

Rajkumar Buyya
Mukaddim Pathan
Athena Vakali *Editors*

Content Delivery Networks

 Springer

META-1012
IPR2026-00346, IPR2026-00347

Lecture Notes Electrical Engineering

Volume 9

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

Content Delivery Networks

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

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
	Mirośław Czyrnek, Ewa Kuśmerek, 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

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

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

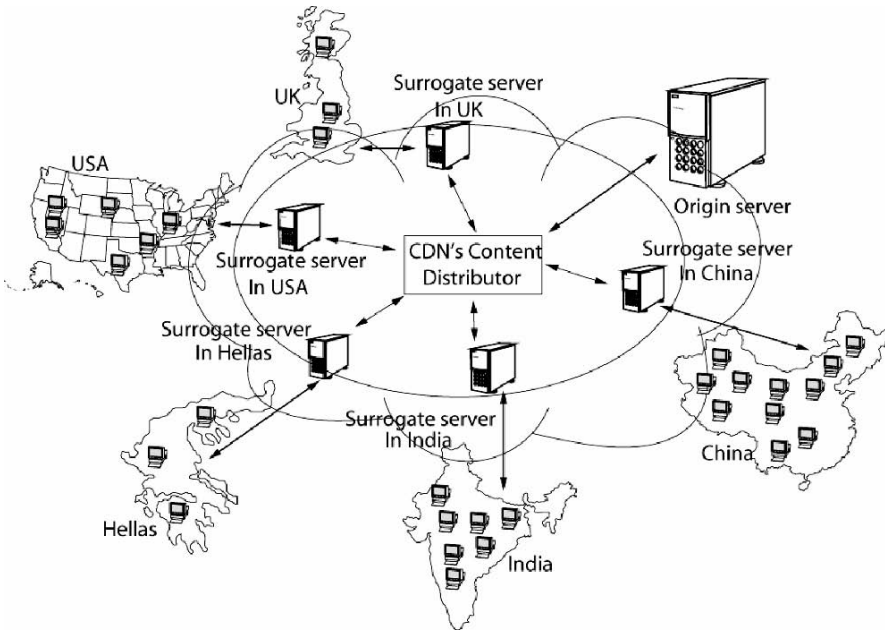


Fig. 1.1 Model of a CDN

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

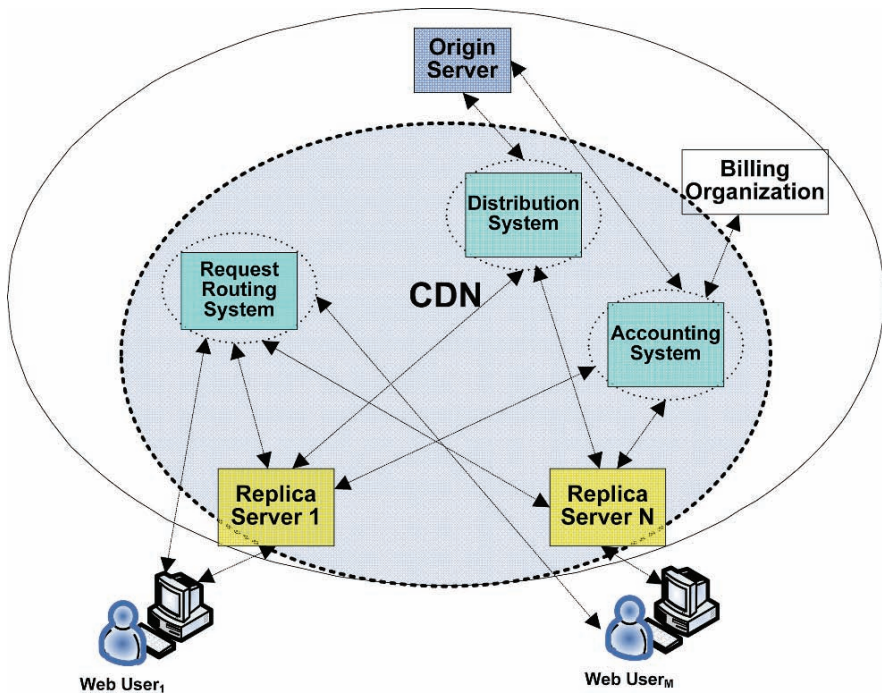


Fig. 1.2 Architectural components of a CDN

