Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS

Article *in* Journal of Advances in Computer Networks · January 2013

DOI: 10.7763/JACN.2013.V1.7

3 authors, including:

Phooi Yee Lau
Institute of Telecommunications
68 PUBLICATIONS **510** CITATIONS

SEE PROFILE

Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS

Khai Hsiang Wong , Phooi Yee Lau , Sungkwon Park

***Abstract*—The research focuses on simulating the concepts of Soarin and analyze if it would be viable to be put to used alike the un-released technology used currently in AKAMAI. Soarin is a content delivery system which is able to increase content network bandwidth dynamically by deploying delivery servers in a wide area. We discussed the concept of Soarin and proposed simulation as evaluation techniques. Simulation results discussed the relationship between the number of surrogate servers: (1) to its placement strategy, and (2) to its throughput.**

***Index Terms*—content delivery network (CDN), QoS, surrogate server**

I. Introduction

Content Delivery Network (CDN) replicates contents over several mirrored servers, named surrogate servers, which strategically place contents at various locations in order to deal with the *flash crowds*. CDN improves network performance by maximizing bandwidth usage, improving content accessibility and maintaining content updates through content replication thus offering fast and reliable applications and services by distributing contents to proxy servers often located closer to users.

The CDN architecture may consist of many surrogate servers that could deliver copies of same content to one or more users. The CDN include the request routing infrastructure, the distribution infrastructure, and the accounting infrastructures. The request-routing infrastructure could consist of a mechanism to redirect content requests from a client to a suitable surrogate, or a group of surrogate, server(s). The distribution infrastructure includes the mechanisms that allow contents be mirrored from the origin server to the surrogate servers. The accounting infrastructure is used to track and to collect data based-on the request-routing, distribution, and delivery functions within the CDN, creating logs and reports of distribution and delivery activities for server management. The client request is sent to a CDN through the request routing infrastructure and surrogate servers will be assigned to deliver the request. The origin server (hosting the content to be delivered) interacts, within the CDN, in two ways

a) It pushes new content to the surrogate servers, (the replica themselves request content updates from the origin server through the distribution infrastructure).

b) It requests logs and other accounting data from the CDN or the CDN itself provides this data to the origin server through the accounting infrastructure.

The optimal performance and reliability often depend on the granularity of the distribution of those surrogate servers; often refer to as edge servers. The establishment of a CDN therefore requires the design of some important features, namely; (1) replica placement mechanisms (to decide the locations of the replica server and to adaptively update the contents prior to the request arrival (pre-fetching). *Note*: Servers are not updated upon request, i.e. unlike in traditional proxy caching, but are pro-actively updating its content. Adaptive nature in replica placement is required, especially to cope with changing traffic conditions though not related to pull behavior as in traditional caching; (2) content update mechanisms (to automatically check the host site for changes and retrieve updated content for delivery to the edges of the network, thus ensuring content freshness). *Note*: Standard mechanisms adopted in proxy caching do not guarantee content freshness since content stored on standard cache servers does not change as the source content changes; (3) active measurement mechanisms (to cooperatively access routers in order to provide immediate access to a real-time picture of the Internet traffic, i.e. to the ability to recognize the fastest route from the requesting subscribers to the replica servers in any type of traffic situations, especially in presence of "flash crowds"). *Note*: A measurement activity is at the basis of the replica selection mechanism; (4) replica selection mechanisms (to cooperative access routers to accurately locate the closest and most available edge server from which the end users can retrieve the required content). *Note*: A robust service must also keep its servers from getting overloaded by means of access control and load balancing; (5) re-routing mechanisms (to quickly re-route content requests in response to traffic bursts and congestion as revealed by the measurement activity). *Note*: the CDN infrastructure, must allow the service providers to directly access the caches and control their availability and to get the statistics information about the accesses to the site, from the cooperative access routers.

This paper focuses on two core work: 1) to understand the concept of Soarin, and (2) to focus on simulating "Soarin" to provide an analysis of what could be possibly an identical structure of Akamai's Content Delivery Network. The

Manuscript received October 8, 2012. This work was supported in part by the Ministry of Culture, Sports and Tourism (MCST) and the Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program 2011.

