participating peers, management of joining and leaving peers, and routing of queries and other messages. The content delivery strategy is responsible for delivering the required content from the source to its destination. The last strategy increases the availability of the content in the P2P system and its efficiency.

The enormous potential and advantages of decentralized infrastructures has already become apparent in the days of Napster. Since then, significant research efforts have been invested in designing self-organized, scalable, robust, and efficient overlay networks. However, it is crucial to note that the performance of a P2P overlay depends on various factors (e.g. application, resources of participating peers, and user behavior) that are less relevant in centralized systems. For example, a specific overlay design can perform well in the case of low churn rate whereas in the case of high churn its performance may decrease to average. Furthermore, content delivery systems pose certain requirements on overlay networks, like finding users that are sharing the demanded files, incentive mechanisms, or enabling efficient inter-peer communication at low costs. Thus, there are many research initiatives to study the direct or indirect influences and dependencies between P2P overlay networks and the underlined networking strategies in a content delivery system.

Considering content delivery strategies, many aspects have to be taken into account separately alongside of interdependencies that might exist. Their influence is crucial for the overall efficiency and performance of a content delivery system. One of the most important aspects is choosing a scheduling strategy for the files to be transmitted. Download strategies as the one used by BitTorrent or network coding are proven to be very efficient for long and large scale downloading sessions [26, 27]. However, with the current trend of content delivery technology, such as Podcasting, new challenges are arising. Therefore, it is necessary to investigate if the aforementioned state-of-the-art strategies are still appropriate, given the requirements of emerging content sharing and delivery strategies.

Not only file sharing, but also the use of live streaming applications is growing fast in community environments. These applications and many others relying on continuous data flows, from IPTV to massive multiplayer online games, have special

needs. They are delay sensitive, need group communication and QoS support. Many solutions have been proposed, but none has been adopted on a wider scale. Nowadays, protocols designed for continuous data flows do not rely exclusively on the classical client/server model, but can also organize the receivers into an overlay network, where they are supposed to collaborate with each other following the P2P paradigm.

Many recent proposals related to Live Audio/Video Streaming using P2P overlays are derived from initial work that extended application-level multicast to the end systems [17]. The first generation control-driven approach focuses on building an initial overlay corresponding to the control plane and is usually implemented as a mesh or a tree. A second overlay, usually a spanning tree, is then created and managed for the actual data transmission. Peercast [12] is the most famous example with a popular implementation and a large audience. A lot of work has been carried out to improve the control plane in order to cope with the high dynamics of the P2P overlay. For example, Nice is using a sophisticated clustering scheme [8]. More recent work tries to improve robustness using a hybrid tree/structure. An example for this is Bullet [31]. A new generation, data-driven approach stresses the need to cope directly with data. Peers exchange data availability and then they choose their neighborhood according to the data they need [8]. Further, epidemic algorithms are currently being proposed in systems such as Donet [63] to improve the data delivery. P2P Live Streaming is already reality. However, so far little has been done to demonstrate their efficiency on a very large scale. Simulation is one way to validate the feasibility of such dynamic infrastructures [50]. An alternative approach is to study proprietary applications in real testbeds, like PlanetLab [46]. The largest P2P Live Streaming deployments are related to IPTV applications and are only associated to proprietary protocols and architectures [41, 47, 48, 53, 56, 57, 58]. Thus, only their behavior but not the protocols itself can be analyzed.

The behavior of peers in a community network plays a key role. At the one end of the scale are altruistic peers that provide resources without expecting any return. At the other end there are so called “free riders” who only consume but do not provide any resources, which is a rational behavior in systems without any sharing incentives. Therefore, it has become clear that some kind of incentive scheme is necessary to achieve an optimal utilization of system resources in a system context as well as for individual peers. This is currently an active research area.

### *15.2.4 Content Services Networks*

On top of the delivery infrastructure resides the content services network. A content services network is an infrastructure that provides a whole range of services to optimize the content experience. Users might be able to access such services for easier navigation, and personalized adaptation of content to their needs. In fact, the idea is to use so called content services in conjunction with the underlying network infrastructure to provide a network of content services and by doing so, create a

content network. Content services network subsumes a number of sub-areas that can be grouped into:

- **Content Services Network Architecture and Services Framework** comprising issues related to the underlying architectural model for content service networks.
- **Service Interaction** encompassing all issues related to service integration and usage in general, such as service discovery, service description, service quality, service level agreement, etc.
- **Service Instances** include specific content services that improve the delivery and user experience in content service networks.

The aim of building a content services network is to integrate, in an open way, tools and mechanisms that enable the creation and re-purposing of assets for the benefit of the communities of users as well as allowing commercial use by innovative companies. In order to achieve this, a suitable model and architecture that allows to easily "plug" such content services into the services network is necessary. Recently, the concept of Service Oriented Architecture (SOA) has been introduced to achieve optimal support for business processes through the underlying IT architecture [13]. The main benefits of a SOA are reusable components that can be easily organized to build flexible and modular applications. Therefore it seems to be an appropriate abstraction for content services networks. At present, the SOA paradigm is mostly realized using Web Services [7].

Two major research issues within a service based content network architecture are related to *service interaction*, i.e. the way services are described and the way appropriate services are discovered. *Service description* is a fundamental issue for ensuring easy user access and a simple management of services. Examples of standards in this area are for instance those defined by the W3C for the Semantic Web [2]. Several formalisms have been proposed, at various expressivity levels, from simple semantic mark-up syntaxes (e.g. RDF [36]) to ontologies (e.g. OWL [19]). An OWL-based Web Service Ontology, OWL-S, has been proposed specifically for Web services, in order to describe their properties unambiguously [37]. A recent initiative defined a *Semantic Web Services Framework* (SWSF) [11], which includes the *Semantic Web Services Language*. There is considerable ongoing work in the area of *service discovery*. Both, UDDI [59] and the ebXML registry [24], for example, support finding services by name, type, and binding according to a taxonomy. Another specification effort is WS-Dynamic Discovery [38], a local area network service discovery mechanism for discovering Web services by using local-scoped multicast.

*Service instances* represent the value added services that are provided within (or at the edge) the communication infrastructure for tailored and adapted content delivery. Many different kinds of services can be envisaged in this context; for example, content adaptation service and QoE. Issues related to content adaptation have been addressed for some time. For instance, Smith, Mohan and Li have presented research dealing with ad-hoc adaptation for heterogeneous terminals. Their work has focused on the definition of techniques for content representation, among which the so-called InfoPyramid [39] plays a major role. Lemlouma and Layaida present in

their work a novel technique for content negotiation [34]. They introduce the Negotiation and Adaptation Core (NAC), a basic system for negotiating and adapting multimedia services to heterogeneous terminals. Lum and Lau highlight in [35] the need for content adaptation and propose to use a Decision Engine as the logical entity in charge of taking decisions on how to adapt a specific content to client's presentation capabilities. Boll, Klas and Wandel propose in [15] a three-stage adaptation strategy, based on Augmentation, i.e. pre-adaptation during which alternative versions of a content are realized; Static Adaptation, i.e. deletion of the non-relevant alternatives; and Dynamic Adaptation, i.e. choice of the most appropriate alternative among those who survived the previous phase.

Previous research efforts towards the assessment of the end user perceived quality are mostly adequate for MPEG-2 videos only. They are based on either objective or subjective procedures. Subjective approaches assume human experience as the only grading factor, i.e. QoE. Objective procedures are performed without human intervention and give more stable results, but do not necessarily reflect the user quality perception. Examples of objective metrics are PSNR, MAE, MSE, and RMSE [52, 60, 61]. The methods for assessing the perceived video quality objectively do not usually take the Human Visual Senses (HVS) sufficiently into account. The human senses cover many errors quite effectively. Thus, objective measurements may not reflect the user perceived quality. Other methods that also consider HVS are therefore required (see [33, 62, 64]). The goal of this work is to provide QoE assessment as a service within the Content Services Infrastructure.

## 15.3 Visionary Thoughts for Practitioners

Industry related and short term research challenges are, in contrast to the long-term research challenges, less speculative and more focused on what can be realized within the coming years. In the following, different aspects in the context of the identified architectural areas are being discussed.

### *15.3.1 Community Networks*

Mobility has been a research topic during the last years and solutions focused on different layers of the OSI stack have been explored. Specifically, the IETF has standardized mobility solutions at the IP layer, i.e. Mobile IPv4 [45] and Mobile IPv6 [29]. In addition, it has standardized three extensions to Mobile IPv6: Fast Handovers [30], Hierarchical Mobile IPv6 [51], and Network Mobility [22].

It has been shown in a number of studies [5, 6] that maintaining the connection while the device is moving is still a big challenge. In addition, these protocols do not explicitly support heterogeneous networking environments. Achieving seamless handover in a heterogeneous environment presents many challenges especially when

considering multihoming. Multihoming [28] is a technique where the main objective is to increase reliability of Internet connections for networks or single nodes. This technique uses multiple interfaces connected to different ISPs. In this way, a multihomed node has different paths available for communication. Many research papers [18, 42, 49] have been published exploiting the benefits of multihoming in static nodes or networks. However, using multihoming in mobile and heterogeneous networking is a relatively new research topic that presents many challenges.

The IEEE 802.21 [23] is a recent effort of the IEEE that aims at enabling handover and interoperability between heterogeneous network types including both 802 and non-802 networks. The IEEE 802.11e task group has refined the 802.11 MAC to provide audio and video applications with QoS guarantees [9]. The recently approved version of IEEE 802.11e introduces an improvement on the DCF algorithm aiming to distinguish traffic categories.

Due to the distributed nature of IEEE 802.11 DCF, the protocol under review is also used for the case of multi-hop communication [14, 16]. Currently, the area of ad hoc and mesh networks [3, 10] enjoys the attention of a significant portion of the scientific community. Communication via multiple hops is closely linked with the problem of routing and is still a hot research topic. Here the knowledge of the available bandwidth in a given area is one of the key factors since most of these routing protocols support QoS based on the bandwidth available within an area.

Finally, in such an open scenario where network management and device's configuration relays on the users misbehavior detection and traffic anomaly detection, security threats such as Distributed Denial of Service (DDoS) attacks should be considered as a critical aspect. Misbehavior detection is especially important in WMN at MAC level while traffic anomaly detection covers the whole CDN.

### *15.3.2 Delivery Infrastructures*

The current challenge to improve P2P-based content delivery infrastructures consists in creating overlays that are better suited to the particular requirements of content services.

Delivery of traditional Web content to user communities may benefit from the possibility of clustering clients according to their network location. One such clustering may be helpful for efficiently moving content replicas or proxy caches towards those parts of the network where clients are more densely distributed. A research proposal for real-time Web clients clustering appeared in [32] and it is based on client IP addresses extraction from the Web server logs and clustering of addresses based on BGP routing information. One such approach may only be pursued in traditional CDNs, where the content provider and the CDN service provider closely cooperate to serve the content provider's objective of optimal content delivery.

The P2P model has been recently applied to Voice over Internet Protocol (VoIP) applications, such as Skype, proving its usefulness for both searching users location

and relaying voice packets. Selecting one or multiple suitable peers to relay voice packets is a critical factor for the quality, scalability, and cost of a VoIP system. However, Ren et al. [53] show that the network probing activity, required for peer selection, may affect the scalability of the whole application and its performance. To reduce the network overhead imposed by several uncorrelated P2P overlays, Nakao et al. [40] propose to establish some basic services into the network (*underlay*) to properly and efficiently support the creation of concurrent application-specific overlays.

Another challenge proposed by P2P applications is the tendency of the huge traffic they produce to "escape" the Traffic Engineering efforts of ISPs [55]. Recently it has been proposed [1] to try and pursue some form of cooperation between P2P applications and ISPs, in order to find a common benefit.

Finally, today's solutions for classical content delivery infrastructures are well designed considering the topology or the wide area network in which they serve. However, they are restricted in the sense that they are not concerned with the last mile to the client and the end user community infrastructures. They regard the last mile just as a link to the client and do not consider the topology of the network connecting the client to the content delivery infrastructure. The recent trends in technology clearly indicate that neighborhood networks and home networks will connect clients to the core CDN. The adaptation of the delivery path also in the neighborhood and end user networks and its proper integration with the wide-area distribution infrastructure is a problem that has yet to be systematically addressed. Combinations of P2P and classical CDNs seem to be one good starting point. Early work on this topic is presented by Cowan et al. [20].

### *15.3.3 Content Services Network*

Content services that are available to typical community network members are mainly concerned with the consumption of audiovisual content, like RealPlayer and Windows Media player. Furthermore, content provisioning is possible through Web based services and streaming services. However, services and applications that are related to the creation and re-purposing of content, including management and editing are not available. Thus, the service offering is currently limited to the provision and consumption of media. Further, there is also little freedom for users to add their own content and create their own communities. Examples such as YouTube and MySpace show that there is a desire for sharing information and content between users. However, in contrast to these examples, in community networks there is a target "audience", i.e. the members of the community.

This implies that on the one hand there is a need for new services that give users more freedom in the way they interact and share content. On the other hand, there should be more services that allow users to create (new) content, set-up their own communities, and control their environment. This goes beyond the existing model (such as FlickR) where users are basically only able to manage and share the

content. In this new model, they would also be able to determine (to a certain extent) how content is delivered (e.g. over a video streaming service), what to do in case of insufficient resources (i.e. what kind of adaptation strategy should be applied), and even what kind of incentive mechanisms should be used. Thus, there should be two different basic service types, viz. the more traditional content services and the content and infrastructure support services. Especially the latter is not sufficiently provided at this moment.

