

◆ Intelligent Delivery of Interactive Advertisement Content

Peng Zhang, Juntao Ma, and Xiaorong Sun

Content networking provides a potential solution for advertising applications. Bell Labs Research China (BLRC) is conducting an interactive multimedia advertising project, which aims to create an advertising system that will enhance the interactivity between customers and advertisers. Rich communication methods including Voice over Internet Protocol (VoIP), short message service (SMS), multimedia message service (MMS), and email will be introduced that will allow customers to set up interactions with advertisers with just a few easy mouse clicks. Moreover, the system provides an extensible infrastructure for dealing with a variety of advertisement media types, such as text, image, voice, flash, and video. In order to promote the delivery efficiency of multimedia advertising content, we borrow some concepts from the content delivery network (CDN). This system consists of one advertisement description server (ADS) and several advertisement media servers (AMS). The former stores the descriptive data of all advertisements and is responsible for delivering description information to user-side applications in a specific format. The latter stores the actual multimedia content of advertisements and works in a load-sharing manner to deliver the media content. The advertisement description server also plays the role of "request dispatcher" to receive advertisement requests from users and forward them to different advertisement media servers. In addition, some statistics-based strategies are deployed to make the advertisement description server more "intelligent" when delivering advertisements. Here, multiple statistical parameters including advertisement impressions, click-through rate, and fees paid by advertisers operate in tandem to determine the selection of advertisements to be delivered. This advertising solution also has been adopted to provide advertisement support for two other projects inside BLRC, MapWeb and the IP Multimedia Subsystem (IMS) personal computer (PC) client. © 2008 Alcatel-Lucent.

Introduction

Advertising over the Internet has gained great momentum in recent years. For example, Google's* advertising products, flagshipped by AdWords [2], which brought \$10.492 billion to the company in

fiscal 2006, are the main source of the company's revenue. Yahoo!'s* ad solutions, including Yahoo! Search Marketing* and the Yahoo! Publisher Network* [7], provide support for a so-called online contextual

Bell Labs Technical Journal 13(3), 143–158 (2008) © 2008 Alcatel-Lucent. • DOI: 10.1002/bltj.20330

Alcatel-Lucent 

Panel 1. Abbreviations, Acronyms, and Terms

ad—Advertisement	ODBC—Open database connectivity
ADS—Advertisement description server	PC—Personal computer
AdsDB—Advertisement database	PDA—Personal digital assistant
AdsDE—Advertisement delivery engine	PPC—Pay per click
AdsMan—Advertisement management Web portal	QoS—Quality of service
AdsPublisher—Advertisement publisher Web portal	ROI—Return on investment
AdsWS—Advertisement Web service module	RPC—Remote procedure call
AMS—Advertisement media server	RTSP—Real Time Streaming Protocol
BLRC—Bell Labs Research China	SMS—Short message service
CDN—Content delivery network	SOAP—Simple Object Access Protocol
CTR—Click-through rate	SQL—Structured query language
FTP—File Transfer Protocol	TV—Television
HTML—Hypertext Markup Language	UDDI—Universal description, discovery, and integration
HTTP—Hypertext Transfer Protocol	URL—Uniform resource locator
ID—Identifier	VoIP—Voice over Internet Protocol
IMS—IP Multimedia Subsystem	WAP—Wireless Application Protocol
IP—Internet Protocol	WS—Web service
IPTV—Internet Protocol television	WsBroker—Web service broker
JDBC—Java database connectivity	WSDL—Web Services Description Language
JSP—JavaServer Pages	WWW—World Wide Web
MMS—Multimedia message service	XML—Extensible markup language

advertisement service. Following the lead of Google and Yahoo!, Microsoft also developed its own advertising system, adCenter [3], which was launched in May 2006.

Compared with traditional media such as television (TV) and newspapers, the Internet and World Wide Web (WWW) provide a more rapid channel for advertising, which largely decreases the cost of publishing and updating advertisements. In addition, the media types supported by the Internet have become far more diversified than before. Providing support for multiple media types is a prevailing trend for Internet-based advertising systems, and we expect that inevitably it will become a basic feature for future advertisement servers.

The goal of our system is to provide customers with interactive advertisements for customers that may contain content of multiple media types such as text, image, flash, video, or mixture of these formats. Since different media types always require different

processes, it is inefficient to ignore the different advertisement formats and store all advertisements in one repository.

Content networking provides a potential solution for our advertising system. Content delivery networks (CDNs) have evolved to overcome the inherent limitations of the Internet in terms of user-perceived quality of service (QoS) when accessing Web content [4]. A CDN replicates content from the originating server to surrogates (or cache servers) scattered over the global network in order to deliver content to end users in a reliable and timely manner from nearby optimal surrogates.

We borrow some concepts from CDN to improve the performance of our advertising system without upgrading existing network infrastructure. In the context of CDNs, content generally refers to any digital data resource and consists of two main parts: the encoded media and metadata [5]. In the same way, the content of an advertisement is also composed of

real encoded media and advertisement metadata. Some examples of advertisement media are static/dynamic images, video/audio clips, and continuous stream media data. Metadata provides a description of an advertisement that includes the properties of the encoded media, such as the media format, length, date of generation, and storage location. Metadata can be seen as a kind of resource to facilitate the interpretation of all advertisements, which provides a way to identify, access, and manage all the advertisements in our advertising system.

Content redundancy is introduced into CDN for improving performance through caching or replicating content over some mirrored Web servers (i.e., surrogate servers), strategically placed at various locations in order to deal with sudden traffic bursts of Web content requests. Multiple servers are also used in our advertising system to keep redundant advertisement media content geographically distributed, and to preclude the prospect of concentrating a large amount of advertisement requests from users in a single server. These servers can be divided into two types: the advertisement description server (ADS) and the advertisement media server (AMS). The former stores metadata of advertisements and delivers the descriptive information to the request users, while the latter delivers the actual ad content (media data) to users. Although the two types of servers have different functionalities, they interact and work in a cooperative manner to deal with advertisement requests from users. The hierarchy of servers and their interactions will be discussed in more detail in the sections following.

To begin, we need to interpret two terms, “interactive advertisement” and “intelligence,” as they apply to our system. An “interactive advertisement” in our system does not refer to an advertisement that contains interactive media content and has the innate ability to interact with consumers. Rather, we mean that there is a functional extension to the advertisement by the aid of existing communication technologies, such as Voice over Internet Protocol (VoIP), short message service (SMS), and email, which allows consumers to participate actively in the interactions with the advertisement content. The information neces-

sary for this interactive extension is generated by the advertisement’s metadata. For example, a phone number could be extracted from metadata to create a Web link which can be activated by a simple mouse click to set up a voice call with a vendor through a Web page.

“Intelligence” is an overused word full of meaning in a variety of definitions. Indeed, when 100 people are asked to define intelligence, they may provide 100 somewhat different definitions. In this paper, the word is used to describe a “smart” advertisement delivery mechanism, i.e., one with the ability to:

1. Choose a media server nearest to a specific client in network distance for delivering advertisement media data,
2. Select appropriate items from a large collection of advertisements and publish in appropriate regions on a client according to the predefined categories, and
3. Adapt to dynamic changes in some runtime parameters (e.g., ad impressions), or adjust system behavior dynamically for the purpose of improving the efficiency.