Khai Hsiang Wong is an undergraduate student at Universiti Tunku Abdul Rahman, 31900 Kampar, Perak, Malaysia. (e-mail: david.wongkh@gmail.com).

Phooi Yee Lau is with the Universiti Tunku Abdul Rahman, 31900 Kampar, Malaysia. (e-mail: laupy@utar.edu..my).

Sung-kwon Park is with the Convergence Communication Laboratory, Hanyang University, Seoul, Republic of Korea. (e-mail: sp2996@hanyang.ac.kr).

remainder of this paper includes: Section II that discusses the background of content caching; Section III that discusses the previous work of content caching; Section IV that describes *Soarin* and outlines the conceptual design framework; Section V that discusses the proposed simulator and platforms; Section VI that evaluates the framework; and Section VII that discusses and concludes the paper with future work.

II. BACKGROUND

Over the last decades, users have witnessed the Internet growth and maturity. As a consequence, there has been an enormous growth in network traffic, driven by rapid acceptance of broadband access, along with increases in system complexity and content richness [1]. The over-evolving nature of the Internet brings new challenges in managing and delivering content to users. As an example, popular Internet sites often suffer congestion and bottleneck due to the large demands made on their services. A sudden spike in users' requests could overload video servers, and as an alternative, a *hotspot* [2] could be used. Coping with such unexpected demand causes significant strain on a video server. Eventually these servers could be totally overwhelmed and video servers holding the content could become temporarily unavailable. Content providers view the Internet as a vehicle to bring rich content to their users, free of charge. A decrease in service quality, along with high access delays mainly caused by long download times could leave users in frustration. Significantly, many big conglomerates earn financial incentives from Internet-based business. Hence, they could be concern on how to improve service quality of experience (QoE) or quality of service (QoS) by users. As such, these past few years, we have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Internet. Once we have it, we can use these infrastructures and form a new type of network, i.e. as content network [3].

III. PREVIOUS WORK

Several content networks attempt to address the performance problem, i.e. improve the QoS by using different mechanisms. One approach is to modify the traditional Internet architecture by improving the server hardware, such as using high-speed processor and multi-processor system, and/or more memory. These approaches are not flexible [4]. Moreover, small enhancements are often not possible and in some scenario, a whole server system may have to be replaced. Caching proxy deployment by an ISP can benefit the narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies should be deployed closer to the users. Caching proxies should also be equipped with server failure detection system in order to maximize the efficient use of caching proxy resources. Users often configure their browsers to send request through these caches rather than sending it all the way to the origin servers. Should this be properly configured, a user's entire browsing session would be send to only a specific caching proxy. Thus, the caches which contain most popular content viewed by all users will be cached in its respective edge proxies. A provider may also deploy different local, regional, international proxy server based-on geographically distributed locations. Such arrangement is referred to as *hierarchical caching*. This may provide additional performance improvements and bandwidth savings [5]. Another scalable solution is to establish server farms. It is a type of content network that has been frequently used for several years. A server farm comprised of multiple Web servers, each of them sharing the burden of answering requests for the same Web site [4]. It make use of Layer 4-7 switch, i.e. Web switch or content switch, that examines content request and dispatches them among the group of servers. A server farm can also be constructed using surrogates instead of a switch. This approach is more flexible and shows better scalability [4]. Moreover, it provides the inherent benefit of fault tolerance in [1]. The deployment and growth of server farms progresses with the upgrade of network links that connects the Websites 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. CDN can increase network bandwidth, i.e. the delivery servers are distributed in all over the Internet. Before using CDN to deliver contents, contents provider can estimate the amount of the access required in order to provide enough processing power and network bandwidth. However CDN cannot provide its services during overload. This is because CDN cannot increase its network bandwidth and has limited processing power flexibly. Cloud computing makes it possible to increase processing power dynamically by increasing the number of servers. However, current cloud-computing systems cannot increase its network bandwidth. This is because it can increase the number of servers only in a local area. Servers have to be deployed in a wide area to enjoy increases in network bandwidth. There are three problems for deploying servers in a wide area which includes (1) how to deploy the servers; (2) where to deploy the servers; and (3) when to deploy the servers. There are some research contributions for wide area live migration [6][7]. The results of these researches can be used in server deployment in a wide area [8][9][10].