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

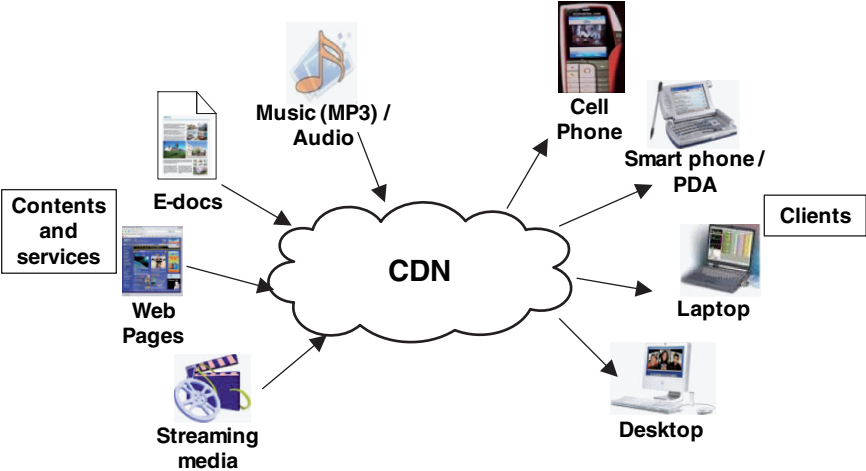


Fig. 1.3 Content/services provided by a CDN

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.

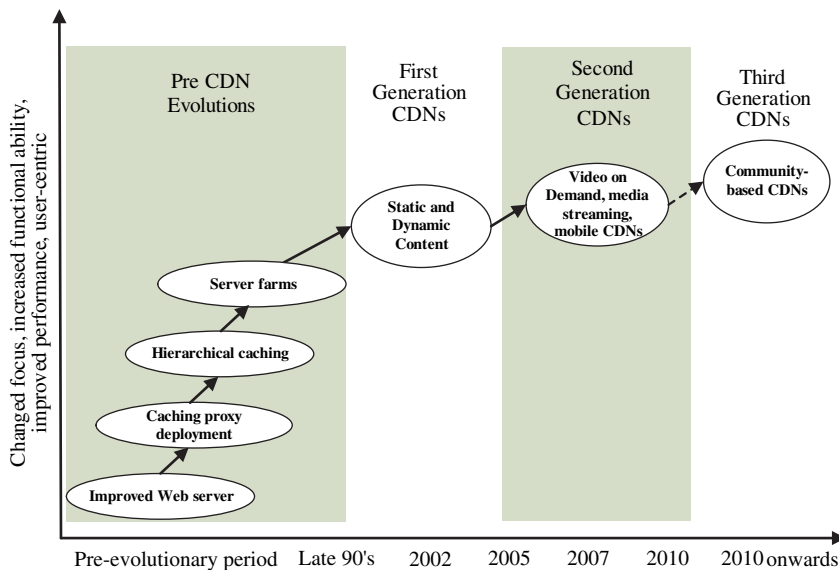


Fig. 1.4 CDN evolutions

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

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

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:

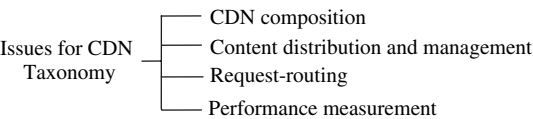
- A taxonomy of CDNs with a complete coverage of this field to provide a comprehensive account of applications, features, and implementation techniques. The main aim of the taxonomy, therefore, is to explore the functional and non-functional features of CDNs and to provide a basis for categorizing the 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.