In order to be open and compatible, in this context, it is important to have a service framework that allows different service providers to offer their content or support services. Inspired by the idea behind Web services, a proper content services network architecture needs to be developed so that it provides a framework in which all the different services for optimized delivery and content usage can be placed. This effectively provides an opportunity for commercial and private service providers to offer services in the longer term, within a community content network environment. Such a content service network framework effectively creates a market place for services alongside a more community oriented service provisioning. In order to achieve this fully, the open research questions in the next section have to be addressed first.

## 15.4 Future Research Directions

In this section, the long term research challenges and some of the research directions that are followed by the CONTENT Network-of-Excellence to address these challenges are presented.

### *15.4.1 Community Networks*

Today's research in content delivery related communication is, for instance, dealing with streaming, network caching, QoS, and P2P issues. These are well developed research areas with an established set of researchers addressing different parts of the problem space. For community-based content networks, WMN are becoming more and more important since they can be deployed without having to invest in an expensive wired infrastructure. However, there are still a number of research issues to be addressed in this context, e.g. regarding link quality, channel assignment and routing, gateway selection, etc. These have to be investigated before WMN can be a fully integral part of content networks. It is envisaged that integrated multihomed networks will be functional by the end of the decade, based on the research progress in WMN, network selection and other related research open issues.

In order to provide seamless communication, an *End-to-End (E2E) infrastructure* is required. This infrastructure will integrate different network types under a unifying architecture dealing with aspects such as E2E QoS provision, E2E QoS

routing, and traffic engineering. Another research strand is dealing with misbehavior detection and the protection of content networks from attacks. This research is going to result in *misbehavior-sensitive networks* that provide resilience mechanisms for the detection and protection of the network. A further, parallel development is the *autonomous distribution* of content. This includes autonomic communication architectures based on P2P principles. Issues here are related to the delivery, like P2P streaming, but also to trust, co-ordination and management aspects. Autonomic content delivery will also include certain service aspects (see below). Highly *interactive* applications within community content networks are still a major challenge. By the next decade we envisage that highly heterogeneous infrastructures based on different network types will be able to cope with this and provide the necessary support.

### *15.4.2 Delivery Infrastructures*

The core challenge for future delivery and service infrastructures for community networks is to develop autonomic *Content Networks* (CNs) that integrate autonomic overlay structures and content services, like content management. CNs will improve the reliability and efficiency of traditional CDNs and reduce their management overhead. Furthermore, CNs will also extend the application spectrum of traditional solutions by, for instance, transparently supporting streaming media to mobile users, providing interactive multimedia applications, or adapting them to a community networking scenario. Research in this context requires dealing with the design of a novel architecture for autonomic CNs, including novel methods for linking content management with content delivery, and new protocols for the efficient transport of control information. Research issues that need to be addressed here are related to the actual delivery, but also how to appropriately orchestrate content management, services functions, and communications. The latter can be achieved using cross-layer information flow to better coordinate the different parts.

Current efforts are often only focused on a particular application domain, like VoD, IPTV, or Web browsing, and targeted towards fairly rigid dissemination structures. In contrast, future P2P technologies need to be adaptive and follow a more flexible approach than the rather constrained approaches in the context of a traditional CDN.

How to capture the systems aspects of the related processes and how to facilitate these developments through an appropriate architectural model have not been sufficiently investigated. Important in this context is that the content delivery infrastructure and the content management functionality are well synchronized. In order to achieve this, the area of cross-layer interaction plays a key role. This includes functional and interface work on interaction between the different layers of the communications architecture in order to facilitate the development and implementation of emerging ubiquitous content networks as well as enabling content management environments that allow faster production and easier access.

Future research needs to expedite the convergence of content production and delivery, and bridge the technological gap between the two areas. As a consequence new possibilities for content creation, programme formats, and end-to-end content delivery within one framework are becoming possible.

One of the guiding principles to improve content delivery is adaptation to network conditions. If designed correctly, adaptation can lead to a much better system utilization and efficiency. However, using adaptation in two sub-systems that are independent of each other, i.e. using self-organizing cooperative caching schemes on top of adaptive overlays without any further precaution, can result in a situation where the adaptation at one level thwarts the adaptation carried out in the other system part. Furthermore, the conditions that can trigger adaptation consider only data that is derived through network measurements. Therefore, cross layer issues represent a particular challenge in this context.

### *15.4.3 Content Services Networks*

While more advanced and better content services are needed for the future, it is also important to structure them in a way that allows to combine existing service instances to more complex services. Different service providers can offer services ranging from infrastructure support to actual content provision. The former, for example, can include a service providing a live video streaming infrastructure according to a specified Service Level Agreement (SLA). This service in turn can make use of other infrastructure services (e.g. a QoE assessment service). A user or community group could rent such a service for the distribution of their content and effectively create a content delivery service on top of it.

In order to establish such a framework the supporting concepts and underlying architecture have to be well specified while still leaving room for flexible service provisioning. The services within such a framework themselves form a content services network with each service providing a distinct, self-contained service function. Services can be distributed throughout the infrastructure and form a mesh of coordinated mechanisms using either standard service interfaces for their coordination or service specific protocols. The role of the content service architect is to allow different services to be placed into the overall service framework and make them part of the content network infrastructure. Services in this context range from *infrastructure services* (e.g. QoS and QoE assessment) over *delivery support services* (such as transcoding and content adaptation), to *content centric services* (e.g. video summarization and indexing).

The service architecture follows a generic Service Oriented Architecture (SOA) model. The service model provides a generic service specification that deals with aspects all content services have to conform to. This description leaves sufficient scope for individual services to provide their own specification detailing the full service interface and functionality. A content service has to provide a set of interfaces through which it communicates with other services or applications. The internal

organization and service structure is not part of this model, neither is the service specific interface description or service specific functionality specification.

A service can be stateful or stateless. Stateful services have to provide an interface to the service user to query the state of its execution. It should also be possible to enter into an agreement about the provisioning of QoS. This requires the specification of an SLA. The SLA itself is service specific and its format needs to be specified in the service description. Context awareness refers to the services that take information from the application or environmental context to control and manage the service. Through cross-layer interaction the service retrieves information from underlying layers and system components. Through this it becomes more aware of the system environment and can either adapt or try to influence the underlying components. User interaction allows the specification of preferences by the user in order to adapt the service to user needs. Figure 15.3 shows the generic content service model.

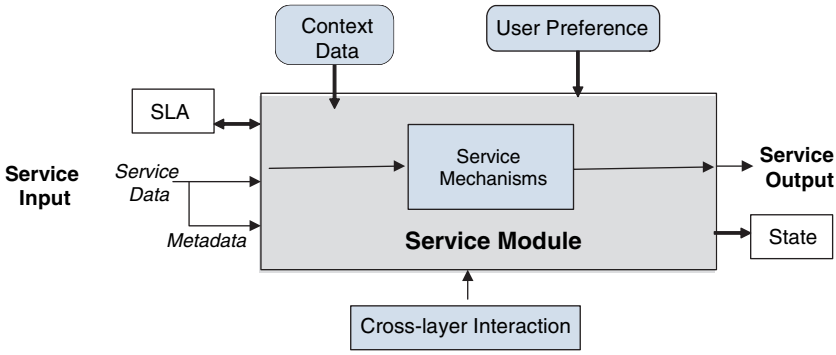



**Fig. 15.3** Content service model

The content service framework provides the context within which the services are placed. Crucial in the context of the service framework is the service description and its representation within the service registry. Services can use other services through this service registry via standard interfaces. Such an approach allows dynamic and automatic composition of content services and opens up new business opportunities for brokerage services.

### *15.4.4 Cross Layer Issues*

It is generally accepted in the research community that besides their advantages, layered system architectures have also clear disadvantages. In order to enable resource aware distributed applications, access to network layer information is necessary. Cross layered approaches are used to achieve this kind of awareness beyond layer interfaces, but they are designed for particular solutions. Thus, understanding

and developing a better architectural solution than strict layering is an important research challenge in general. However, cross layer issues are especially important in the context of future CNs for community networks since autonomic solutions, like self-adapting functions, need to be applied. As mentioned earlier, independent adaptation of different functions might influence each other since they share resources. For instance, both might have an impact on network traffic. The first step towards addressing this challenge is to identify a set of metrics for each layer, including QoS parameters and resource consumption parameters and to model their dependencies between the layers. This first step seems trivial, but to carry it out successfully, this set of metrics and their definitions need to be accepted and used by the entire research community working in this area. Nowadays, many different and incompatible metrics and definitions are used. Modeling the dependency among parameters needs also to include the understanding of the functional behavior of the system elements. To provide the proper tools for this challenge, the CONTENT Network-of-Excellence investigates the development of a generic benchmarking suite for CNs following a modular approach in which the different levels of a CN might be considered as the system under test and the other levels represent the environment and the workload.

## 15.5 The CONTENT Approach

The CONTENT architectural framework does not provide a blueprint for the implementation of community based content networks, it much rather provides guidelines and develops basic principles according to which such networks can be developed. In order to validate the proposed principles and mechanisms within the framework a number of aspects are currently being assessed. The strategy hereby is to implement key elements and assess their performance through measurements and simulations. This is carried out in the context of the three architectural layers, or in the case of cross layer activities, related to inter-layer aspects. We illustrate this in the following with three sample research activities and results in CONTENT.

At the community network level, simulation and measurement in a real testbed is being used for studying performance and QoS in the case of mobile terminals performing both vertical and horizontal handovers. Also simulation is used to validate new proposals for available bandwidth estimation in wireless networks. Finally, measurements in real testbeds are being made to analyze and define the appropriate metrics for QoS at network level and other metrics which may be useful for upper layers. As a sample of the preliminary results obtained, Fig. 15.4 shows a comparison of the instantaneous available bandwidth estimation in a community network using both the *pathChirp* tool and our proposal. The proposal under study is based on in-line measurements and does not provoke congestion to make the estimation of the available bandwidth. The graph shows how this new proposal behaves and approximates the real available bandwidth.

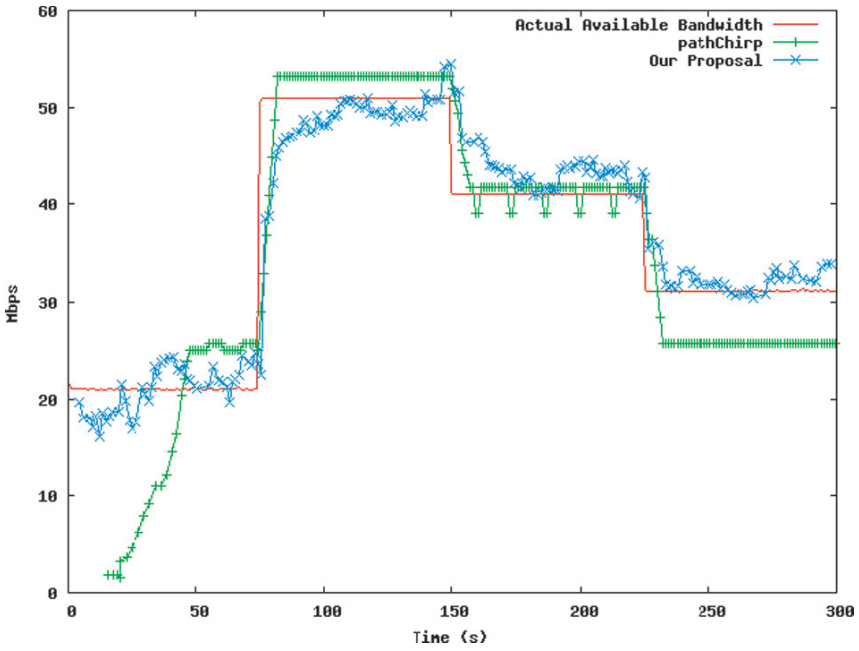



**Fig. 15.4** Available bandwidth estimation

As part of the delivery infrastructure the principles of P2P caching are for example being investigated. Initial work in this area has focused on how P2P caches can be structured, dynamically established, and how the different elements are coordinated. The goal of P2P caching is to bring content close to the user. However, in contrast to "normal" caching, caches are not required after a content item has reached a certain popularity since at this stage it will be widely available in the vicinity. The idea is that peers are elected (based on request frequency) to join the P2P cache. Content is cached according to requests and after this content is available, it is taken from the cache. A simulation study has been carried out assuming between 5000 and 7000 nodes and different download scenarios. The study shows that the major overhead is caused by coordination interaction between the caching nodes. This is offset by bandwidth savings due to bringing content closer. It is also found that the bandwidth that can be saved is considerable, whereas the additional effort is marginal. However, the size of the content items and access patterns are crucial. Further work can be carried out to establish how this changes with varying download speeds and content penetration scenarios.

Besides the use of simulation tools, some prototypes are being developed to show the applicability of the research results in a realistic scenario in the field of content delivery in community networks. Several application scenarios are identified, based on existing commercial services, to validate the architectural framework. In particular, we investigate a VoD application scenario that enhances a community Web portal with video and by building a P2P application as add-on to their client-based

community portal. Both the community members and the community provider can offer videos for downloading. These videos can be for free or paid content. For example, a golf player's community can offer typical paid content such as professional golf videos (e.g. report of a PGA tournament). It can also offer private videos but paid content (e.g. a video of a golf trainer about how to improve your practice) or totally free content.