IV. SERVER PROLIFERATION

Server proliferation deploys physical machines, named *Execution Server* (ES) that are installed with virtual machine monitor throughout the Internet in advance. These ES would execute virtual machines on them. The other server is *Distribution Server* (DS). DS stores the virtual machines HDD images. In Server proliferation, services (i.e. Web server, streaming server) will be executed inside virtual machines. When a new virtual machine is required, the virtual machine's HDD image will be distributed from DS to one of the ES. The distributed virtual machine is executed on the ES. Server proliferation realizes the increasing and decreasing processing power and network bandwidth of server system by

increasing and decreasing servers in a wide area dynamically, shown in Figure 1. Without server proliferation, a CDN will be just like cloud computing which makes it possible only to increase servers dynamically. By increasing servers, it is possible to use the CPU and the network of the new servers, thus, the processing power and network bandwidth could be increased. However, compared with processing power, it is more difficult to increase network bandwidth. This is due to the (1) network bottleneck, and (2) typical cloud-computing system is constructed in an iDC (Internet Data Center) as the uplink network of the iDC could become the network bottleneck of the cloud system.

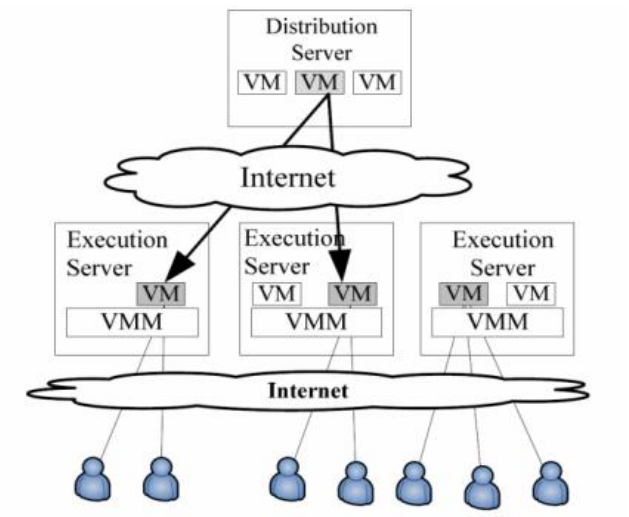


Fig. 1: General View of Server Proliferation

In contrast, server proliferation can increase both of processing power and network bandwidth. Server proliferation uses virtual machine as a basis, thus, it can deploy servers in a wide area; therefore, each deployed servers can use other different uplink network. In this case, where SOARIN is concerned, it does not matter how many servers are to be deployed in the specific area; with virtualization, the high costs required for server deployments can be avoided. Thus, it is possible to increase network bandwidth of the system, without much effort, as virtual machine is easy to increase and decrease dynamically.

A. SOARIN

In this section, we describe the Flexible Contents Delivery System with Dynamic Server Deployment: **SOARIN**. SOARIN enables network bandwidth to be flexibly increased by increasing the DSs, anytime. It is possible to add network bandwidth even after content distribution has started. In addition, SOARIN can decide when and where to increase the number of DS. When the DSs are being increased dynamically, new problem could arise, such as when and where to increase the number of surrogates. In SOARIN, DSs are constructed inside virtual machines; this problem is equal to another problem, i.e. which physical machine/criteria to execute another virtual machine. This selection criterion is called *Server Deployment Policy*. There are many varieties of server deployment policy and SOARIN allows using various server deployment policies for contents holders' requirements. The examples of the server deployment policy are discussed later.

SOARIN uses server proliferation to deploy DSs dynamically. However it lacks the capability to decide the timing and location when deploying virtual machine. Hence, the introduction of *Observation Server* (OS) and *Control Server* (CS), in addition to ES and DS in server proliferation explained earlier. OS collects several kinds of metrics using server deployment policy. For example, it collects the CPU load and network traffic of ESs, calculates the distance from ESs to DSs. CS controls the overall SOARIN's system. CS selects ES to deploy new virtual machine using the information from OS based-on server deployment policy. CS will then direct DS to transfer the HDD image of the virtual machine to the selected ES to deploy new DS. After that, CS directs ES to execute the virtual machine for new DS. Finally CS updates request navigation policy to use the new DS. Figure 2 illustrate a general overview of OS, CS, DS and ES.