We now present a business model and an introduction to the system architecture. Later sections will focus on how advertisement content is stored and distributed among servers. The interactions among different servers will be classified and described in more detail. We will then discuss the strategy and algorithms used for advertisement selection. Finally, we present our conclusion.

Business Model

Figure 1 shows the business model of our advertising system, which illustrates how the system interacts with users or other systems when it is deployed. There are four roles involved in the business model: advertiser, ad operator, ad publisher, and ad users. The arrowed links in the figure represent the relationships among roles. Roles and their relationships are two primary elements in the business model.

The *ad operator* is a company or organization tasked with managing and hosting ad servers and operating advertisement-related services. Ad servers

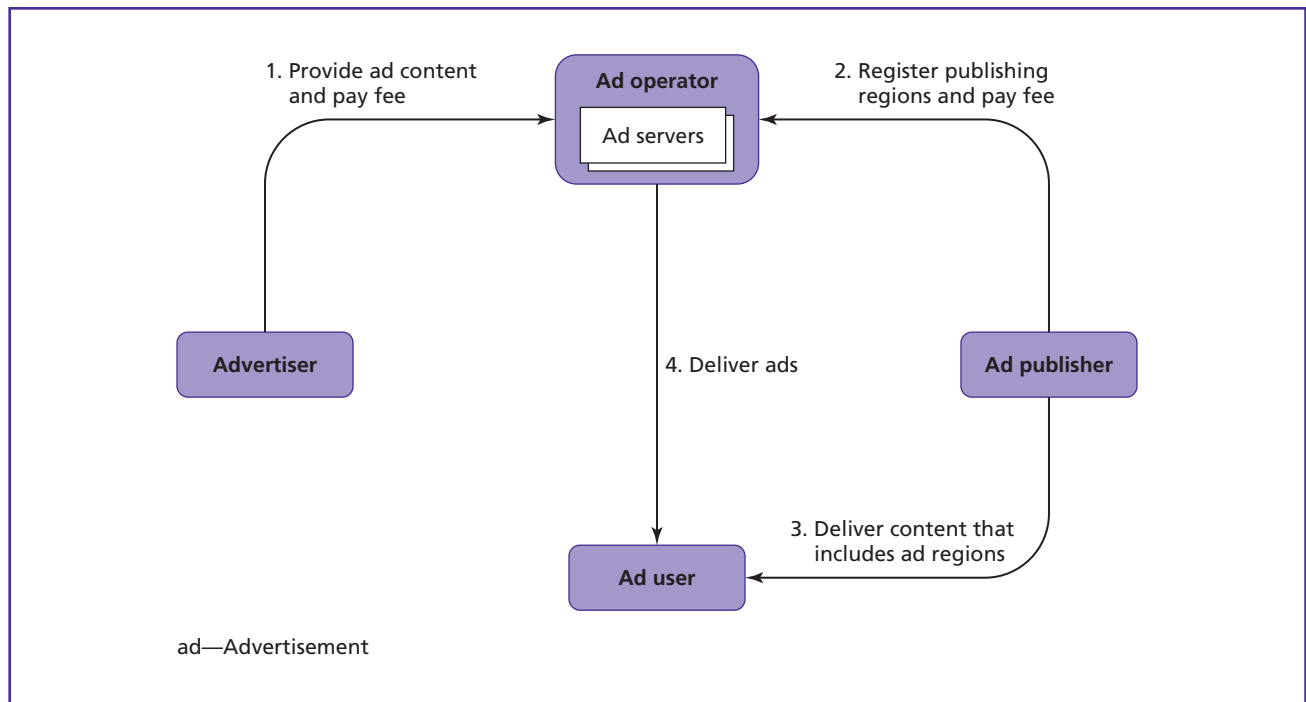


Figure 1.
Business model of the advertising system.

form the major part of our advertising system. They serve as the advertisement repositories and deliver advertisements to users on request. In addition, ad servers provide the interface necessary to support advertising management/maintenance operations.

The *advertiser* is an ad provider, responsible for creating the content of one or more advertisements and submitting it the ad operator for publishing. The advertiser may need to pay a sum of money to the ad operator for using the advertising service. The advertiser may also manage his own advertisements via a management interface supported by ad servers.

The *ad publisher* provides the channel that carries the advertising content to the public. Channels used to publish advertisements may include high-traffic Web pages, widely used Internet applications, or Internet Protocol television (IPTV). The cooperation between ad operator and ad publisher is beneficial for both sides. From an ad operator's perspective, the relationship helps to increase the popularity of advertisements since more ad publishers translate to more connections to advertisement users. On the other hand, the relationship also provides an additional way

for an ad publisher to earn extra profits from the ad operator. Our system introduces a new concept called "region" to describe the space that is part of the public resource provided by an ad publisher to house advertisement content. Actually, a region is a block of area in which an advertisement could be displayed to users. Like the advertiser, an ad publisher can also log on to an ad server to register and manage his own region information through the management interface supported by ad servers.

The final player in the business model is the *ad user*, or the recipient of advertisements. The ad users in our advertising system have the ability to interact with the ad provider through phone calls, SMS/multimedia message service (MMS), and email.

The role entities in the business model interact as follows:

1. An advertiser produces an advertisement and submits it to the servers owned by an ad operator.
2. An ad publisher registers the regions available for publishing advertisements and is paid according to a predefined payment policy such as pay-per-click (PPC).

3. A user sends out a request to the ad publisher for a specific public resource (e.g., a Web page). The ad publisher responds with the requested content in which the ad regions are included.
4. The ad server delivers the advertisement to the user. The advertisement will be placed in the regions and displayed.

System Architecture

As mentioned earlier, our advertising system consists of two types of servers—ADS and AMS. **Figure 2** illustrates how the two types of servers are structured. There is only one ADS server, which stores, manages, and delivers the advertisement descriptive information (metadata), while there are numerous AMS servers, which serve as storage repositories for the replicated media data that will be delivered to users upon request. The end users can interact with the server cluster by specifying their advertisement request through a cell phone, smart phone/personal

digital assistant (PDA), computer, Internet Protocol (IP) phone, or other device. In the implementation of our advertising system, we provide a convenient and cost-effective means to operate AMS servers by using existing mature techniques, e.g., Hypertext Transfer Protocol (HTTP) servers, File Transfer Protocol (FTP) servers, or Real Time Streaming Protocol (RTSP) streaming media servers. Therefore, the discussion in this section is focused on the ADS server only.

The ADS server contains five modules: the advertisement delivery engine (AdsDE), advertisement management Web portal (AdsMan), advertisement publisher Web portal (AdsPublisher), advertisement Web service module (AdsWS), and advertisement database (AdsDB). The main functionalities of the five modules are described in the following.