## 15.6 Conclusion

Community networks provide many opportunities for new content services as well as for the communication and interaction of community members. In this model the community members provide the resources in terms of networks and nodes, e.g. in the form of wireless mesh networks, and they provide, manage, and use content. Since these are typical end users that do not necessarily have special training in network management and system administration, autonomic solutions at the network and overlay level are very important to reduce the necessary human intervention in order to establish and maintain content delivery infrastructures. Wireless networks and mobility play an important part in these delivery infrastructures. Therefore, it is important that services can be dynamically adapted to available resources. To provide the foundation for self-adapting solutions, one of the most important research challenges is to understand and model cross layer interactions and dependencies among functions and among metrics. Furthermore, services need to be dynamically composed out of simple service instances to exactly provide the services that is required by the users with respect to their functional needs and the available resources.

**Acknowledgements** This work has been supported by the CONTENT Network-of-Excellence, which is funded by the European Commission in the 6th Framework Programme (Project No. IST-FP6-038423). It includes contributions from all project partners. Therefore, the authors would like to especially thank their colleagues from University Pierre and Marie Curie (Paris 6), University of Coimbra, National and Kapodistrian University of Athens, Technische Universität Darmstadt, AGH University of Science and Technology, and Delft University of Technology.

## References

1. Aggarwal V, Feldmann A, Scheideler C (2007) Can ISPS and P2P users cooperate for improved performance. SIGCOMM Computer Communications Review, Vol. 33, no. 3, pp. 29–40
2. Akkiraju R, Farell J, Miller JA, Nagarajan N, Sheth A, Verma K (2005) Web Service Semantics - WSDL-S. W3C Workshop on Frameworks for Semantics in Web Services
3. Akyildiz IF, Wang X, Wang W (2005) Wireless mesh networks: a survey. Computer Networks, Vol. 47, no. 4, pp. 445–487
4. http://www.amherstma.gov/departments/Information_Technology/community_wireless.asp

5. Aparicio AC, Serral-Gracià R, Jakab L, Domingo-PascualJ (2005) Measurement Based Analysis of the Handover in a WLAN MIPv6 Scenario. Dovrolis C (Ed.): Passive and Active Measurements 2005 Boston USA, LNCS 3431, pp. 207–218
6. Aparicio AC, Julian-Bertomeu H, Núñez-Martínez J, Jakab L, Serral-Gracià R, Domingo-Pascual J (2005) Measurement-Based Comparison of IPv4/IPv6 Mobility Protocols on a WLAN Scenario. Networks UK, Publishers of the HET-NETs '05 Technical Proceedings (ISBN 0-9550624-0-3) Ilkley, UK
7. Austin D, Barbi A, Ferris C, Garge S (2004) Web Services Architecture Requirements. W3C Working Group Note, http://www.w3.or/TR/wsa-reqs/
8. Banerjee S, Lee S, Bhattacharjee B, Srinivasan A (2003) Resilient multicast using overlays. In Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 102–113, New York, NY, USA, ACM Press
9. Joe I, Batsell SG (2000) Reservation CSMA/CA for Multimedia Traffic over Mobile Ad-hoc Networks. ISCC'2000, Antibes – Juans les Pins, France
10. Aguayo D, Bicket J, Biswas S, Judd G, Morris R (2004) Link-Level Measurements from an 802.11b Mesh Network. Proc. of ACM SIGCOMM 2004, Portland, Oregon, USA
11. Battle S, Bernstein A, Boley H, Grosof B, Gruninger M, Hull, R, Kifer M, Martin D, McIlraith S, McGuinness D, Su J, Tabet S (2005) Semantic Web Services Framework (SWSF) Overview Version
12. Bawa M, Deshpande H, Garcia-Molina H (2002) Streaming live media over peers. HotNets-I, Princeton, NJ, pp. 107–112
13. Berbner R, Heckmann O, SteinmetzR (2005) An Architecture for a QoS driven Composition of Web Service based Workflows. Proceedings of Networking and Electronic Commerce Research Conference (NAEC2005) Riva del Garda, Italy
14. Biswas S, Morris R (2005) Opportunistic Routing in Multi-Hop Wireless Networks. Proceedings of SIGCOMM 2005, Philadelphia, PA, USA, pp. 69–74
15. Boll S, Klas W, Wandel J (1999) A cross-media adaptation strategy for multimedia presentations. Proceedings of the seventh ACM International Conference on Multimedia (Part 1), pp. 37–46, New York, NY, USA
16. Broch J, Maltz DA, Johnson DB, Hu YC, Jetcheva J (1998) A performance comparison of multi-hop wireless ad hoc network routing protocols. Mobile Computing and Networking, pp. 85–97
17. Chu YH, Rao SG, Seshan S, Zhang H (2002) A Case for End-System Multicast. In IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multicast Communications, Vol. 20, Issue 8, pp. 1456–1471
18. Cidon I, Rom R, Shavitt Y (1999) Analysis of multi-path routing. IEEE/ACM Transactions on Networking, Vol. 7, no. 6, pp. 855–867
19. Dean M, Connolly D, van Harmelen F, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA (2002) OWL Web Ontology Language 1.0 Reference. W3C Working Draft
20. Cowan DD, Mayfield CI, Tompa FW, Gasparini W (1998) New role for community networks. Communications of the ACM, Vol. 41, Issue 4
21. Darlagiannis V, Mauthe A, Steinmetz R (2006) Sampling Cluster Endurance for Peer-to-Peer based Content Distribution Networks. Proceedings of Multimedia Communication and Networking (MMCN 2006, part of IS&T/SPIE Electronic Imaging 2006 Symposium)
22. Devarapalli V, Wakikawa R, Petrescu A, Thubert P (2005) Network Mobility (NEMO) Basic Support Protocol. RFC 3963
23. Dutta A, Das S, Famolari D, Ohba Y, Taniuchi K, Kodama T, Shulzrinne H (2005) Seamless Handover across Heterogeneous Networks - An IEEE 802.21 Centric Approach. Proceedings of WPMC2005
24. ebXML Registry Services and Protocols, Committee Draft 01 (2005)
25. http://www.fon.com
26. C. Gkantsidis, T. Karagiannis, P. Rodriguez, M. Vojnovic Planet Scale Software Updates, ACM/SIGCOMM'06, Pisa. Sep 2006

27. C. Gkantsidis and P. Rodriguez, Network Coding for Large Scale Content Distribution, IEEE/INFOCOM'05, Miami. March 2005.
28. G. Huston, Architectural Approaches to Multi-homing for IPv6, RFC 4177, 2005
29. Johnson D, Perkins C, Arkko J (2004) IP Mobility Support for IPv6. RFC 3775
30. Koodli R, ed (2005) Fast Handovers for Mobile IPv6 RFC 4068
31. Kostic D, Rodriguez A, Albrecht J, Vahdat A (2003) Bullet: high bandwidth data dissemination using overlay mesh. ACM SIGOPS Operating Systems Review, ACM SOSP
32. Krishnamurthy B, Wang J (2000) On network-aware clustering of Web clients. Proceedings of ACM SIGCOMM
33. Kuhmünch C, Kühne G, Schremmer C, Haenselmann T (2001) Video-scaling algorithm based on human perception for spatio-temporal stimuli. Technical Report Lehrstuhl Praktische Informatik IV, University of Mannheim, Germany
34. Lemlouma T, Layada N (2003) Media resources adaptation for limited devices. Proceedings of the 7th ICCC/IFIP International Conference on Electronic Publishing, Universidadedo Minho, Portugal
35. Lum WY, Lau FCM (2002) A context-aware decision engine for content adaptation. IEEE Pervasive Computing, 1(3):41–49
36. Manola F, Miller E (2004) RDF Primer. W3C Recommendation
37. Martin D, ed (2003) OWL-S: Semantic Markup for Web Services. Technical Overview (associated with OWL-S Release 1.1)
38. Microsoft Corporation I (2004) Web Services Dynamic Discovery (WS-Discovery)
39. Mohan R, Smith JR, Li CS (1999) Adapting multimedia internet content for universal access. IEEE Transactions on Multimedia, 1(1):104–114
40. Nakao A, Peterson L, Bavier A (2003) A Routing Underlay for Overlay Networks. Proceedings of the ACM SIGCOMM Conference
41. http://wiki.ninux.org/
42. Ogier R, Ruenburg V, Shacham N (1993) Distributed algorithms for computing shortest pairs of disjoint paths. IEEE Transactions on Information Theory, Vol. 93, no. 2, pp. 443–456
43. (2006) Wi-Fi service breaches ISP conditions. OUT-LAW News, 27/09/2006. http://www.out-law.com/page-7335
44. Parker A (2005) P2P: Opportunity of Thread. Panel presentation at IEEE Workshop on Web Content Caching and Distribution, Sophia Antipolis, France
45. Perkins C (2002) IP Mobility Support for IPv4. RFC 3344
46. PlanetLab: http://www.planet-lab.org/
47. PPLive web site, http://www.pplive.com.
48. PPStream web site, http://www.ppstream.c
49. Raju J, Garcia-Luna-Aceves JJ (1999) A new approach to on-demand multipath routing. IEEE ICCCN
50. Silverston T, Fourmaux O (2006) Source vs Data-Driven Approach for Live P2P Streaming. Proceedings of IEEE ICN 2006, Mauritius
51. Soliman H, Castelluccia C, Malki EK, Bellier L (2005) Hierarchical Mobile IPv6 Mobility Management (HMIPv6). RFC 4140
52. Stockhammer T, Hannuksela MM, Wiegand T (2003) H.264/AVC in Wireless Environments. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, Issue 7, pp. 657–673
53. Ren S, Guo L, Zhang X (2006) ASAP: an AS-Aware Peer-relay protocol for high quality voIP. Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS'06), Lisbon, Portugal
54. Rosson MB, Carroll JM (1998) Network communities, community networks. CHI 98 conference summary on Human factors in computing systems CHI '98
55. Seetharaman S, Ammar M (2006) On the Interaction between Dynamic Routing in the Native and Overlay Layers. IEEE INFOCOM
56. SOPcast web site, http://www.sopcast.com.
57. http://tibtec.org/?q=node/60

58. TVants web site, http://www.tvants.com.
59. UDDI Spec Technical Committee (2003) UDDI Version 3.0.1. http://uddi.org/pubs/uddi_v3.htm
60. Verscheure O, Frossard P, Hamdi M (1999) User-Oriented QoS Analysis in MPEG-2 Video Delivery. Journal of Real-Time Imaging, special issue on Real-Time Digital Video over Multimedia Networks, Vol. 5, no 5, pp. 305–314
61. Wiegand T, Schwarz H, Joch A, Kossentini F, Sullivan GJ (2003) Rate-constrained coder control and comparison of video coding standards. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, no 7, pp. 688–704
62. Winkler S (2005) Digital Video Quality - Vision Models and Metrics. Wiley
63. Zhang X, Liu J, Li B, Yum TP (2005) Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. Proceedings IEEE Infocom
64. Wang Z, Sheikh HR, Bovik AC (2003) Objective Video Quality Assesment. In The Handbook of Video Databases: Design and Applications, eds Furht B, Marqure O, CRC Press, pp. 1041–1078
65. CONTENT Network of Excellence, IST-FP6-038423. http://www.ist-content.eu/

# Chapter 16
# Internetworking of CDNs

Mukaddim Pathan, Rajkumar Buyya, and James Broberg

## 16.1 Introduction

The current deployment approach of the commercial Content Delivery Network (CDN) providers involves placing their Web server clusters in numerous geographical locations worldwide. However, the requirements for providing high quality service through global coverage might be an obstacle for new CDN providers, as well as affecting the commercial viability of existing ones. It is evident from the major consolidation of the CDN market, down to a handful of key players, which has occurred in recent years. Unfortunately, due to the proprietary nature, existing commercial CDN providers do not cooperate in delivering content to the end users in a scalable manner. In addition, content providers typically subscribe to one CDN provider and thus can not use multiple CDNs at the same time. Such a closed, non-cooperative model results in disparate CDNs. Enabling coordinated and cooperative content delivery via *internetworking* among distinct CDNs could allow providers to rapidly "scale-out" to meet both flash crowds [2] and anticipated increases in demand, and remove the need for a given CDN to provision resources.

CDN services are often priced out of reach for all but large enterprise customers. Further, commercial CDNs make specific commitments with their customers by signing Service Level Agreements (SLAs), which outline specific penalties if they fail to meet those commitments. Hence, if a particular CDN is unable to provide Quality of Service (QoS) to the end user requests, it may result in SLA violation and end up costing the CDN provider. Economies of scale, in terms of cost effectiveness and performance for both providers and end users, could be achieved by

Mukaddim Pathan
GRIDS Lab, Department of CSSE, The University of Melbourne, Australia, e-mail: apathan@csse.unimelb.edu.au

Rajkumar Buyya
GRIDS Lab, Department of CSSE, The University of Melbourne, Australia, e-mail: raj@csse.unimelb.edu.au

James Broberg
GRIDS Lab, Department of CSSE, The University of Melbourne, Australia, e-mail: brobergj@csse.unimelb.edu.au

R. Buyya et al. (eds.), *Content Delivery Networks*,
© Springer-Verlag Berlin Heidelberg 2008

leveraging existing underutilized infrastructure provided by other CDNs. For the purposes of this chapter, we term the technology for interconnection and interoperation between CDNs as "peering arrangements" of CDNs or simply "CDN peering", which is defined as follows:

**Definition of 'peering arrangement'** – *A peering arrangement among CDNs is formed by a set of autonomous CDNs* $\{CDN_1, CDN_2, \ldots, CDN_n\}$*, which cooperate through a mechanism* ***M*** *that provides facilities and infrastructure for cooperation between multiple CDNs for sharing resources in order to ensure efficient service delivery. Each* $CDN_i$ *is connected to other peers through a 'conduit'* $C_i$*, which assists in discovering useful resources that can be harnessed from other CDNs.*

While the peering of CDNs is appealing, the challenges in adopting it include designing a system that virtualizes multiple providers and offloads end user requests from the primary provider to peers based on cost, performance and load. In particular we identify the following key issues:

- *When to peer?* The circumstances under which a peering arrangement should be triggered. The initiating condition must consider expected and unexpected load increases.
- *How to peer?* The strategy taken to form a peering arrangement among multiple CDNs. Such a strategy must specify the interactions among entities and allow for divergent policies among peering CDNs.
- *Who to peer with?* The decision making mechanism used for choosing CDNs to peer with. It includes predicting performance of the peers, working around issues of separate administration and limited information sharing among peering CDNs.
- *How to manage and enforce policies?* How policies are managed according to the negotiated SLAs. It includes deploying necessary policies and administering them in an effective way.