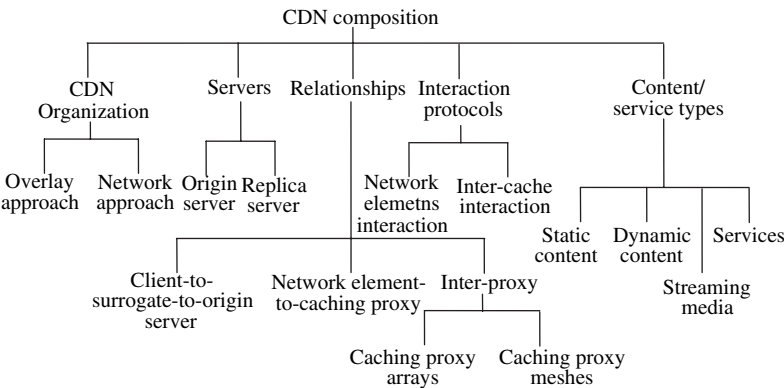


Fig. 2.2 CDN composition taxonomy

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

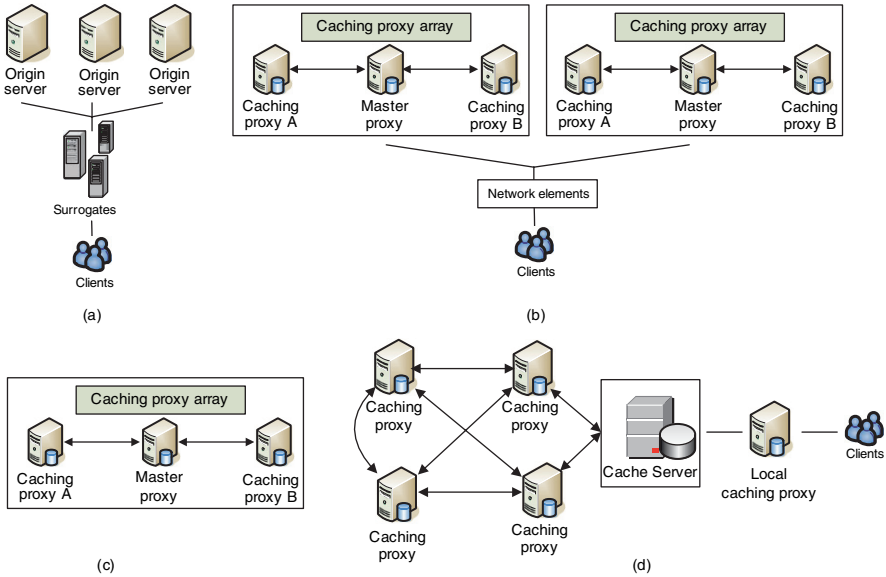


Fig. 2.3 Relationships: (a) Client-to-surrogate-to-origin server; (b) Network element-to-caching proxy; (c) Caching proxy arrays; (d) Caching proxy meshes

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

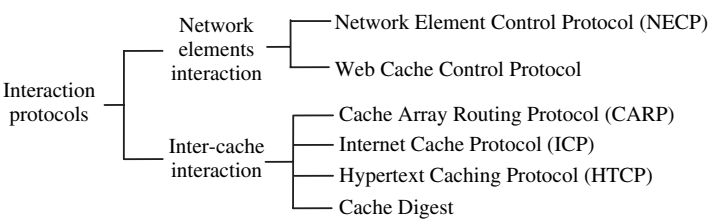


Fig. 2.4 Interaction protocols

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 connection-less 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.