Advertisement Database

The advertisement database (AdsDB) provides data storage for the ADS server. The implementation

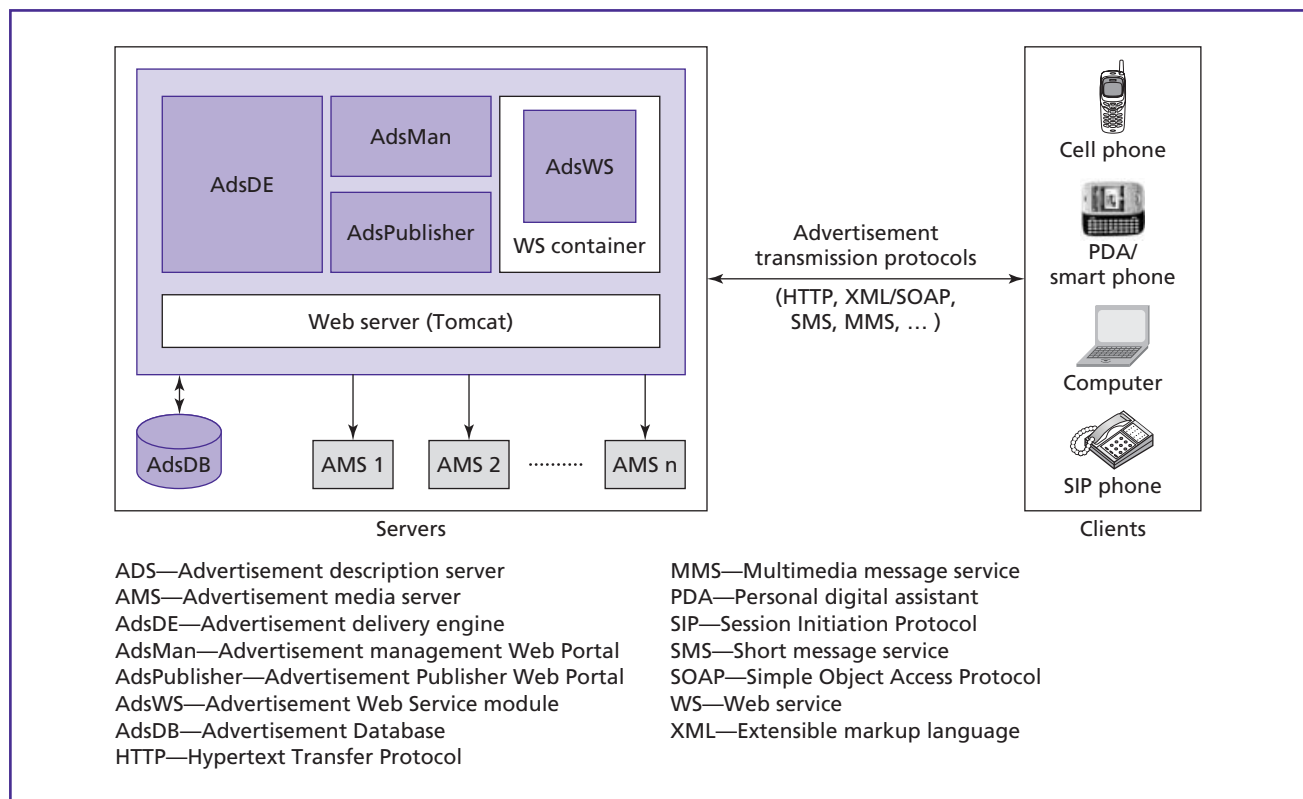


Figure 2.
Architecture of advertisement delivery system.

of the AdsDB module is based on MySQL*—the AdsDB appears as a database running on the MySQL server. Although the AdsDB (plus the MySQL server) is a standalone module and does not rely on the existence of the ADS server, we prefer to consider it as a part of the ADS server because only the ADS server has privileges to access this module through standard structured query language (SQL) statements/scripts. In the advertising system, the AdsDB is designated as the repository for the description/metadata of all advertisements, including title, author, media format, size, language, and date of generation. For each advertisement, there exists a collection of uniform resource locators (URLs) maintained in the AdsDB, which point to a group of servers and the absolute paths where replicas of the advertisement media content are stored. Using the media URL, a user can send a request to a specific AMS server to obtain the media content. Besides metadata, the AdsDB module stores additional information such as end user data as well as the access log. The operations related to the data in AdsDB could be performed through standard methods such as SQL and open database connectivity (ODBC).

Advertisement Delivery Engine

The advertisement delivery engine (AdsDE) is the critical module in the system. It receives advertisement requests from end users, parses user requirements, selects suitable advertisements, and delivers them to advertisement users. In our implementation, the AdsDE module works as a Web application which runs over a Tomcat Web server. The AdsDE needs to access the AdsDB module to obtain advertisement metadata. The database connection between the AdsDE module and the AdsDB is built and maintained using Java* database connectivity (JDBC) technology.

As shown in **Figure 3**, the AdsDE contains four functional submodules/components: fetcher, forward, WsBroker, and strategy. The fetcher component provides an access interface for external users or client applications, which receives the requests from client users and sends the expected advertisements. The forward component is tasked with capturing user requests for advertisement media data and redirecting them to a specific media server for further processing. The Web service broker (WsBroker) can invoke

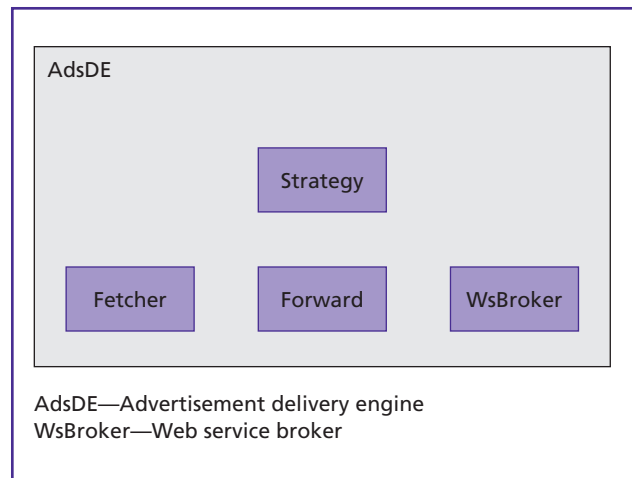


Figure 3.
Components of AdsDE.

Web services from third-party providers, e.g., set up a voice call connection or send out an SMS/MMS message. The WsBroker enables users to interact with advertisements. For example, if a user is interested in an advertisement, he/she can click on the advertisement display window to initiate a phone call with the advertiser.

All three components—fetcher, forward, and WsBroker—support the logging function, which permits each event about advertisement delivery to be recorded. The logged information is used for further offline processing such as data analysis and statistics. The strategy component performs most of the statistical tasks. Moreover, the strategy component is also responsible for providing a group of optimized parameters so that the fetcher component can act intelligently to “deliver the right advertisement to the right person,” for the purpose of increasing ad traffic and generating more profits. A statistics-based approach is used in this system to enable the delivery process to operate more “intelligently.”

Advertisement Management

The advertisement management (AdsMan) module is a Web portal which enables advertisers to manage and maintain their advertisements remotely through common Web browsers. With the support of this Web portal, advertisers can log on to the system, create new advertisements, modify existing advertisements, or delete obsolete advertisements. In addition,

advertisers can view an overall evaluation report about their own advertisements from the Web portal, including performance parameters such as total clicks, advertisement impression ratio, and return on investment (ROI). AdsMan is implemented as a collection of JavaServer Pages* (JSP) plus Java classes supported by the Struts framework.