Therefore, an ad-hoc or planned peering of CDNs requires fundamental research to be undertaken to address the core problems of measuring and disseminating load information, performing request assignment and redirection, enabling content replication and determining appropriate compensation among participants on a geographically distributed "Internet" scale. Moreover, to ensure sustained resource sharing between CDN providers, peering arrangements must ensure that sufficient incentive exists for all participants [18]. These issues are deeply interrelated and co-dependent for a single CDN. However, they must now be considered in a coordinated and cooperative manner among many peered CDNs, whilst satisfying the complex multi-dimensional constraints placed on each individual provider. Each provider must ensure that their individual SLAs are met when serving content for its own customers to end users, while meeting any obligations it has made when participating in a group of many providers.

In this chapter, we present an approach for CDN peering, which helps to create "open" CDNs that scale well and can share resources with other CDNs, and thus evolving past the current landscape where non-cooperative CDNs exist. In our architecture, a CDN serves end user requests as long as the load can be handled by itself. If the load exceeds its capacity, the excess end user requests are offloaded to

the CDN network of the peers. We also present two new models to support peering of CDNs and identify the challenges associated with realizing these models.

The remainder of the chapter is organized as follows. In Sect. 16.2, we demonstrate the significance and relevance of CDN peering. Next we present the related work highlighting their shortcomings. In Sect. 16.4, we present our approach for CDN peering, followed by the new models to assist CDN peering. Then we discuss the challenges in implementing peering CDNs. In Sect. 16.7, we also identify related core technical issues to be addressed. Finally, we conclude the chapter in Sect. 16.8.

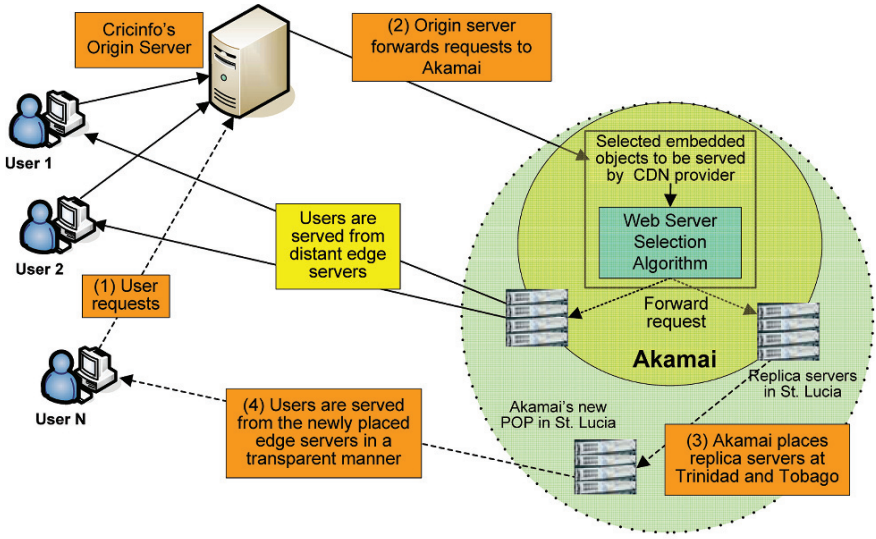
## 16.2 Significance of CDN Internetworking

As noted in earlier chapters, popular Web sites often suffer congestion, bottlenecks, and even lengthy downtime due to large demands made on the resources of the provider hosting them. As discussed in Chap. 11, this phenomenon can manifest itself as instances of unexpected flash crowds resulting from external events of extreme magnitude and interest or sudden increases in visibility after being linked from popular high traffic Websites like Slashdot[1] or Digg.[2] Increases in demand on Web servers can also be more predictable, such as the staging of a major events like the Olympic Games or the FIFA World Cup. The level of demand generated for many popular Web sites can often be impossible to satisfy using a single Web server, or even a cluster. In 1998, the official Soccer World Cup Website received 1.35 billion requests over 3 months, peaking at 73 million requests per day, and 12 million requests per hour [2]. Similarly high volumes were seen during the 1998 Winter Olympic Games, with the official Website servicing 56.8 million requests on a peak day (and a maximum of 110,414 requests per minute) [13]. During Sept. 11, 2001, server availability approached 0 % for many popular news Websites with pages taking over 45 sec. to load, if at all [15]. Given that end users will wait as little as 10 sec. before aborting their requests, this can lead to further bandwidth and resource wastage [12].

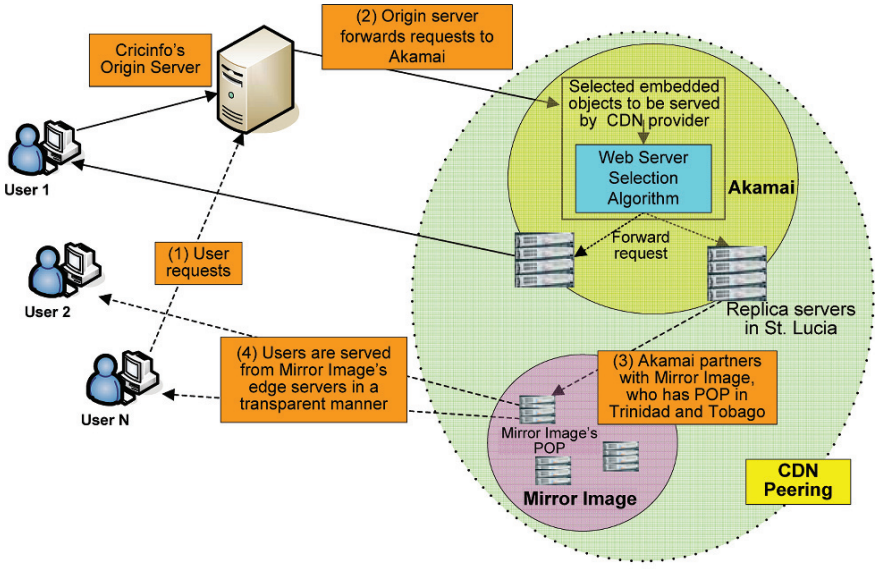
Peering CDNs could be a solution to handle flash crowds, Web resources overprovisioning, and adverse business impact. It is evident that significant gains in cost effectiveness, performance, scalability and coverage could be achieved if a framework existed that enabled peering between CDNs to allow coordinated and cooperative load sharing. To better understand the peering of CDNs, consider the following scenario in Fig. 16.1. Suppose that the ICC Cricket World Cup is being held in the Caribbean, and www.cricinfo.com is supposed to provide live media coverage. As a content provider, www.cricinfo.com has an exclusive SLA with the CDN provider, Akamai [10]. However, Akamai does not have a Point of Presence (POP) in Trinidad and Tobago (a Caribbean island), where most of the cricket matches will be held.

[1] http://www.slashdot.org

[2] http://www.digg.com




(a) Expansion of a CDN provider through placing new POP




(b) Peering between CDN providers

**Fig. 16.1** A CDN peering scenario

As being the host of most of the cricket matches, people of this particular part of Caribbean are expected to have enormous interest in the live coverage provided by www.cricinfo.com. Since Akamai is expected to be aware of such event well in advance, its management can take necessary initiatives to deal with the evolving situation. In order to provide better service to the clients, Akamai management might decide to place its surrogates in Trinidad and Tobago, or they might use their other distant edge servers (as shown in Fig. 16.1(a)). Firstly, placing new surrogates just for one particular event would be costly and might not be useful after the event. On the other hand, Akamai risks its reputation if it can not provide agreed QoS for client requests, which could violate the SLA and still cause profit reduction. Hence, the solution for Akamai could involve cooperating with other CDN provider(s) to form a peering arrangement in order to deliver the service that it could not provide otherwise (depicted in Fig. 16.1(b)).

Peering arrangements between CDNs may vary in terms of the purpose, scope, size, and duration. We anticipate that in case of flash crowds, such a peering arrangement should be automated to react within a tight time frame—as it is unlikely that a human directed negotiation would occur quickly enough to satisfy the evolved niche. In case of long-duration events (as in Fig. 16.1), we would expect negotiation to include a human-directed agent to ensure that any resulting decisions comply with participating companies' strategic goals.

## 16.3 Related Work

Internetworking of resource providers is gaining popularity in the research community. An example of such a research initiative is InterGrid [3], which describes the architectures, mechanisms, and policies for internetworking grids so that grids can grow in a similar manner as Internet. Analyses of previous research efforts suggest that there has been only modest progress on the frameworks and policies needed to allow peering between providers. In CDNs context, the reasons for this lack of progress range from technological problems that need solving, to legal and commercial operational issues for the CDNs themselves. For CDNs to peer, they need a common protocol to define the technical details of their interaction as well as the duration and QoS expected during the peering period. Furthermore, there can often be complex legal issues involved (e.g. embargoed or copyrighted content) that could prevent CDNs from arbitrarily cooperating with each other. Finally, there may simply be no compelling commercial reason for a large CDN provider such as Akamai to participate in CDN peering, given the competitive advantage that its network has the most pervasive geographical coverage of any commercial CDN provider.

The internet draft by Internet Engineering Task Force (IETF) proposes a Content Distribution Internetworking (CDI) Model [9], which allows CDNs to have a means of affiliating their delivery and distribution infrastructure with other CDNs who have content to distribute. According to the CDI model, each content network treats neighboring content networks as *black boxes*, which uses commonly defined

protocol for content internetworking, while internally uses its proprietary protocol. Thus, the internetworked content networks can hide the details from each other. The CDI Internet draft assume a federation of CDNs but it is not clear how this federation is built and by which relationships it is characterized.

A protocol architecture [21] for CDI attempts to support the interoperation and cooperation between separately administered CDNs. In this architecture, performance data is interchanged between CDNs before forwarding a request by an authoritative CDN (for a particular group), which adds an overhead on the response time perceived by the users. Moreover, being a point-to-point protocol, if one end-point is down the connection remains interrupted until that end-point is restored. Since no evaluation has been provided for performance data interchange, the effectiveness of the protocol is unclear.

CDN brokering [3] allows one CDN to intelligently redirect end users dynamically to other CDNs in that domain. This DNS-based system is called as Intelligent Domain Name Sever (IDNS). The drawback is that, the mechanism for IDNS is proprietary in nature and might not be suitable for a generic CDI architecture. Although it provides benefits of increased CDN capacity, reduced cost, and better fault tolerance, it does not explicitly consider the end user perceived performance to satisfy QoS while serving requests. Moreover, it demonstrates the usefulness of brokering rather than comprehensively evaluating a specific CDN's performance.

Amini et al. [1] present a peering system for content delivery workloads in a federated, multi-provider infrastructure. The core component of the system is a peering algorithm that directs user requests to partner providers to minimize cost and improve performance. However, the peering strategy, resource provisioning, and QoS guarantees between partnering providers are not explored in this work.

From a user-side perspective, Cooperative Networking (CoopNet) [15] provides cooperation of end-hosts to improve network performance perceived by all. This cooperation between users is invoked for the duration of the flash crowd. CoopNet is found to be effective for small Web sites with limited resources. But the main problem of the user-side mechanisms is that they are not transparent to end users, which are likely to restrict their widespread deployment. Hence, it can not be used as a replacement and/or alternative for cooperation among infrastructure-based CDNs.

CoDeeN [16, 23] provides content delivery services, driven entirely by end user demands. However, utilizing its services is not transparent to the end users, as they require them to "opt-in" by setting their browser proxy manually to interact with the CoDeeN network. This user-driven approach means that CoDeeN is essentially an elaborate caching mechanism rather than a true CDN. The authors also noted that the system could be easily abused by bandwidth hogs, password crackers, and licensed content theft, requiring CoDeeN to implement some rudimentary measures such as IP blacklisting and privilege separation for local and external users. Currently, CoDeeN only runs on PlanetLab nodes. Cooperation with external content providers is mentioned by the authors but has yet to be explored.

CoralCDN [11] utilizes a novel Peer-to-Peer (P2P) DNS approach to direct users to replica nodes in the CoralCDN overlay network, reducing the stress on origin servers and improving performance for users. CoralCDN is a cooperative network,

but there is no means for nodes (or providers) to participate in peering or internetworking with nodes that are outside of PlanetLab. The nodes that can participate are only offered a coarse level control over their participation (such as allowing individual servers to specify their maximum peak and steady-state bandwidth usage) but there is no fine grained control over exactly what content a node has agreed to serve, nor are there service guarantees. Naturally, given that the service is free and research oriented, content is served on a best effort basis and no compensation is given for participating nodes.