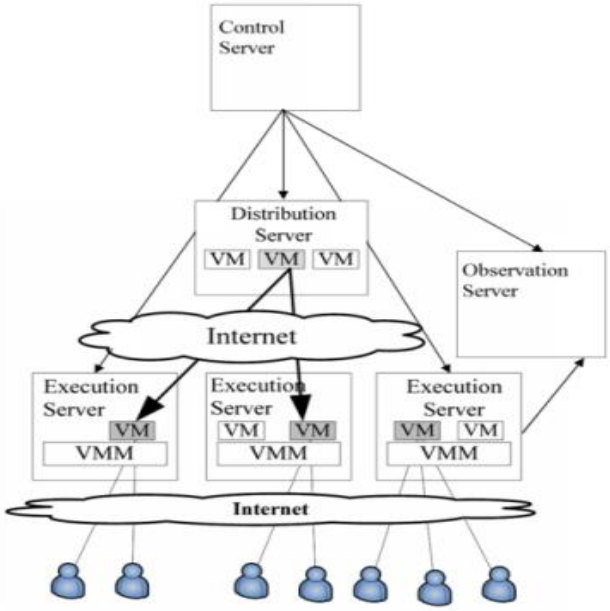


Fig. 2: General Overview of CS and OS

1) Server Deployment Policy of SOARIN

As mentioned above, SOARIN is able to use various server deployment policies. In this work, however, only one Server Deployment Policy of SOARIN will be used to deal with a scenario of a sudden spike in user connected (peak hour) which is where we take into account the distance between the ES and DS; i.e. how far an ES would be from the DS (origin server).

2) Distance between ES and DS

This policy uses the distance between DS and ES as a criterion. Some metrics can be used to calculate the distance which includes number of hops, round trip time, and AS (Autonomous System) path length between DS and ES. DS measure this information periodically. OS collects this information from DS. This policy chooses the nearest ES from DS; therefore it may deploy new DS in a short time. As a result, it is possible to correspond to a sudden surge in the volume of request (peak hour). In this instant, when a surrogate is not able to fulfill the client's request, the request will be directed to the nearest surrogate or origin to fulfill that request.

V. Platform and Tool

CDNsim [11] has been designated to provide a realistic simulation for CDNs, simulating surrogate servers, TCP/IP protocol, and other CDN functions. The main advantage of this tool is its high performance, its extensibility, and its user interface, which can be used for configuring its parameters. CDNsim provides an automated environment for conducting experiments and extracting client-, server-, and network-statistics. The purpose of CDNsim is to be used as a testbed for CDN evaluation and experimentations subsequently making it a suitable choice to be used for the objectives of this work. In terms of simulating a CDN, there are various software out there that can serve such purpose (i.e. NS-2, OPNET). However, the reason we select CDNsim is because it was built on top of OMNET, i.e. an extension specifically used to simulate CDN on a Linux Platform.