Advertisement Publisher

The advertisement publisher (AdsPublisher) is another Web portal module, which enables advertisement publishers to maintain their advertisement publishing regions. A publishing region is a block of area provided by an ad publisher, in which advertisements can be loaded and viewed. Ad publishers can execute some region-specific operations—such as region creation, modification, and deletion—with the aid of the AdsPublisher portal.

Advertisement Web Service

“Web service” has become a very popular term in the world of Internet and telecom. Described as a standardized way of integrating Web-based applications, Web service leverages open standards such as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI) over an Internet Protocol backbone. In order to facilitate the integration of advertisements with other third-party applications, Web services are also supported by our advertising system through the advertisement Web service (AdsWS) module. The AdsWS module provides a Web service called “AdsService,” and other third-party applications can communicate with this module to use the service through SOAP. AXIS, an implementation of SOAP from Apache, works as a Web service (WS) container in the ADS server, which provides the execution environment for the AdsWS module.

Distribution of Advertisement Content

Our advertising system is designed to support most media types, including text, image, and video/audio clips. As mentioned previously, the content of each advertisement can be divided into two portions—metadata and media data.

Metadata and Media Data

The content of an advertisement in our advertising system is stored at different locations in the network. The ADS server and AMS servers provide separate places for storing/delivering metadata and media data.

Separate storage and delivery can generate certain benefits. The metadata consist of simple textual/binary descriptions so very little time is required for processing. However, the media data are always complex and of significant length, making media processing a very time-consuming task. In some cases, notably video advertisements, the media processing work must continue throughout the advertisement’s display. From the viewpoint of functionality, it would be better to distribute advertisement request processing across specialized servers, in order to prevent the accumulation of requests at a single service point. Moreover, distribution would be helpful for improving system performance, since multiple media servers can share the processing workload. In addition, the system could be constructed economically by using existing mature media servers, such as HTTP or RTSP servers.

Hierarchy of Servers

The servers used by the advertising system are composed of a single ADS server and several AMS servers, organized hierarchically as shown in **Figure 4**. The collaboration between the ADS server and the AMS servers is very similar to that of a master/slave relationship, where the AMS servers act as “workers” to deliver media content and the ADS server plays the role of “coordinator” to assign delivery tasks to various media servers. The cooperation between the two kinds of servers includes three phases:

1. First, the client sends a request for an advertisement to the ADS server by specifying some parameters, such as media type, language, category of interest, or display region.
2. When the ADS server receives the request, it extracts the request parameters and searches the internal AdsDB module to locate an appropriate advertisement. If the expected advertisement contains media data, the request will be redirected to an AMS server.

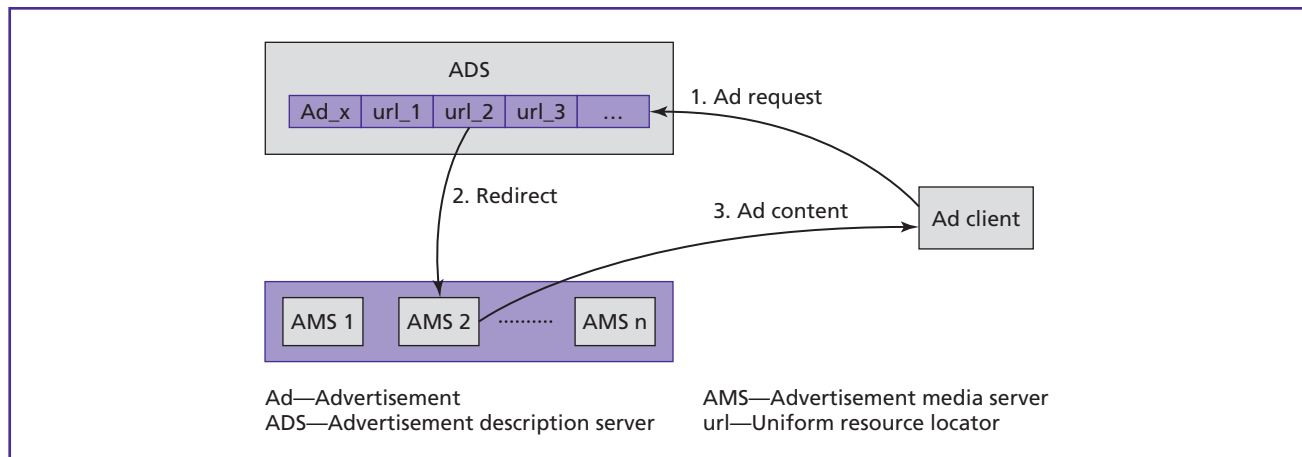


Figure 4.
ADS server and AMS servers.

3. Upon receiving the advertisement media request, the AMS server transmits the media content back to the client side.

As described previously, there is a database module—the AdsDB—inside the ADS server that stores the description/metadata for all advertisements. For each advertisement, a list of URLs is included in the metadata, which refers to all the locations of the media data replicas. A URL defines the server address, absolute file path, and communication protocol, which are necessary to access the media of a specific advertisement. For example, url_2 in Figure 4 corresponds to one media content replica of an advertisement, ad_x, stored on AMS server 2. By clicking on “url_2,” a user can send a request to the media server for the media content and view it on the client host.

Selection of URL

As shown in Figure 4, the ADS server makes a decision regarding the selection of the URL to be used when redirecting a media request. An appropriate URL selection approach assists in reducing the media download time and balancing the server’s workload. The URL selection actually is the determination of the media server since each URL corresponds to one specific media server. The system adopts a “shortest-in-distance” strategy which is designed to deliver advertisement media data from the media server closest to the client side.

The distance between two network elements could be measured in terms of network proximity, or the

matched length of the two IP addresses in binary format. The calculation of matched length is from higher bit to lower bit. For example, the binary form of IP address “211.68.92.238” is “11010011 01000100 01011100 11101110,” and that of “211.68.85.135” is “11010011 01000100 01010101 10000111.” The matched portion between the two addresses is “11010011 01000100 0101,” and the length (or network proximity) is 20. The value of network proximity describes the distance between two hosts with different IP addresses, and the scope is from 0 to 32. Generally speaking, a smaller network distance (or larger network proximity) always means two network elements are adjacent to each other, and this knowledge may be used to shorten the download time for advertisement media and improve the system throughput.

Following this method, when receiving a request, the ADS server can extract the IP address of the client host. Then the ADS server searches the database to select the advertisement that will be delivered to the client. If the advertisement has more than one media location, i.e., URL, the ADS server will calculate the distances of each corresponding media server to the client host. After that, the media server that is nearest to the client will be selected and the request will be redirected.

Except for the “shortest-in-distance” strategy, there are many other useful selection methods when more factors are taken into consideration, such as the

load-balance among media servers, optimization of client-perceived latency, minimized delivery cost, and so on. But the network distance-based method is much simpler to implement because of its low complexity.