Globule [19, 20] is an open-source collaborative CDN that allows almost any Web-hosting server to participate by installing a customized Globule Apache model, leveraging the ubiquitous nature of Apache as the leading Web server platform. Globule enables server-to-server peering, ad-hoc selection, creation, and destruction of replicas, consistency management and relatively transparent redirection (via HTTP or DNS) of clients to high-performing replicas. Participants in the Globule CDN can act as a *hosting server*, a *hosted server*, or both. This means they can serve content for other users sites as well as their own, in addition to leveraging other participants resources to replicate their own sites. Bandwidth and resource limits can be applied to hosted servers but depend on appropriate facilities being available on the hosting server to enforce this (such as bandwidth limiting Apache modules and "jail" environments to cap resource usage) rather than being handled by Globule itself. A brokerage service is offered where participants can register and access other participants' details in order to initiate negotiations for hosting requests. Such negotiations could include pricing and compensation agreements but this has not been explored deeply in Globule. Security and data integrity aspects (such as dealing with malicious users) are recognized but still remain an open problem for the Globule CDN.

DotSlash [25] is a community driven "mutual" aid service that offers support for small sites that would not have the resources to cope during instances of flash crowds. Provided the site in question has configured itself to access DotSlash, the service automatically intervenes during the flash crowd, allocating and releasing "rescue" servers depending on the load, and is phased out once the flash load passes. A service directory is utilized to allow participants to find each other easily. Participants in DotSlash can only exist in three fixed and mutually exclusive states—*SOS state* where a participant is overloaded and receiving help from other participants, *rescue state* where a participant is aiding another participant in SOS state, and *normal state*. Given the community-driven nature of DotSlash, there is no facility available for internetworked nodes to receive compensation (monetary or resources in-kind) for participating in the peering arrangement.

## 16.4 Architecture for CDN Internetworking/Peering

Internetworking between different CDNs remains an open problem. The notion of CDN internetworking through a peering mechanism is appealing as a means to address unexpected flash crowds, as well as anticipated short or long term increases in

demand, when a single CDN has insufficient resources. They could also allow CDNs (that may not have resources in a particular location) to utilize the resources of other CDNs, by forming a peering arrangement. Thus, peering CDNs can address localized increases in demand for specific content. However, as discussed in Sect. 16.3, many *collaborative* CDNs exist, who function in isolation from each other and commercial CDNs operate with differing policies, methodologies, and QoS expectation. As such, in order for these disparate CDNs to peer, we need to formalize the manner in which they will peer, how they interact, and how QoS levels are set and managed.

In previous work [5, 17], we have presented a policy-driven peering CDNs framework (depicted in Fig. 16.2). The terminologies used to describe the system architecture are listed in Table 16.1. The initiator of a peering negotiation is called a *primary* CDN; while other CDNs who agree to provide their resources are called *peering* CDNs. The endpoint of a peering negotiation between two CDNs is a contract (SLA) that specifies the peer resources (Web servers, bandwidth etc.) that will be allocated to serve content on behalf of the primary CDN. The primary CDN manages the resources it has acquired insofar that it determines what proportion of the Web traffic (i.e. end user requests) is redirected to the Web servers of the peering CDNs.

Figure 16.3 illustrates the typical steps to create a peering arrangement. We summarize these steps in the following:

***Step 1.*** *Creation of a peering arrangement starts when the (primary) CDN provider realizes that it cannot handle a part of the workload on its Web server(s). An initialization request is sent to the mediator.*

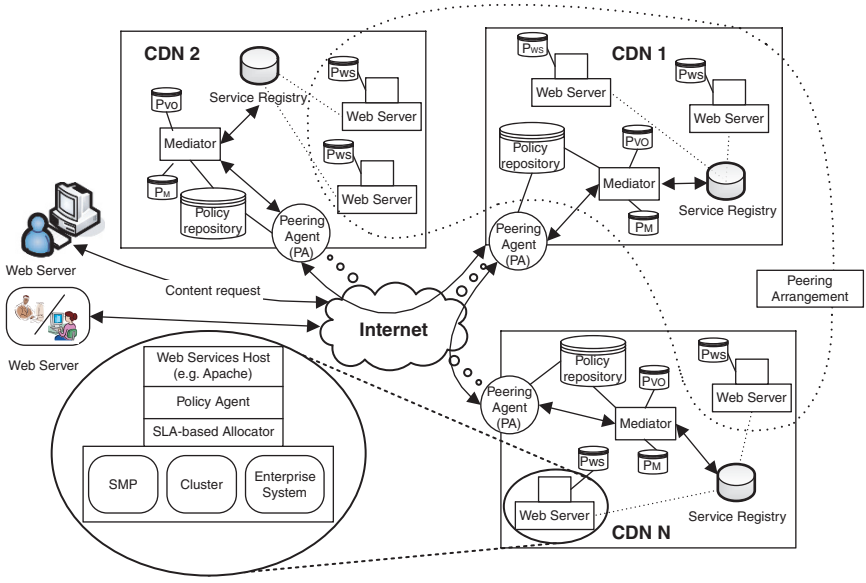



**Fig. 16.2** Architecture of a system to assist the creation of peering CDNs

**Table 16.1** List of commonly used terms

| Terminology | Description |
|---|---|
| *Web server (WS)* | A container of content |
| *Mediator* | A policy-driven entity, authoritative for policy negotiation and management |
| *Service registry (SR)* | Discovers and stores resource and policy information in local domain |
| *Peering Agent (PA)* | A resource discovery module in the peering CDNs environment |
| *Policy repository (PR)* | A storage of Web server, mediator and peering policies |
| $P_{WS}$ | A set of Web server-specific rules for content storage and management |
| $P_M$ | A set of mediator-specific rules for interaction and negotiation |
| $P_{Peering}$ | A set of rules for creation and growth of the peering arrangement |

***Step 2.*** *The mediator instance obtains the resource and access information from the SR, whilst SLAs and other policies from the PR.*
***Step 3.*** *The mediator instance on the primary CDN's behalf generates its service requirements based on the current circumstance and SLA requirements of its customer(s). Hence, it needs to be expanded to include additional resources from other CDNs.*
***Step 4.*** *The mediator instance passes the service requirements to the local Peering Agent (PA). If there are any preexisting peering arrangements (for a long term scenario) then these will be returned at this point. Otherwise, it carries out short term negotiations with the PA identified peering targets.*
***Step 5.*** *When the primary CDN acquires sufficient resources from its peers to meet its SLA with the customer, the new peering arrangement becomes operational. If no CDN is interested in such peering, peering arrangement creation through re-negotiation is resumed from Step 3 with reconsidered service requirements.*

An existing peering arrangement may need to either disband or re-arrange itself if any of the following conditions hold: (a) the circumstances under which the peering was formed no longer hold; (b) peering is no longer beneficial for the participating

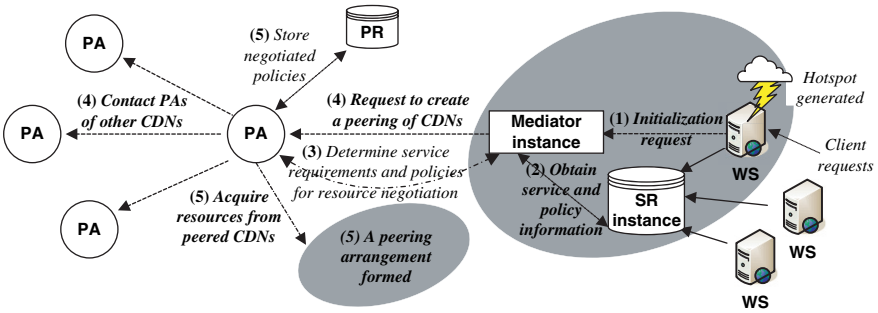



**Fig. 16.3** Typical steps for creating a peering arrangement

CDNs; (c) an existing peering arrangement needs to be expanded further in order to deal with additional load; or (d) participating CDNs are not meeting their agreed upon contributions.

We have chosen to adapt the IETF policy-based framework to administer, manage, and control access to network resources [24]. Whilst the usage of such a framework has received preliminary investigation for individual CDNs [22], it had not been considered under a framework with multiple peering CDNs. The policy framework consists of four basic elements: *policy management*, *policy repository*, *policy enforcement point (PEP)*, and the *policy decision point (PDP)*.

In the standard IETF policy framework, the admin domain refers to an entity which administers, manages, and controls access resources within the system boundary. An administrator uses the policy management tools to define the policies to be enforced in the system. The PEPs are logical entities within the system boundary, which are responsible for taking action to enforce the defined policies. The policies that the PEPs need to act on are stored in the policy repository. The results of actions performed by the PEPs have direct impact on the system itself. The policy repository stores polices generated by the administrators using the policy management tools. The PDP is responsible for retrieving policies from the policy repository, for interpreting them (based on *policy condition*), and for deciding on which set of policies are to be enforced (i.e. *policy rules*) by the PEPs. Choosing where these logical elements reside in a CDN system will obviously have a significant effect on the utility and performance experienced by participating CDNs and end users, and must be considered carefully and specifically depending on the particular CDN platform that is implementing them.

A policy in the context of peering CDNs would be statements that are agreed upon by the participants within the group of peered CDNs. These statements define what type of contents and services can be moved out to a CDN node, what resources can be shared between the participants, what measures are to be taken to ensure QoS based on negotiated SLAs, and what type of programs/data must be executed at the origin servers.

The proposed model for peering CDNs in Fig. 16.2 has been mapped to the IETF policy framework, as shown in Table 16.2. The policy repository is responsible for storing policies generated by the policy management tool used by the administrator of a particular peering group of CDNs – typically the initiator of the grouping. The policy repository virtualizes the Web server, mediator, and peering policies. These policies are generated by the policy management tool used by the administrator of a particular peering group. The distribution network and the Web server components (i.e. Web Services host, Policy Agent, SLA-based Allocator) are the instances of PEPs, which enforce the peering CDN policies stored in the repository. The peering agent and mediator are instances of the PDPs, which specify the set of policies to be negotiated at the time of collaborating with other CDNs, and pass them to the peering agent at the time of negotiation. The policy management tool is administrator dependent, and will vary depending on the CDN platform. A direct benefit of using such policy-based architecture is to reduce the cost of operating of CDNs by promoting interoperability through a common peering framework, and thus allowing CDNs to meet end user QoS requirements under conditions of heavy load.

**Table 16.2** Policy mapping

| Policy Framework Component | Peering CDNs Component | Specified Policies | Description |
|---|---|---|---|
| *System* | *Peering CDNs* | All policies in the system | The distributed computing and network infrastructure for peering CDNs |
| *Admin domain* | *Peering arrangement* | Negotiated peering policies | An administrative entity for resource management and access control |
| *Policy management tool* | *Administrator dependent* | – | An administrator dependent tool to generate policies |
| *Policy repository* | *Policy repository* | Web server, peering and mediator policies | Storage of policies in the system |
| *Policy Enforcement Points (PEPs)* | *Web Services host, Policy Agent, SLA-based allocator* | Web server policies | A logical entity which ensures proper enforcement of policies |
| *PDPs* | *Mediator* | Mediator policies, peering policies | An authoritative entity for retrieving policies from the repository |

### *16.4.1 Performance Gain Through Peering*

We develop the performance models based on the fundamentals of queuing theory to demonstrate the effects of peering between CDNs and to characterize the QoS performance of a CDN.

It is abstracted that $N$ independent streams of end user requests arrive at a conceptual entity, called *dispatcher*, following a Poisson process with the mean arrival rate $\lambda_i, i \in \{1, 2, \ldots, N\}$. The dispatcher acts as a centralized scheduler in a particular peering relationship with independent mechanism to distribute content requests among partnering CDNs in a user transparent manner. If, on arrival, a user request can not be serviced by CDN $i$, it may redirect excess requests to the peers. Since this dispatching acts on individual requests of Web content, it endeavors to achieve a fine grain control level. The dispatcher follows a certain policy that assists to assign a fraction of requests of CDN $i$ to CDN $j$.

For our experiments, we consider an established peering arrangement consisting of three CDNs. It is assumed that the total processing of the Web servers of a CDN is accumulated and each peer contains same replicated content. The service time of each CDN's processing capability follows a general distribution. The term 'task' is used as a generalization of a request arrival for service. We denote the processing requirements of an arrival as 'task size'. Each CDN is modeled as an M/G/1 queue

with highly variable Hyper-exponential distribution which approximates a heavy-tailed Bounded Pareto service distribution (α, k, p) with variable task sizes. Thus, the workload model incorporates the high variability and self-similar nature of Web access.

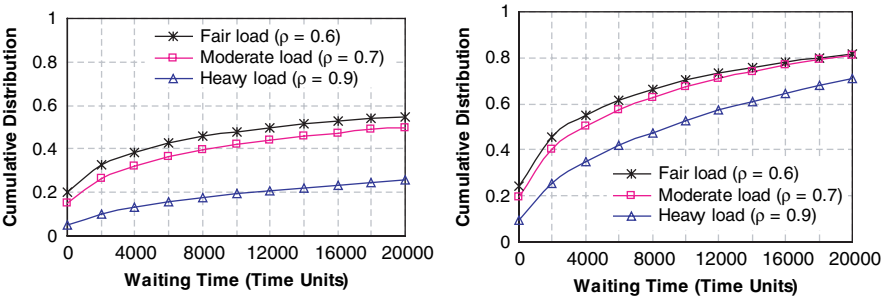
In our performance models, participating providers are arranged according to a non-preemptive Head-Of-the-Line (HOL) priority queuing system. It is an M/G/1 queuing system in which we assume that user priority is known upon their arrival to a CDN and therefore they may be ordered in the queue immediately upon entry. Thus, various priority classes receive different grades of service and requests are discriminated on the basis of known priority. In our model, an incoming request (with priority $p$) joins the queue behind all other user requests with priorities less than or equal to $p$ and in front of all the user requests with priority greater than $p$. Due to this nature of the peering CDNs model, the effect of peering can be captured irrespective of any particular request-redirection policy.