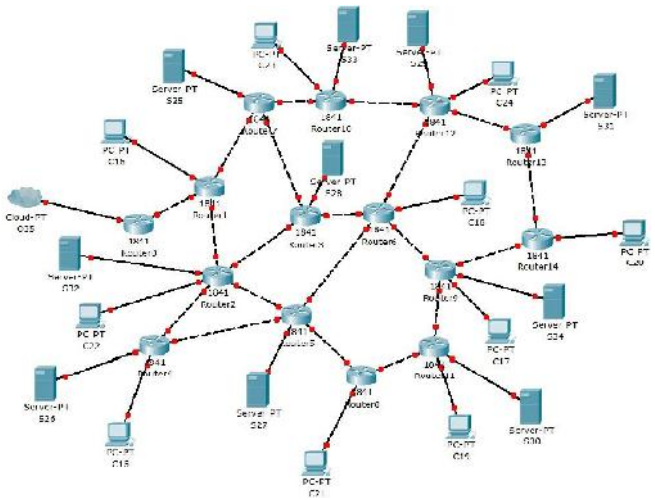


Fig. 3: 10 surrogates with 10 clients

In terms of SOARIN, the ability to manipulate topologies allowed in CDNsim enabled us to create a simple topology. Using the CDNsim.py we are able to build the topology for all types of CDN which includes all the options of manipulating the number of clients, servers and origins, see Figure 3. The sh script enable us to simulate the topology coupled with using the .ned and .hi inside the topology and by calling the .cc file in CDNsim lib, INET lib and HACK folder. After running the simulation, the simulation process output will be logged into stats as STDOUT file, for analysis. On modifying the topology, CDNsim allows us to modify the model of the topology. CDNsim provides the basic model for building a CDN. By assuming that the central unit is the request routing system, we organized the CDN as such: (1) one of the nodes does not contain a certain file, (2) central unit, being the one that controls all surrogate servers within the singular CDN, will route the missing file from the origin server or from the closest node that has the file. When the simulation is complete, from the *stat* file, we will be able trace the routing of the CDN, which shows the traffic of the packet transfer between nodes.

```
1.000000,SURROGATE,s28,MISS,2
1.000000,ORIGIN,o35,HIT,2
COMPLETED,CLIENT,c16,2,0,1.000000,1.003455,-
```

Fig. 4: Sample Record 1

By cross-referencing the SDTOUT and stats files recorded above, we will be able to see how each of the line referenced is made. Referring to Figure 4 and 5, we look at Line 8, time: 1.0, C16 requested for Object 2. Upon checking, the closest surrogate which responded is s28 replying that only the origin has object 2. Hence, s28 re-directs the request to the origin,o35 which then proceeds to fulfill the request at time 1.003455

6	UTIL_DOWN	1.0019	s28	58512	2
7	UTIL_UP	1.00204	s28	58512	2
8	UTIL_DOWN	1.00341	c16	58512	2

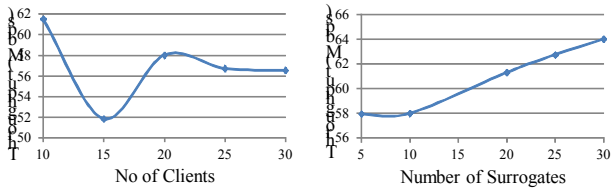
Fig. 5: Sample Record 2

VI. Simulation Results and Discussion

A. Simulation Case 1 (Closest Surrogate)

This section discusses the simulation in terms of Closest Surrogate. In reference to CDNsim, Closest Surrogate is where the client is redirected to the closest surrogate server in terms of network hops. Upon a cache miss, the surrogate server retrieves the object from the closest alternative surrogate server that contains the requested object. The object is stored in the cache and then it is served to the client. If the object is not outsourced at all in any surrogate server then the surrogate server retrieves the object from the closest origin server. Consecutive cooperation may occur if in the meantime the alternative surrogate server removed the object from the cache causing another cache miss. There will be 2 models which will be simulated in terms of Closest Surrogate where Model 1 has a fixed number of 10 surrogates but will have increasing number of clients and Model 2 where it has a fixed number of 20 clients but will have an increasing number of surrogates.

Fig. 5: Results for (a) Surrogate Fixed (b) Client Fixed



(a) (b)

Fig. 6: Results for (a) Surrogate Fixed (b) Client Fixed

Results

It was observed that "Closest Surrogate" is able to handle the increases of clients but needs time to stabilize the increased capacity. In Figure 6 (a), we are able to observe that when the number of clients climbed to 15, the throughput drops quite drastically. However, as the number of clients continues to increase, we are able to see that the throughput slowly improves and though not as good as the 1:1 surrogate-client ratio; is still quite impressive despite having the surrogate-client ratio at 1:2. As we look at Figure 6(b), when the number of clients is fixed and the number of surrogates is increased, we are able to see that the throughput goes into an upward pattern.