Interactions and Scenarios

The network elements in the advertising system need to exchange messages with each other to perform interaction activities. The interactions include client-server interaction and interserver interaction. The client-server interaction occurs when a user sends an advertisement request to a server or a server responds with a delivery of advertisement metadata or media data. The interserver interaction takes place between the ADS server and the media servers for the purpose of performing collaborative tasks and managing advertisement content distributed at different server nodes.

Interactions among network elements must be based on a certain communication protocol. At present, the network elements in the advertising system employ two widely used protocols, HTTP and SOAP, to carry the interactive messages between them. The HTTP-based interaction is used mainly for the third-party Internet applications and websites, while the SOAP-based interaction is used for the Web service invocations from advertisement users. The following section describes the two kinds of interactions in more detail.

HTTP-Based Interaction

Hypertext Transfer Protocol [1] works in a request/response manner between a client and a server. Generally, the client initiates an HTTP request and the server replies with a response which may contain the requested resource content such as Hypertext Markup Language (HTML) files or images, error information, or other information. HTTP is widely used across the World Wide Web to transfer information.

HTTP defines eight methods on specified network resources, such as HEAD, GET, POST, and PUT [1]. In our advertising system, the GET method is used to make an advertisement request and the response contains either the advertisement metadata or a redirection notification to a media server.

The definition of redirection notification is also based on the HTTP protocol. The first line of an HTTP response is called the status line, which includes a numeric status code and a textual reason description. If the status code is “302,” it indicates that the requested resource could not be provided and the request should be redirected to another server. Upon receiving this response, the client needs to resend the request in a subsequent HTTP message to the destination as defined in the “location” field of the response. In this way, the ADS server can conveniently redirect the request for advertisement media data to a media server by setting the “location” field in its response to a specific URL.

A number of scenarios of HTTP-based interactions are detailed below:

Typical HTTP interactions at advertisement delivery. **Figure 5** shows a typical scenario of HTTP-based interactions that usually occur when a client wishes to obtain advertisement information from our advertising servers. At first, the client sends out an advertisement request via the HTTP GET method, with parameters specified (such as the client user’s name, and the number of advertisements requested, as shown in the “request” message format in **Figure 6**). Upon receiving the request, the ADS server will return a description of one advertisement that meets the requirements, contained in the body of the “200 OK” response message. As shown in **Figure 6**, the reply description/metadata are structured in an XML-style format and the media data URL is included inside the “src” tag, i.e., “http://135.252.6.54:8080/ads/pics/ads_pic_1.jpg”. The client extracts the URL information by parsing the embedded XML textual content.

With the obtained media data URL, the client sends a new request to start a second round of interaction. The request will be transferred to a media server as specified by the URL. Then the media data will be transferred back to the client, carried in the payload of the “200 OK” response message.

As shown in this scenario, a client completes two steps to obtain an advertisement—the first step is to acquire metadata, and the second step is to make use of the information obtained to acquire the media data.

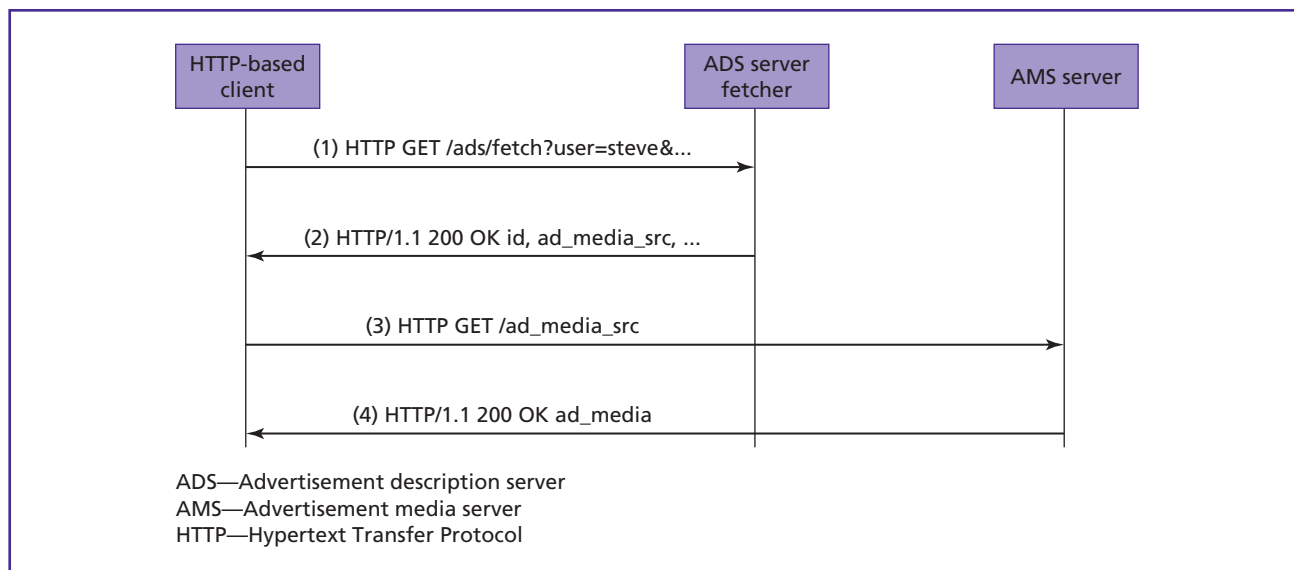


Figure 5.
General HTTP message interactions on advertising.