For our experiments, we consider the expected waiting time as an important parameter to evaluate the performance of a CDN. The expected waiting time corresponds to the time elapsed by a user request before being served by the CDN. In our peering scenario, we also assume an SLA of serving all user requests by the primary CDN in less than 20000 time units.

#### 16.4.1.1 QoS Performance of the Primary CDN

First, we provide the evidence that a peering arrangement between CDNs is able to assist a primary CDN to provide better QoS to its users. The Cumulative Distribution Function (C.D.F) of the waiting time of the primary CDN can be used as the QoS performance metric. In a highly variable system such as peering CDNs, the C.D.F is more significant than average values.

Figure 16.4(a) shows the C.D.F of waiting time of the primary CDN without peering at different loads. From the figure, we see that for a fair load $\rho = 0.6$, there is about *55 %* probability that users will have a waiting time less than the threshold




(a) without peering (b) in a peering arrangement

**Fig. 16.4** Cumulative distribution of waiting time of the primary CDN

of *20000* time units. For a moderate load $\rho = 0.7$, there is about *50 %* probability to have a waiting time below the threshold, while for a heavy load $\rho = 0.9$, the probability reduces to $> 24\,\%$.

Figure 16.4(b) shows the C.D.F of the primary CDN with peering at different loads. By comparing Fig. 16.3(a) and Fig. 16.3(b), it can be found that for a fair load $\rho = 0.6$, there is about *80 %* probability that users will have a waiting time less than the threshold of *20000* time units. Therefore, in our scenario, peering assists the primary CDN to achieve a QoS performance improvement of about *31 %*. For a moderate load $\rho = 0.7$, there is $> 81\,\%$ probability for users to have waiting time below the threshold, an improvement of about *38 %*. For a heavily loaded primary CDN with $\rho = 0.9$, the probability is about *70 %*, which leads to an improvement of $> 65\,\%$. Moreover, for loads $\rho > 0.9$, still higher improvement can be predicted by the performance models. Based on these observations, it can be stated that peering between CDNs, irrespective of any particular request-redirection policy, achieves substantial QoS performance improvement when comparing to the non-peering case.

#### 16.4.1.2 Impact of Request-Redirection

Now, we study the impact of request-redirection on the expected waiting time of users on the primary CDN. A request-redirection policy determines which requests have to be redirected to the peers. We have evaluated different request-redirection policies within the peering CDNs model. Here, we only demonstrate the performance result using *Uniform Load Balanced (ULB)* request-redirection policy that distributes the redirected content requests uniformly among all the peering CDNs. Our aim is to show that even with a simple request-redirection policy, our performance model exhibits substantial performance improvement on the expected waiting time when compared to the non-peering case.

In our experiments, no redirection is assumed until primary CDN's load reaches a threshold load ($\rho = 0.5$). This load value is also used as the *baseline load* for comparing waiting times at different primary CDN loads. Any load above that will be 'shed' to peers. Each peer is ready to accept only a certain fraction (acceptance threshold) of the redirected requests. Any redirected request to a given peer exceeding this acceptance threshold is simply dropped to maintain the system equilibrium. We consider lightly loaded peers (load of peer 1 and peer 2 are set to $\rho = 0.5$ and $\rho = 0.4$ respectively), while tuning the primary CDN's load ($0.1 \leq \rho \leq 0.9$). It can be noted that a weighted average value of waiting time is presented in order to capture the effect of request-redirection.

From Fig. 16.5, we find that, without request-redirection when the primary CDN's load approaches to *1.0*, the user perceived performance (in terms of waiting time) for service by the primary CDN tends to infinity. On the other hand, with request-redirection the waiting time of the primary CDN decreases as the requests are redirected to the peers. It is observed that for a fair load $\rho = 0.6$, there is about *43 %* reduction in waiting time, while for a moderate load $\rho = 0.7$, it becomes about

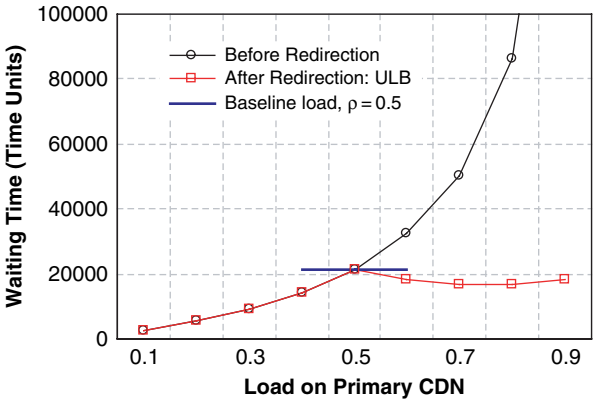



**Fig. 16.5** Impact of request-redirection on waiting time of the primary CDN for uniform request-redirection policy

*66 %*, and for a heavy load $\rho = 0.9$, it reaches to $> 90\,\%$. From the results, it is clear that even a naive request-redirection policy like ULB can guarantee that the maximum waiting time is below *20000* time units (as per the SLA). Therefore, better performance results can be anticipated with a scalable and efficient request-redirection policy. Our results also confirms that redirecting only a certain fraction of requests reduces instability and overload in the peering system because the peers are not overwhelmed by bursts of additional requests.

## 16.5 New Models for CDN Peering

In this section, we propose two new models to assist CDN peering. They are *brokering-based* and *QoS-driven (customized) brokering-based* models. They can be used to complement our peering CDNs model presented in Sect. 16.4. To better understand the uniqueness of these endorsing models and to compare them with existing ones, we first revisit *conventional*, *P2P-based*, and *Internetworked/peered* CDNs. Then we present our newfangled ideas for forming peering CDNs. In Table 16.3, we compare the existing and proposed CDN models and summarize their unique features.

### *16.5.1 Existing CDN Models*

In a *conventional* CDN, end users request content from a particular content provider's Web site. The actual content itself is served by the CDN employed by the content provider from the edge server nearest the end user. There is typically an

**Table 16.3** Comparison of CDN models

| Features | Typical CDN Models | | Advanced Models for CDN Peering | | |
|---|---|---|---|---|---|
| | Conventional CDNs | P2P-Based CDNs | Peering CDNs | Brokering-Based | QoS-Driven (Customized) Brokering-Based |
| Nature of Content Delivery | Based on Web server Collaboration | Based on peering and content availability | Based on CDN internet-working/peering | Based on CDN performance | Based on user defined QoS (Customized) |
| Responsibility for effective content delivery | CDN Provider | Peers/Users | Primary CDN Provider | Content Provider | Content Provider |
| Entities in agreement | CDN-Content Provider | No real agreement (Self-interested users) | CDN-Content Provider, CDN-CDN | CDN-Content Provider | CDN-Content Provider |
| Agreement nature | Static | N/A | Short-term or long-term | Policy-based | Dynamic |
| Scalability | Limited | High | High | High | High |
| Cooperation with external CDNs | No | No | Yes | Yes | Yes |
| Cooperation between CDNs | No | No | Yes | No, CDNs work in parallel | No, CDNs work in parallel |
| Cooperation between users | No | Yes | No | No | No |

agreement between the content provider and the CDN provider specifying the level of service that the content provider expects its end users to receive, which may include guaranteed uptime, average delay, and other parameters. Examples of conventional CDNs include Akamai, Limelight Networks, and Mirror Image. They are typically singular entities that do not collaborate with each other to deliver content and meet their service obligations. This approach is most suited to providers that already have pervasive, globally deployed infrastructure and can deploy edge servers close to the majority of their customers, and have enough capacity to deal with peak loads (caused by flash crowds) when their occur. Whilst cooperation between CDNs does not occur, the Web servers of a CDN cooperate among themselves (collaborative content delivery) to ensure content is replicated as needed and all SLAs are met. Responsibility for effective content delivery rests solely on the CDN provider that has agreed to deliver content on behalf of a content provider.

In a *P2P-based* CDN, content providers utilize end users nodes (either fully or as a supplement to a traditional CDN) in order to deliver its content in a timely and efficient manner. Examples of P2P-based CDNs include CoDeeN, Coral, and Globule. The first two are deployed on the volunteer nodes in PlanetLab, while the third runs on end user nodes. CoopNet and DotSlash are other examples where the first allows end users to cooperate during the period of flash crowds to improve user perceived network performance; and the latter is a community-driven "mutual" aid service to alleviate flash crowds. In this type of CDNs, end users can cooperate to improve the performance perceived by all, especially in the same geographical area as many users around the same edge can assist each other in receiving content. This cooperation can be invoked dynamically in the time of need (flash crowds). No real agreement exists that defines a minimal level of participation from contributing end users, making specific QoS targets hard to enforce for content providers. Given that the users themselves are self-interested entities that receive no compensation for participating in such a peering arrangement, they will only perform content delivery when it suits them.

In *Internetworked/peered* CDNs, like the conventional CDNs, a content provider employs a particular CDN provider to serve its content to end users. The chosen CDN could peer with other CDN(s) to assist it to deliver content and meet any SLA it may have established with the content provider. Examples of peering CDNs include IETF CDI model [9], CDN brokering [3], peering of multi-provider content delivery services [1] and our peering CDNs [5, 17]. However, we note that it is ultimately the primary CDN provider's responsibility to ensure that the target QoS level is met. In this case, end users request for content from a particular content provider's Web site. Content can be served by any CDN in the peering relationship. A centralized dispatcher (or an authoritative CDN) within a particular peering relationship, typically run and managed by the initiator of the peering, is responsible for redirecting requests to multiple peers. The agreement between multiple CDNs is separate from that made between a content provider (customer) and the primary CDN. As such, the originating CDN is responsible for the performance of any peering CDN it employs to meet its obligation to the content provider.

### *16.5.2 Brokering-Based Peering CDNs*

Figure 16.6 shows the first of the two models that we propose to assist the creation of peering CDNs. In this case, "cooperative" content delivery is achieved by the content provider, who leverages the services of multiple CDNs to ensure appropriate geographical coverage and performance targets are met. Content provider has the responsibility for efficient content delivery. The interaction flows are: (1) users request content from the content provider by specifying its URL in the Web browser. Client's request is directed to content provider's origin server; (2) the content provider utilizes a brokering system of its own in order to select CDN(s) for delivering content to the end users. A given content provider can select multiple CDNs (based on a CDN's QoS performance, capabilities, current load, and geographical location) for delivering content to its users. The selected CDNs do not need to be aware that they are working in parallel with each other, as the content provider handles the management and separation of responsibilities; (3) a *policy-based* agreement between the content provider and CDN(s) is established; (4) once peering is established, the proprietary algorithm of the selected CDN(s) chooses optimal Web server to deliver desired content to the user.

In order to join in a peering arrangement according to this model, CDN providers can compete each other to provide improved performance. Content provider will keep track of CDNs' performance. Hence, selection of CDN(s) can be based on history information on performance for similar content. It can also give preferential treatment to its users based on certain policy (can be as simple as "receive service according to payment" or any other complex policy).

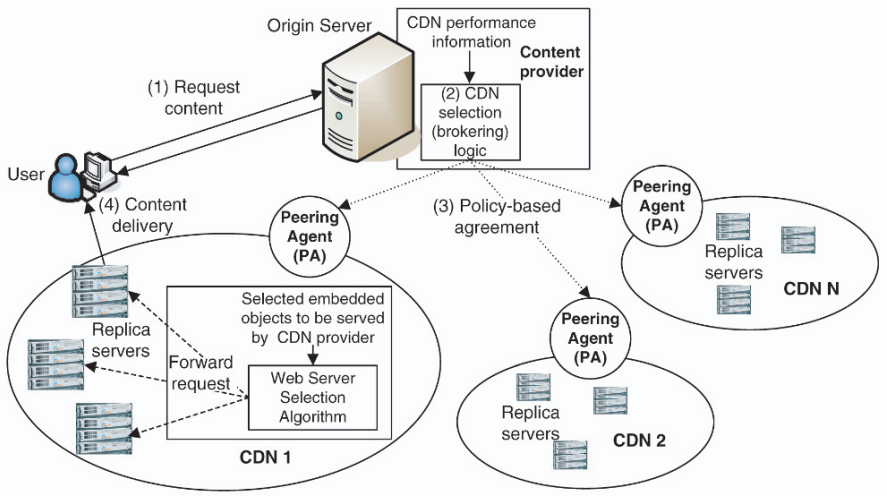



**Fig. 16.6** Brokering-based approach to form peering CDNs

### *16.5.3 QoS-Driven (Customized) Brokering-Based Peering CDNs*

While the model in the previous section considers the performance of each potential participant for creating peering CDNs, it does not specifically consider the QoS required by the end users. Users can have dynamic requirements depending on situations (e.g. flash crowds) that will "customize" content delivery. Therefore, sophistication on user-defined QoS is required to be adopted in the model, which may depend on the class of users accessing the service. Hence, in Fig. 16.7 we show an improvement on the previous model to assist peering CDNs formation. In this model, content provider performs the participant selection dynamically based on the individual user (or a group of users) QoS specifications. The interaction flows are: (1) users requests content from the content provider with specific QoS requirements and it reaches the content provider's origin server; (2) content provider uses a dynamic algorithm (based on user-defined QoS) to select CDN(s); (3) content provider establishes *dynamic* agreement with the CDNs it utilizes to ensure user QoS targets are met; (4) once peering is established with the selected CDN(s), desired content is delivered from the optimal Web server of the selected peer(s).