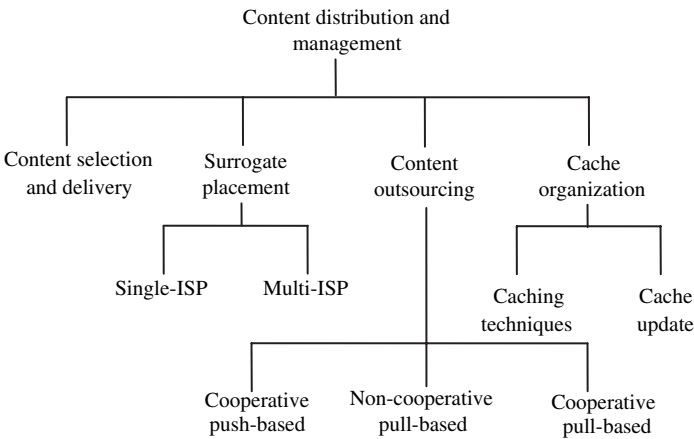


Fig. 2.5 Content distribution and management taxonomy

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

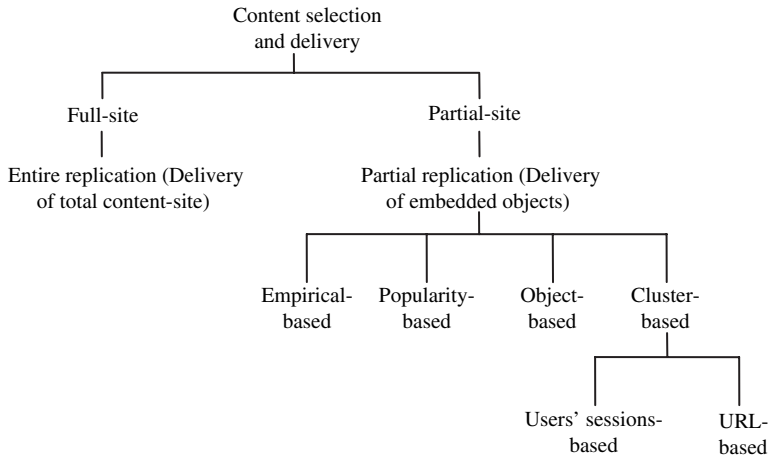


Fig. 2.6 Taxonomy of content selection and delivery

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

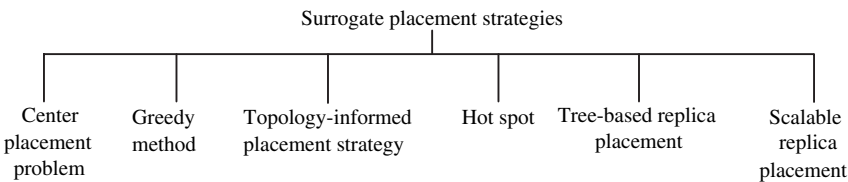


Fig. 2.7 Surrogate placement strategies

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 $c : c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$, construct a set of square graphs $G^2_1, G^2_2, \dots, G^2_m$. 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 cache 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

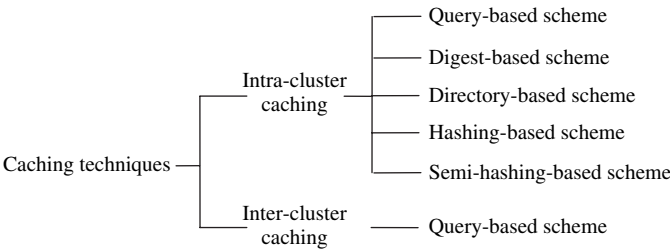


Fig. 2.8 Caching techniques taxonomy

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

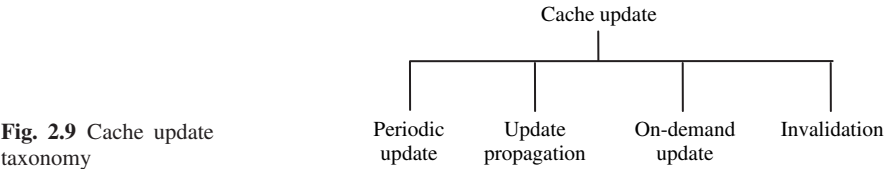


Fig. 2.9 Cache update taxonomy

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