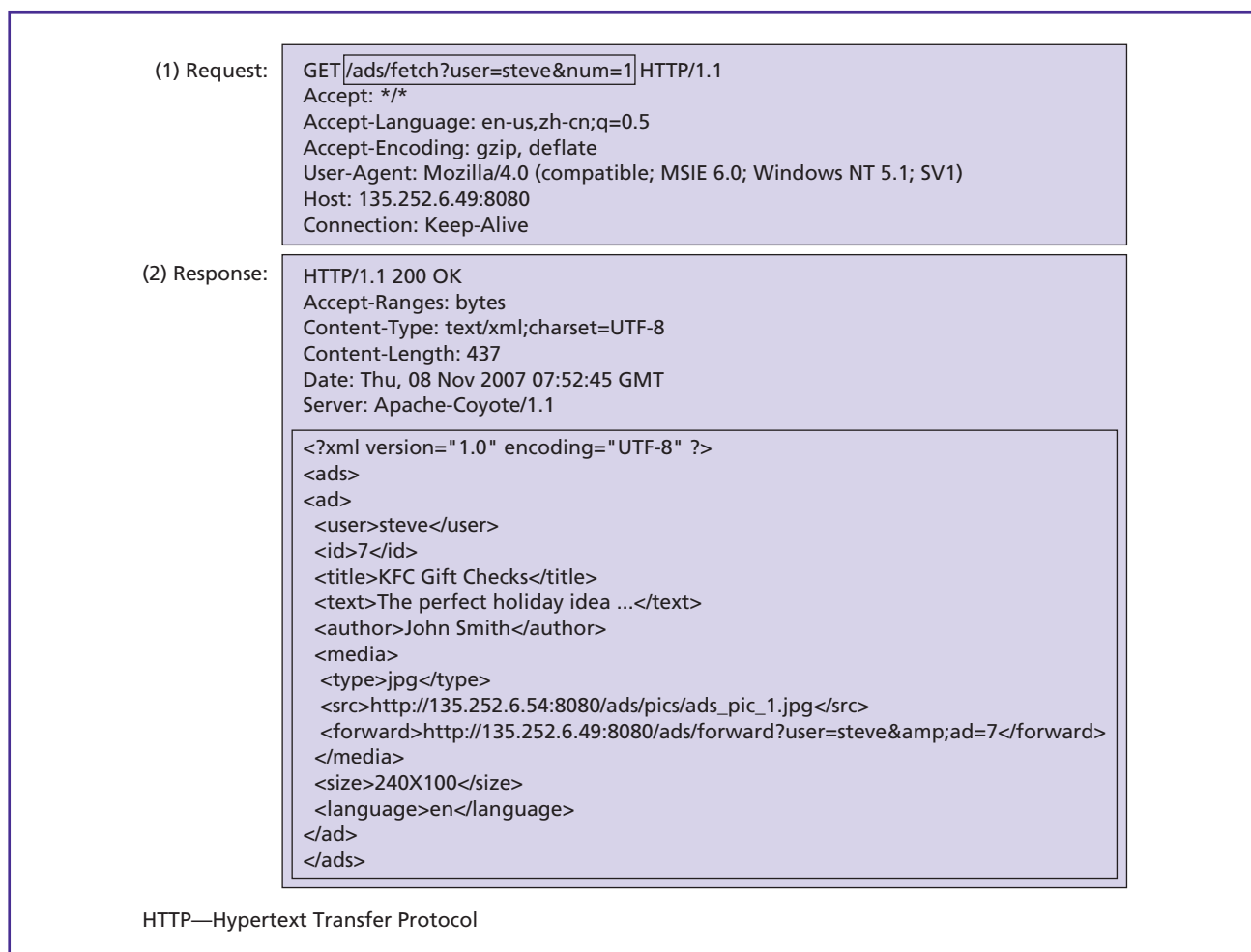


Figure 6.
HTTP messages for an advertisement request and response.

Redirection of advertisement media requests. In the advertising system, each advertisement has a unique identifier (ID) number, by which the advertisement is solely identified. A client may send a request for advertisement media data to the ADS server, with the ID specified. In such a case, the ADS server is required to redirect the user's request to an appropriate media server for delivering media data.

Figure 7 shows an example of a redirection scenario. An HTTP GET request is sent to the ADS server in which the expected advertisement ID is defined. As depicted in **Figure 8**, the advertisement ID is specified in the expression "ad = 7." When the request arrives at the ADS server, a response message will be created, which includes a status code "302" and a "location" field. The information is used to notify the client that the request for the advertisement media data should be redirected to another URL. After receiving this response, the client will initiate a request with the new URL to obtain the media.

SOAP-Based Interaction

Simple Object Access Protocol is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment which uses XML technologies to define an extensible messaging framework over a variety of underlying

protocols [6]. SOAP provides a convenient mechanism for supporting remote procedure calls (RPCs) or remote object access across different operating systems, with different technologies and programming languages.

As mentioned previously, a special AdsWS module in our system is used to provide the Web service interface for third-party application access to the advertisement-related functions using the SOAP protocol. From the viewpoint of upper-layer implementations, SOAP hides the complexity of under-layer communications and largely simplifies the interactions between server and client. With the assistance of code-generation tools (e.g., WSDL2Java), third-party developers can rapidly create the stub code running on the client side according to the WSDL file published by our system. Since most interaction tasks are encapsulated into the stub code, the remaining client applications only need to launch a local procedural call to obtain the information required without considering additional details about the interactions.

A diagram is provided in **Figure 9** for the purpose of depicting the process of SOAP-based interaction more clearly. In our implementation, all SOAP messages are assembled into the body of HTTP messages and exchanged over networks.

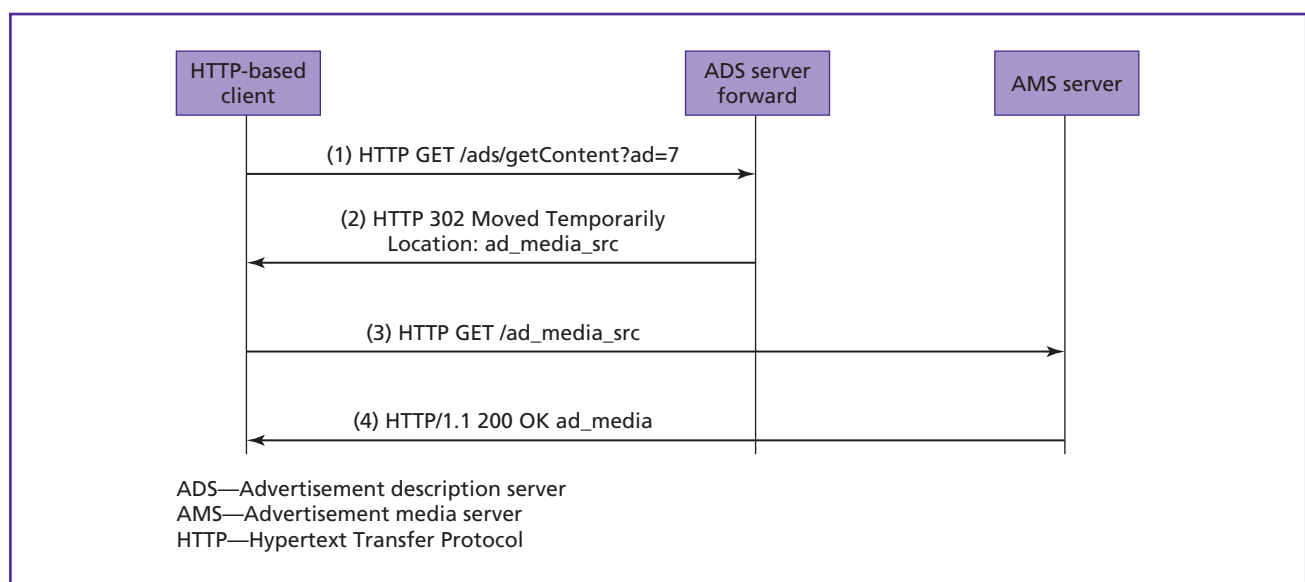


Figure 7.
Redirection of advertisement media request.