Such peering arrangements are user-specific and they vary in terms of QoS target, scope, size, and capability. It is evident that content provider has the responsibility for effective content delivery through dynamic peering arrangements. Thus, if a particular peering arrangement fails to meet the target QoS to effectively deliver content to the users, content provider re-negotiate with the CDN providers to establish new peering arrangement(s). In Fig. 16.7, we show that in the initial peering arrangement, CDN 1 is responsible for delivering content to the users. As the user QoS requirements change (shown in dotted line), content provider revokes the (customized) CDN selection logic to re-establish a new peering arrangement. In new

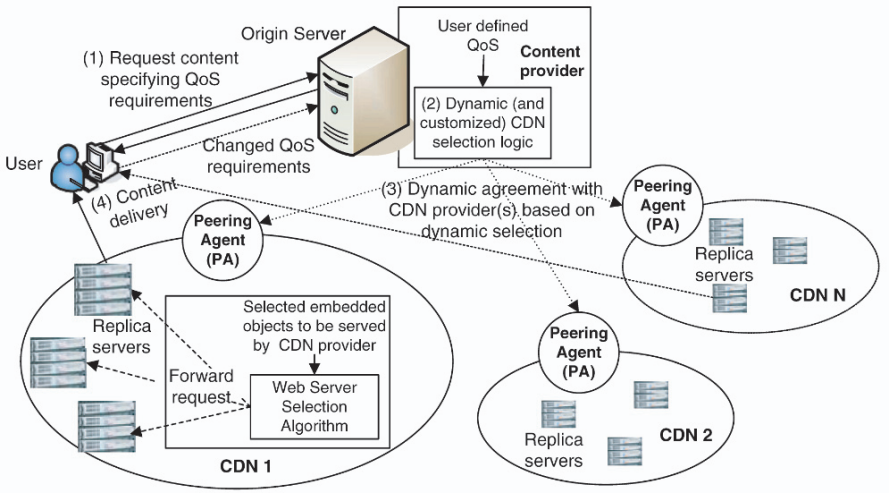



**Fig. 16.7** QoS-driven (customized) brokering-based approach to form peering CDNs

peering arrangement, CDN N is the new participant, which delivers content to the end users from its Web server.

## 16.6 Challenges in Implementing the CDN Peering

There are a number of challenges, both technical and non-technical (i.e. commercial and legal), that have blocked rapid growth of peering between CDNs. They must be overcome to promote peering CDNs. In this section, we outline some of the more common stoppers for uptake of CDN peering.

- *Legal/copyright issues.* There can often be complex legal issues associated with the content to be delivered (e.g. embargoed or copyrighted content) that could prevent CDNs from arbitrarily cooperating with each other. Interactions between peering partners must consider any legal issues associated with the content to be served when delegating it to participating mirror servers from different CDN providers. For instance, if a content provider needs some software or documents that contained logic or information that was embargoed by certain governments (i.e. its access is restricted), all participating CDN providers would have to ensure this was enforced to comply with the appropriate laws. Currently, academic CDNs such as CoDeeN and Coral offer little to no control on the actual content a participating node delivers, and as such participants in these systems could be inadvertently breaking these laws. Content that is copyrighted (e.g. publications, digital media) needs to be carefully managed to ensure that the copyright holder's rights are respected and enforced. The operation (e.g. caching and replication) of some CDNs are user-driven rather than initiated by the content provider, who would prefer to distribute their content on their own terms rather than have it populated in caches worldwide without their consent.
- *Global reach.* As discussed in the previous section, the most common scenario for CDN providers is a centrally managed, globally distributed infrastructure. Companies such as Akamai and Mirror Image have their own far-reaching global networks that cover the vast majority of their customers needs. Indeed, their pervasive coverage is essentially their competitive advantage, and allows them to target the higher end of the customer market for these services. However, few providers can match their global reach, and as such they have little commercial or operational incentive to peer with other smaller providers.
- *Consolidation in CDN market.* Direct peering might be advantageous for small CDN providers, if they wish to compete with larger providers based on coverage and performance. In recent years there has been an enormous consolidation of the CDN marketplace from 20-30 providers down to 5-10 providers of note. It is clear that smaller providers found it difficult to compete on coverage and performance with Akamai and Mirror Image, and subsequently ceased operation or were acquired by the larger providers.
- *Challenges in brokering-based CDN peering.* An approach where a content provider itself manages the selection and contribution of many CDNs to distribute

its content seems appealing, especially, if they have the resources and know-how to manage such an effort. CDN providers could be chosen on their respective merits (e.g. locality, performance, price) and their efforts combined together to provide a good experience for their customers. However, enforcing QoS to ensure a good end user experience (essentially trying to create a robust and predictable overlay network) could be challenging when dealing with multiple providers, especially when they are not actually collaborating, rather simply operating in parallel.

- *Challenges in P2P-based CDN peering.* There has been a growing trend in the last decade toward exploiting user-side bandwidth to cooperatively deliver content in a P2P manner. Whilst initially this started against the wishes of content providers (e.g. Napster, Gnutella), eventually content providers embraced P2P technology, in particular BitTorrent, in order to distribute large volumes of content with scalability and performance that vastly exceeded what was possible with a traditional globally distributed CDN. Content providers have utilized this effectively to distribute digital media (movies, music), operating systems (e.g. Linux) and operating systems patches, games and game patches. With end user bandwidth increases as a result of the proliferation of high-speed broadband, content providers leverage the masses, which upload data segments to peers as they download the file themselves. However, this approach is only effective for popular files, and can lead to poor end user experience for a content that is not being 'seeded' by enough users. As such, it is difficult for content providers to guarantee any particular QoS bounds when the nodes distributing the content are simply end users themselves that may have little motivation to cooperate once they have received their data.
- *Lack of incentives for cooperation.* Further complicating the widespread dependence of this approach is a backlash by Internet Service Providers (ISPs) who are unhappy with the content providers pushing the burden and cost of content delivery onto end users (and subsequently the ISPs themselves). Many ISPs are now actively blocking or throttling BitTorrent and other P2P traffic in response to this trend, to minimize increased utilization and reduction in revenue per user and the resulting cost it places on the ISP in provisioning additional capacity. Many ISPs in more geographically isolated countries (on the so-called 'edges') such as Australia and New Zealand are in particularly unique situations, depending on a small number of expensive data pipes to North America and Europe. As a result, the broadband access offered by ISPs in these regions have fixed data quotas (rather than 'unlimited') that end users are restricted to, in order to ensure they remain profitable. These conditions further discourage widespread adoption and participation by end users in cooperative content delivery.

## 16.7 Technical Issues for Peering CDNs

Proper deployment of peering CDNs exhibits unique research challenges. In this section, we present some of those unique issues that are to be addressed for peering

CDNs. While there are some solutions existing for related problems in the CDN domain, the notion of internetworking/peering of CDNs poses extra challenges. Therefore, we provide a research pathway by highlighting the key research questions for the realization of peering CDNs.

### *16.7.1 Load Distribution for Peering CDNs*

The load distribution strategy for peering CDNs includes *request assignment* and *redirection*, *load dissemination*, and *content replication*. Coordination among these core issues is another important consideration for successful exploitation of load distribution strategy.

*Request redirection* and *assignment* to geographically distributed Web servers of peers requires considering end user's location, server loads, and link utilization between the end user and server in addition to task size (i.e. processing requirements of a content request). It should also address the need to handle dynamically changing conditions, such as flash crowds and other unpredictable events. Request assignment and redirection can be performed in a CDN at multiple levels – at the DNS, at the gateways to local clusters and also (redirection) between servers in a cluster [7, 8]. Commercial CDNs predominantly rely on DNS level end-user assignment combined with a rudimentary request assignment policy (such as weighted round robin, or least-loaded-first) which updates the DNS records to point to the most appropriate replica server [10]. In the peering CDNs, end-users can be assigned via DNS (by the peering agents of participating CDNs updating their DNS records regularly) and also via redirection at the CDN gateway (i.e. mediator, PA and policy repository as a single conceptual entity) when appropriate.

To deal with *Load dissemination issue*, the behavior of traffic can be modeled under expected peak load since in this case the server load is most severely tested. *Load information* can be measured and disseminated within individual CDNs and among other CDNs. A load index can provide a measure of utilization of a single resource on a computer system. Alternatively, it can be a combined measure of multiple resources like CPU load, memory utilization, disk paging, and active processes. Such load information needs to be disseminated among all participating CDNs in a timely and efficient manner to maximize its utility. Such indices will also be crucial to identify situations where forming a peering arrangement is appropriate (e.g. when servers or entire CDNs are overloaded) or when CDNs resources are underutilized and could be offered to other CDN providers. In this context, a hierarchical approach can be anticipated, where current bandwidth and resource usage of web servers in a CDN is reported to the CDN gateway in a periodic or threshold-based manner. The gateways of participating CDNs then communicate aggregated load information describing the load of their constituent servers.

*Content replication* occurs from origin servers to other servers within a CDN. Existing CDN providers (e.g. Akamai, Mirror Image) use a non-cooperative pull-based approach, where requests are directed (via DNS) to their closest replica server [10].

If the file requested is not held there, the replica server pulls the content from the origin server. Co-operative push-based techniques have been proposed that pushes content onto participating mirror servers using a greedy-global heuristic algorithm [6]. In this approach, requests are directed to the closest mirror server, or if there is no suitable mirror nearby, it is directed to the origin server. In the context of peering CDNs, this replication extends to participating servers from other CDNs in a given peering arrangement, subject to the available resources it contributes to the collaboration.

In summary, the following questions are to be addressed for distributing loads among peering CDNs:

- How to deduce a dynamic request assignment and redirection strategy that calculates ideal parameters for request-routing during runtime?
- How to ensure reduced server load, less bandwidth consumption (by particular CDN server) and improve the performance of content delivery?
- How do participating CDNs cooperate in replicating content in order to provide a satisfactory solution to all parties?
- What measures can be taken to ensure that the cached objects are not out-of-date? How to deal with uncacheable objects?

### *16.7.2 Coordination of CDNs*

Any solution to the above core technical issues of load distribution must be coordinated among all participants in a peering arrangement in order to provide high performance and QoS. A cooperative middleware must be developed to enable the correct execution of solutions developed to address each core issue. Related to this issue, the key question to be addressed is:

- What kind of coordination mechanisms need to be in place which ensure effectiveness, allow scalability and growth of peering CDNs?

### *16.7.3 Service and Policy Management*

Content management in peering CDNs should be highly motivated by the user preferences. Hence, a comprehensive model for managing the distributed content is crucial to avail end user preferences. To address this issue, content can be personalized to meet specific user's (or a group of users) preferences. Like Web personalization [14], user preferences can be automatically learned from content request and usage data by using data mining techniques. Data mining over CDN can exploit significant performance improvement through dealing with proper management of traffic, pricing and accounting/billing in CDNs. In this context, the following questions need to be addressed:

- How to make a value-added service into an infrastructure service that is accessible to the customers?
- What types of SLAs are to be negotiated among the participants? What policies can be generated to support SLA negotiation?
- How can autonomous policy negotiation happen in time to form a time-critical peering arrangement?

### *16.7.4 Pricing of Content and Services in CDNs*

A sustained resource sharing between participants in peering CDNs must ensure sufficient incentives exist for all parties. It requires the deployment of proper pricing, billing, and management systems. The key questions to be addressed in this context are:

- What mechanisms are to be used in this context for value expression (expression of content and service requirements and their valuation), value translation (translating requirements to content and service distribution) and value enforcement (mechanisms to enforce selection and distribution of different contents and services)?
- How do CDN providers achieve maximum profit in a competitive environment, yet maintain the equilibrium of supply and demand?

## 16.8 Conclusion

Present trends in content networks and content networking capabilities give rise to the interest for interconnecting CDNs. Finding ways for distinct CDNs to coordinate and cooperate with other content networks is necessary for better overall service. In this chapter, we present an approach for internetworking CDNs, which endeavors to balance a CDN's service requirements against the high cost of deploying customer dedicated and therefore over-provisioned resources. In our approach, scalability and resource sharing between CDNs is improved through peering, thus evolving past the current landscape where disparate CDNs exist. In this chapter, we also present two new models to promote CDN peering and identify the associated research challenges. Realizing the concept of CDN peering should be a timely contribution to the ongoing content networking trend.

**Acknowledgements** Some of the materials presented in this chapter appeared in a preliminary form at IEEE DSOnline [5], UPGRADE-CN'07 [17], and TCSC Doctoral Symposium—CCGrid'07 [18]. This work is supported in part by the Australian Research Council (ARC), through the discovery project grant and Department of Education, Science, and Training (DEST), through the International Science Linkage (ISL) grant. The material in this chapter greatly benefited from discussions with K. H. Kim and Kris Bubendorfer.