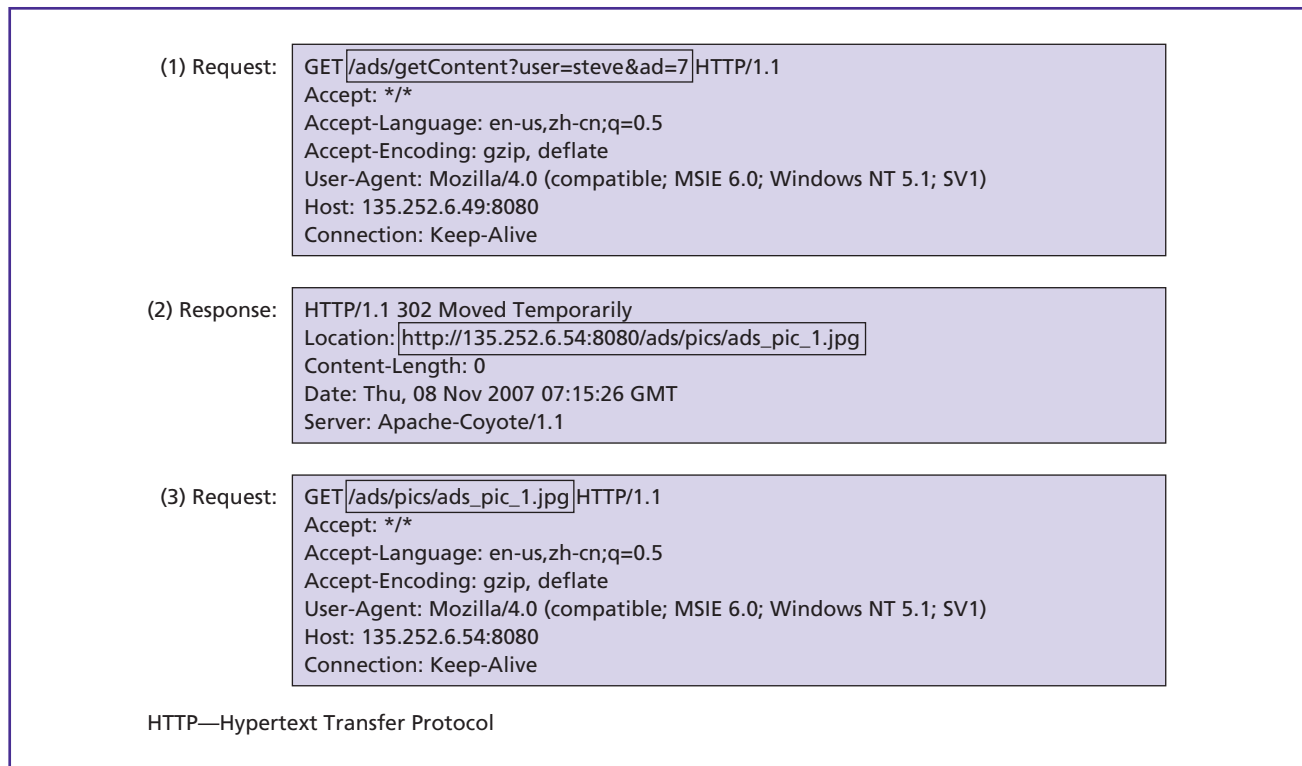


Figure 8.
HTTP messages in redirection.

Figure 9 shows the SOAP request generated when the Web service method “getAnAdById” is called with the parameter “id” equal to 7. The SOAP response shows the result that is returned to the client when the request processing is completed on the ADS server. Upon receiving the SOAP response, the client will call the stub code generated to parse the message and create corresponding objects for further processing.

Selection of Advertisements

As discussed, two distinct roles come into play with our advertising system—those of the advertiser and the publisher. The former provides the advertisement content, and the latter provides the region for publishing the content. Since the main task of our system is to perform ad delivery for client display, here a problem emerges—Given a region, how do we select an appropriate advertisement for delivery? A good selection method will ensure that the right

advertisement is delivered to the right region and viewed by the right person, to increase access to ads and generate more income. Our solution to this problem includes the following two strategies

Selection of Advertisements by Category Matching

Almost everything can be classified into a category. An advertisement, no matter what content it includes, certainly belongs to one or more categories. Similarly, a region might be expected to display the advertisements associated with specific categories. We therefore introduce the concept of “category” to build the relationship between advertisements and regions. Every region is assigned a set of categories and only the advertisements matching those categories are permitted to display in the region. The “category” assists in preventing spam and improving user satisfaction.

Therefore, the selection of advertisements for delivery could be described as a category-matching

SOAP request:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/....">
  <soapenv:Body>
    <getAnAdById xmlns="http://ws.ads.blrc.com">
      <id>7</id>
    </getAnAdById>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP response:

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/....">
  <soapenv:Body>
    <getAnAdByIdResponse xmlns="http://ws.ads.blrc.com">
      <getAnAdByIdReturn>
        <ad_author>John Smith</ad_author>
        <ad_desc>The perfect holiday idea ...</ad_desc>
        <ad_forward_url>http://135.252.6.49:8080/ads/forward?ad=7&...</ad_forward_url>
        <ad_id>7</ad_id>
        <ad_title>KFC Gift Checks</ad_title>
        <ad_type>jpg</ad_type>
        <ad_url>http://135.252.6.54:8080/ads/pics/ads_pic_1.jpg</ad_url>
        <create_date>2007-01-29 13:57:49.0</create_date>
      </getAnAdByIdReturn>
    </getAnAdByIdResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP—Simple Object Access Protocol

Figure 9.
SOAP messages in interaction.

operation within a specific region. The principle behind the matching is simple: each advertisement can be added into the group of delivery candidates on the condition that it maps to one or more categories within the region. **Panel 2** contains a simple expression for the matching operation written in SQL.

Algorithm for Calculating the Priorities of Selected Advertisements

Using our strategy for category matching, we can obtain a set of candidate advertisements for delivery. But, generally, only one advertisement can be displayed in a region at a certain time. Thus, we need to determine which advertisement is the most suitable one in the candidate group for delivery. In other words, the advertisement candidates should be ordered and only the top one in the queue will be the

final selection. Here we adopt a priority-based algorithm to perform the ordering operation.

We begin with some basic definitions:

- i is the sequence number for an advertisement.
- j is the sequence number for a specific statistics parameter. **Table I** lists some statistics parameters for measuring the degree of success/popularity for an advertisement. Actually, besides those listed in

Panel 2. Matching Operation in SQL

```
select distinct ads.*
from ads, region
where ads.category = region.category and
      region.id = xxx
```

Table I. Advertisement statistics parameters.

Sequence	Item	Description
1	Impressions	The times of an advertisement being delivered
2	Clicks	The times of an advertisement being clicked
3	CTR	The result of dividing the “clicks” number by the “impressions” number
4	Downloads	The download times of an advertisement medium
5	Expense	The advertisement expense paid by the advertiser

CTR—Click-through rate

Table I, there are other statistics parameters that can be used to measure advertisements. In order to simplify the implementation, the system adopted only the listed parameters. The parameter set may be extended by adding more parameters through some simple configurations.

- v_{ij} is the statistics value of the i^{th} advertisement for the j^{th} statistics parameter. For each advertisement, a periodic statistical task could be performed to obtain the parameter values to measure the operation of the advertisement. The statistics-related work is done by the strategy component in the AdsDE module.
- w_j is the weight of the j^{th} statistics parameter. Different parameters always have different attributes and are generally not comparable. We multiply each parameter with a weight to get a product that can be comparable, which is used for generating the final overall priority.
- ε_j : the importance ratio of the j^{th} statistics parameter. Different parameters generally have different degrees of importance. For example, some users may be interested in advertisements with a high click-through rate (CTR) value, while others may pay more attention to the impression. So we need to have a parameter to measure how much the average user cares about each statistical parameter. Here a conditional equation should be satisfied: $\sum \eta_j = 1$.