## References


1. Amini, L., Shaikh, A., and Schulzrinne, H. Effective peering for multi-provider content delivery services. In *Proc. of 23rd Annual IEEE Conference on Computer Communications (INFOCOM'04)*, pp. 850–861, 2004.
2. Arlitt, M. and Jin, T. Workload characterization of the 1998 world Cup Web site. *IEEE Network*, 14:30–37, 2000.
3. Assuncao, M., Buyya, R., and Venugopal, S. Intergrid: A case for internetworking islands of grids, *Concurrency and Computation: Practice and Experience (CCPE)*, Willey press, New York, USA, 2007.
4. Biliris, A., Cranor, C., Douglis, F., Rabinovich, M., Sibal, S., Spatscheck, O., and Sturm, W. CDN brokering. *Computer Communications*, 25(4), pp. 393–402, 2002.
5. Buyya, R., Pathan, M., Broberg, J., and Tari, Z. A case for peering of content delivery networks, *IEEE Distributed Systems Online*, 7(10), 2006.
6. Cardellini, V., Colajanni, M., and Yu, P. S. Efficient state estimators for load control policies in scalable Web server clusters. In *Proc. of the 22nd Annual International Computer Software and Applications Conference*, 1998.
7. Cardellini, V., Colajanni, M., and Yu, P. S. Request redirection algorithms for distributed Web systems. *IEEE Trans. on Parallel and Distributed Systems*, 14(4), 2003.
8. Colajanni, M., Yu, P. S., and Dias, D. M. Analysis of task assignment policies in scalable distributed Web-server systems. *IEEE Trans. on Parallel and Distributed Systems*, 9(6), 1998.
9. Day, M., Cain, B., Tomlinson, G., and Rzewski, P. A Model for Content Internetworking. IETF RFC 3466, 2003.
10. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing*, pp. 50–58, 2002.
11. Freedman, M. J., Freudenthal, E., and Mazières, D. Democratizing content publication with coral. In *Proc. of 1st Symposium on Networked Systems Design and Implementation*, San Francisco, CA, pp. 239–252, 2004.
12. Guo, L., Chen, S., Xiao, Z., and Zhang, X. Analysis of multimedia workloads with implications for internet streaming. In *Proc. 14th international Conference on World Wide Web (WWW)*, pp. 519–528, 2005.
13. Iyengar, A. K., Squillante, M. S., and Zhang, L. Analysis and characterization of large-scale Web server access patterns and performance. *World Wide Web*, 2(1–2), 1999.
14. Mobasher, B., Cooley, R., and Srivastava, J. Automatic personalization based on Web usage mining, *Communications of the ACM*, 43(8), pp. 142–151, 2000.
15. Padmanabhan, V. N. and Sripanidkulchai, K. The Case for Cooperative Networking. In *Proc. of International Peer-To-Peer Workshop (IPTPS02)*, 2002.
16. Pai, V. S., Wang, L., Park, K. S., Pang, R., and Peterson, L. The dark side of the Web: an open proxy's view. In *Proc. of the Second Workshop on Hot Topics in Networking (HotNets-II)*, Cambridge, MA, USA, 2003.
17. Pathan, M., Broberg, J., Bubendorfer, K., Kim, K. H., and Buyya, R. An architecture for virtual organization (VO)-based effective peering of content delivery networks, UPGRADE-CN'07, In *Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC 2007)*, Monterey, California, USA, 2007.
18. Pathan, M. and Buyya, R. Economy-based content replication for peering CDNs. TCSC Doctoral Symposium, In *Proc. of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, Brazil, 2007.
19. Pierre, G. and van Steen, M. Globule: A platform for self-replicating Web documents. In *Proc. of the 6th International Conference on Protocols for Multimedia Systems (PROMS'01)*, The Netherlands, pp. 1–11, 2001.
20. Pierre, G. and van Steen, M. Globule: a collaborative content delivery network. *IEEE Communications*, 44(8), 2006.
21. Turrini, E. An architecture for content distribution internetworking. Technical Report UBLCS-2004-2, University of Bologna, Italy, 2004.

22. Verma, D.C., Calo, S., and Amiri, K. Policy-based management of content distribution networks, *IEEE Network*, 16(2), pp. 34–39, 2002.
23. Wang, L., Park, K. S., Pang, R., Pai, V. S., and Peterson, L. Reliability and security in the CoDeeN content distribution network. In *Proc. of Usenix Annual Technical Conference*, Boston, MA, 2004.
24. Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S, Huynh, A., Carlson, M., Perry, J., and Waldbusser, S. Terminology for policy-based management, IETF RFC 3198, 2001.
25. Zhao, W. and Schulzrinne, H. DotSlash: A self-configuring and scalable rescue system for handling Web hotspots effectively. In *Proc. of the International Workshop on Web Caching and Content Distribution (WCW)*, Beijing, China, 2004.

# Index


Acquisitions, 15, 72
Adaptive request routing, 312
Akamai, 9, 17, 223, 257, 361, 391
Anycasting, 55, 57
Application-Level Multicast (ALM), 47, 84, 89, 90, 102, 137
Autonomous distribution, 380
Autonomous System (AS), 46, 156

Back-end, 106, 116
BackSlash, 282
Betweenness centrality, 356
BitTorrent, 13, 185, 283, 373, 408
Border Gateway Protocol (BGP), 17, 46, 61, 96
Broadband Services Forum (BSF), 10
Bursty traffic, 8, 212, 218
Byte hit ratio, 145

Cache Array Routing Protocol (CARP), 39, 40
Cache digest, 39, 41
Caching proxy, 8, 39
CAN, 87, 283
CDN peering, 55, 58, 390
Centralized directory model, 58
Centralized Directory Service (CDS), 86
Chord, 87, 200, 283
Client automaton, 305, 307
Client latency, 90
Client polling, 137
Client-side redirection, 157, 169, 176
CoDeeN, 22, 139, 357, 394
Code red attack, 279
Collaborative CDNs, 396
Collaborative Learning on-Demand (CLoD), 314
Consolidation, 15, 72, 270, 389
Content Access Point (CAP), 20
Content-Aware Caching (CAC), 117, 118, 136, 361
Content-Blind Caching (CBC), 117, 361
Content Distribution Internetworking (CDI), 393
Content Management and Online Reporting (CMOR), 19
Content negotiation, 4, 376
Content outsourcing, 4, 43, 47, 150, 348
Continuous Route Optimization Software (CROS), 18
Cooperative Association for Internet Data Analysis (CAIDA), 258
CoralCDN, 23, 66, 139, 282, 357, 394
Cumulative Distribution Function (CDF), 101, 145, 215, 334, 400

Data grid, 11
D-dimensional tori, 198
De Bruijn graphs, 197
Delivery support services, 381
Delta consistent, 137
On-demand update, 51, 66, 135, 142
Dispatcher, 157, 399
Distributed database, 13, 294
Distributed Denial of Service (DoS), 14, 60, 109, 130, 207, 277, 377
Distributed Hash Table (DHT), 23, 58, 87, 282
Distributed location services, 87, 88, 89
Distributed Routing and Location (DOLR) system, 80, 94, 102
Distributed Tutored Video Instruction (TVI), 314
DNS-based Request Routing, 55, 288
Document routing model, 58
DotSlash, 281
Download time, 8, 17, 43
Dynamic adaptation, 376
Dynamic cache partition, 133, 142
Dynamic content, 25, 41, 108, 134, 155, 214, 343, 361

Edge computing, 26, 110, 114, 136, 361
Edge server, 5, 109, 157, 393
Edge Side Includes (ESI), 18, 113, 135
EdgeStream, 18
Encoded media, 5
End-to-End (E2E), 379
Entire replication, 43
Extensible Rules Engine (XRE), 16, 20

First-mile bottleneck, 252
Flash crowds, 3, 4, 10, 14, 17, 95, 158, 211, 275
Flooded request model, 58
Free-riding, 184
Frequent network disconnections, 346
Front-end, 106, 111
Full compression, 143

Gateway, 379, 409
Geographical proximity, 60, 61
Global content caching, 16
Global Server Load Balancing (GSLB), 55
Globule, 24, 54, 139, 395
Gnutella, 33, 98, 184, 408

Head-Of-the-Line (HOL), 400
Heavy-tailed behavior, 277
Hierarchical caching, 9, 281
Hit ratio, 145
Horizontal handover, 372
Hosted server, 395
Hosting server, 395
Hotspot, 3, 23, 275
HTTP redirection, 56, 159, 292
HTTP streaming, 18, 107
Hypertext Caching Protocol (HTCP), 39, 41
Hypertext Transfer Protocol (HTTP), 5, 18, 40, 56, 107, 138, 159

IBM WebSphere, 26, 361
Incentives, 184, 212, 374
Infrastructure services, 381
Intelligent Domain Name Server (IDNS), 394
Inter-cluster caching, 49
Interleaved caching, 112
Internet Cache Protocol (ICP), 39, 40
Internet Congestion Tunnel Through (ICTT), 18
Internet Data Centers (IDCs), 80, 89, 283
Internet Engineering Task Force (IETF), 10, 393
Internet Service Providers (ISPs), 7, 47, 109, 129, 323
Internetworking, 389

Intra-cluster caching, 49
Invalidation, 51, 113, 137
IPTV, 16, 373

KaZaA, 33, 185

Layer 4–7 switch, 9
Limelight Networks, 19, 156
Live streaming, 254, 373
Live Webcasting, 16
Load dissemination, 409
Load index, 409

Materialized Query Table (MQT), 118
Mean response time, 145
Media Independent Handover (MIH), 372
Mergers, 15, 72
Metadata, 5, 322, 382
Middle-mile bottleneck, 252
Mirror Image, 20, 156
Mirroring, 71, 145, 159, 252
Misbehavior-sensitive networks, 380
Mobile Ad-Hoc Networks (MANETs), 353, 355
Mobile nodes constraints, 346
Multihoming, 71, 252, 372, 377
Multimedia content, 107, 140, 211, 320
Mutual consistency, 137

Naïve placement, 92, 94
Nash Equilibria, 192, 195
National Internet Measurement Infrastructure (NIMI), 258
Negotiation and Adaptation Core (NAC), 376
Network Element Control Protocol (NECP), 39
Network fragmentations, 346
Network Mobility (NEMO), 372
Network Operations Control Center (NOCC), 16
Network probing, 60, 257, 378
Network traffic, 3, 58, 128, 282, 328, 343, 383

Open Systems Interconnection (OSI), 5
Origin server, 5, 37, 109, 239
Ortiva Wireless, 28, 350
Overlay, 37, 87, 157, 185, 251

Page fragments, 107, 142
Partial compression, 143
Partial replication, 44, 117, 228
Password cracking, 279
Pastry, 87, 199, 283
Peer-to-Peer (P2P), 11, 139, 183, 214, 282, 340
Percentile-based pricing, 212, 221

Periodic update, 50, 135
Points of Presence (PoP), 47, 61, 109, 130, 391
Policy Decision Point (PDP), 398
Policy Enforcement Point (PEP), 398

Quality of Experience (QoE), 370
Quality of Service (QoS), 3, 8, 37, 61
Query response time, 171, 175

RaDaR, 84
Read-one write-all, 170
Real Time Performance Monitoring Service (RPMS), 18
Real Time Streaming Protocol (RTSP), 5
Reflectors, 255
Remote redirection ratio, 166, 171
Replica placement, 45, 81, 86, 127, 293, 320
Replica search, 90, 94
Replica server, 5, 25, 37, 94, 318, 409
Replicated Directory Service (RDS), 86
Request For Comments (RFCs), 10
Request redirection, 4, 69, 131, 155
Request response time, 164, 171, 177
Resilient Overlay Networks (RON), 258
Reverse proxy, 22, 159, 286
Routing mesh, 87

Search-Insertion-Deletion-Update (SIDU) operations, 140, 144
Segment caching, 112
Self-provisioning, 215, 219
Sequential caching, 112
Server automaton, 305, 306
Server farm, 9, 252
Server load, 43, 63, 90, 135, 281
Server migration, 158, 177
Server placement, 45, 47, 226
Server Sharing, 158
Server-side redirection, 157, 159, 168
Service interaction, 375
Service Level Agreement (SLA), 89, 381, 389
Service Oriented Architecture (SOA), 26, 375
SlashDot, 3, 4, 65, 275
Small to Medium Enterprise (SME), 8
Smart placement, 92, 94
Social optima, 192, 196, 201
Static adaptation, 376
Static cache partition, 132, 142
Static content, 7, 18, 41, 107, 132, 156, 281
Streaming media, 42, 61, 83, 88, 361
Strongly consistent, 137
Surrogate, 5, 43, 45, 87, 110, 130, 286, 345, 357
Surrogate placement, 36, 45
SYN attack, 279
Synthetic workload, 95

Tapestry, 80, 87, 199
Time-To-Live (TTL), 50, 112, 136, 289
Total ordering, 170
Traffic monitoring, 60, 354
Tutored Video Instruction (TVI), 314

Uniform Resource Locator (URL), 5, 9, 55, 157, 293
Unpredictable-update, 135
Update propagation, 51, 119
URL rewriting, 57, 66, 85, 157, 292
User Generated Videos (UGV), 7

Vehicular Ad-Hoc Networks (VANETs), 353, 355
Vehicular Ad-Hoc Server (VAHS), 356
Vehicular Information Transfer Protocol (VITP), 355
Vertical handover, 372
Video-on-Demand (VoD), 10, 380
Virtual Private Networks (VPN), 161, 255, 270
Voice over Internet Protocol (VoIP), 255, 270, 377

Weak consistent, 137
Web Cache Control Protocol (WCCP), 39, 40
Web Cache Invalidation Protocol (WCIP), 137
Web cluster, 5, 71, 158
Web Content Distribution Protocol (WCDP), 137
Web traces, 95, 100, 278, 357
Wireless Mesh Networks (WMN), 372
Wireless Sensor Networks (WSN), 372

Zipf distribution, 202, 279