Then the formula for calculating the priority of the i^{th} advertisement can be shown as follows:

$$p_i = \sum_j v_{ij} \times w_j \times \eta_j \quad (1)$$

Herein, we define

$$V_i = [v_{i1} v_{i2} \cdots v_{in}], W = [w_1 w_2 \cdots w_n]^T, \\ Y = [\eta_1 \eta_2 \cdots \eta_n]^T,$$

with n being the total number of parameters.

We will get a formula written in a matrix form

$$p_i = [v_{i1} v_{i2} \cdots v_{in}] \\ \times \left(\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \cdot \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix} \right) = V_i \times (W \cdot Y) \quad (2)$$

With this formula, the AdsDE module can obtain the priorities for all the advertisement candidates, by which the candidate list could be sorted in a descending order. The advertisement at the top of the list will be selected for delivery.

For all advertisements in the system, we can get

$$P = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & v_{1n} \\ v_{21} & v_{22} & v_{2n} \\ v_{m1} & v_{m2} & v_{mn} \end{bmatrix} \\ \times \left(\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \cdot \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix} \right) = V_i \times (W \cdot Y) \quad (3)$$

From equation 3, we can see that the Y vector determines the contribution of each parameter to the final priority result. Thus, we may generate a new

delivery strategy by adjusting the values in the Y vector. For example, if we set the value of the “clicks” parameter to 1, and simultaneously set the other parameters to 0, the strategy related to the “clicks” parameter will take effect when computing the priorities of advertisements.

Furthermore, the vector for V is obtained from the statistical result based on detailed log information and reflects the current status of all advertisements. Therefore, if an advertisement is successful or popular to users, its related statistics parameters are always larger, resulting in a higher priority and a higher possibility of delivery.

Conclusion and Future Work

In this paper, we introduced an advertising system implemented by Bell Labs Research China (BLRC) and provided a detailed description of the advertisement delivery mechanism. Similar to the concepts of originating server and cache servers in a CDN, two kinds of servers—the ADS server and AMS server—are involved in the advertising system. They are distributed over networks but work cooperatively to perform advertisement delivery. This multiserver architecture provides for increased scalability and higher efficiency without the need to upgrade the existing network infrastructure. The collaborative relationship among servers is built on the foundation of a series of interactions, either inside the group of servers or between servers and clients. Finally, the strategy and algorithm used for ad selection were also discussed.

At present, the initial phase of this project is now complete, and a standalone advertising prototype—supporting features such as rich media types, interactive advertisements, extensible interfaces for third-party applications, flexible advertisement selection strategy, and Web-based maintenance and management—is now available. The system has been successfully integrated with two systems inside BLRC, MapWeb and the IP Multimedia Subsystem (IMS) personal computer (PC) client. The former is a location-based communication platform that provides users with a map interface for retrieving location-related information and communicating with adjacent entities. The latter is implemented for

laptop/desktop personal computers and allows for multimedia blended communications with wireless users by using several emerging technologies including instant messaging, voice over IP, and peer-to-peer video service.

However, this advertising system is far from completion. We plan to continue work on the following aspects:

- *Performance evaluation.* There is still a lack of quantitative analysis on the performance evaluation, which can help to measure the efficiency of the system. For example, further work is needed on how many customers can be served by the system, where bottlenecks exist, and what components should be optimized. The technologies of measurement and analytical modeling will be used to evaluate the system performance.
- *Support for more protocols and more devices.* At present, the system supports the HTTP and SOAP protocols. We are planning to offer support for additional protocols, for example, Wireless Application Protocol (WAP) and RTSP, to provide support for additional applications running on devices including mobile phones and IPTV.
- *Location.* The current system does not support location-based advertising, though actually advertisements are often location-related. For example, a supermarket may wish to advertise specific goods to nearby persons rather than widely broadcast an advertisement. Even though not all applications need the location feature, we still need to support this feature to make the advertising system more generic for coping with more requirements.

Other topics such as advertisement content filtering (to prevent spam) and user profile support (to improve user satisfaction) still need further careful study.

Acknowledgements

We would like to thank Dawei Huang, the director of BLRC, for his sincere encouragement and valuable suggestions. We also wish to express special appreciation for two internship students, Jun Zhang and Bo Zhou, who made good contributions to the system implementation. And we are grateful to two of

our BLRC colleagues, Xiangdong Shi and Jianfeng Chen, for their active cooperation during the integration with the MapWeb and IMS PC clients.

***Trademarks**

adCenter is a trademark of Microsoft Corporation.

AdWords is a trademark of Google Inc.

AXIS is a trademark of the Apache Software Foundation.

Google is a trademark of Google Inc.

Java and JavaServer Pages are trademarks of Sun Microsystems Inc.

MySQL is a registered trademark of MySQL AB Inc.

Tomcat is a trademark of the Apache Software Foundation

Yahoo!, Yahoo! Publisher Network, and Yahoo! Search Marketing are trademarks of Yahoo! Inc.

References

- [1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol—HTTP/1.1," IETF RFC 2616, June 1999, <<http://www.apps.ietf.org/rfc/rfc2616.txt>>.
- [2] Google, "AdWords," <<https://adwords.google.com>>.
- [3] Microsoft, "adCenter," <<http://adcenter.microsoft.com>>.
- [4] G. Pallis and A. Vakali, "Insight and Perspectives for Content Delivery Networks," Commun. ACM, 49:1 (2006), 101–106.
- [5] T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti, and G. Urvoy-Keller, "From Content Distribution Networks to Content Networks—Issues and Challenges," Comput. Commun., 29:5 (2006), 551–562.
- [6] World Wide Web Consortium, "SOAP Version 1.2," W3C Rec., 2nd ed., Apr. 2007, <<http://www.w3c.org>>.
- [7] Yahoo!, "Yahoo! Publisher Network," <<http://publisher.yahoo.com>>.

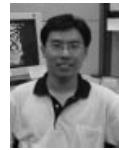
(Manuscript approved May 2008)

PENG ZHANG is a member of technical staff at Bell Labs



Research China in Beijing. He received his Ph.D. in computer architecture from Xi'an Jiaotong University in China. His research interests include the fields of network management, the IP version 6 (IPv6) network, object-oriented technologies, software agents, content delivery networks, and distributed computing. Dr. Zhang is currently engaged in the research and development of Web services, service-oriented architecture (SOAP), and advertising applications.

JUNTAO MA is a technical manager at Bell Labs



Research China in Beijing. He received his Ph.D. degree in computer science from Northeastern University, Boston, Massachusetts. Dr. Ma is currently engaged in the research and development of convergent applications, including service-oriented architecture (SOA) and location-based systems. His research interests include software engineering, mobile computing, and data networking.

XIAORONG SUN is a member of technical staff at Bell



Labs Research China in Beijing. She received her Ph.D. degree in communication cryptographs from Xi'an Electronic Science and Technology University in China. Dr. Sun's current research interests include network protocol and software, the IP Multimedia Subsystem (IMS), and Web services. ♦