B. Simulation Case 2(Random Surrogate)

This section discusses the simulation in terms of Random Surrogate. In reference to CDNsim, the client is redirected to a random surrogate server. Upon a cache miss, the surrogate server retrieves the object from a random alternative surrogate server that contains the requested object. The object is stored in the cache and then it is served to the client. If the object is not outsourced at all in any surrogate server then the surrogate server retrieves the object from a random origin server. Consecutive cooperation may occur if in the meantime the alternative surrogate server removed the object from the cache causing another cache miss. Again, there will be 2 models which will be simulated in terms of Random Surrogate where Model 1 has a fixed number of 10 surrogates but will have increasing number of clients (10, 15 and 20) and Model 2 where it has a fixed number of 20 clients but will have an increasing number of surrogates(5,10 and 20).

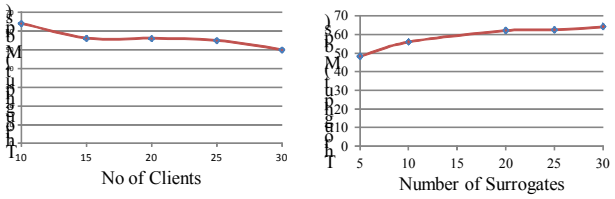


Fig. 7: Results for (a) Surrogate Fixed (b) Client Fixed

Results

It was observed that "Random Surrogate" works best at 1:1 Surrogate-Client ratio. In Figure 7(a), we are able to observe that as the number of client increases, the throughput goes through a decline pattern. While when the number of clients is the same as the number of surrogates, the throughput is recorded to be quite good. However, as the number of clients starts outnumbering the number of surrogates, the throughput starts to go down. As we look at Figure 7 (b), when the number of clients is fixed and the number of surrogates is increased, the throughput slowly climbs back as the surrogate-client ratio goes back to 1:1.

C. Discussions

It was found that "Closest Surrogate" performs better when compared side by side with "Random Surrogate". Both the "Closest Surrogate" and "Random Surrogate" is put to the test with the same models and while we are able to see that "Closest Surrogate" needs a bit of time to accommodate the increases in capacity, "Random Surrogate" works best only at 1:1 surrogate-client ratio. While in the perfect world, for every number of clients we will have a surrogate accommodating its requests, however, this is impractical in real life. Hence, a 1:1 surrogate-client ratio in real life is impossible and that we can only expect that a CDN uses "Closest Surrogate" protocols as it is clearly seen that, despite needing some time to get used to an increase in capacity, it generally is able to recover and perform just as well as a 1:1 surrogate-client ratio.

VII. Conclusion and Future Work

Existing CDNs have always struggled to use the perfect replica placement mechanism to fully expose the abilities of CDN, i.e. to provide on-demand content with satisfactory performance. Akamai has been very successful in doing so on a large scale all over the world. However, the technology that they have been implementing has never been revealed nor shared to the general public in the hopes of monopolizing the CDN market. The concept of *Soarin* is rather similar to what has been seen with Akamai's technology. Though we may never be able to be sure what Akamai's been using, *So*arin looks a good fit for the vague description of Akamai's technology. The biggest stumbling block however, is that *Soarin* has never been be proven to be a good concept to work with. This is where the objective of this work comes in. This work started by analyzing how *Soarin* works. We would then simulate the concepts, analyze the results and go on to propose an extended-*Soarin* for a new scenario. While we have understood that using the "Closest Surrogate" protocols works best in a CDN, we are still far from fully able to utilize *Soarin*. Before putting *Soarin* to real life practices, it has to go through further studies before qualifying as a proper model to Akamai's unrevealed technology. For future works in terms of an extended-*Soarin*, one should to go full scale and not just simulate and actually use a proper physical OS to monitor the uplinks, downlinks, network traffic, throughput of a CDN, and a more in-depth study done on how to improve "Closest Surrogate" (maybe of not using network hops as reference and using another metric).

References

[1] M. Hofmann, and L. R. Beaumont, *Content Networking: Architecture, Protocols, and Practice*, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129-131, 2005.

[2] S. Adler, "The Slash Dot Effect: An Analysis of Three Internet Publications," *Linux Gazette Issue*, Vol. 38, 1999.

[3] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, "A Model for Content Internetworking (CDI)," *Internet Engineering Task Force* RFC 3466, February 2003. www.ietf.org/rfc/rfc3466.txt

[4] M. Hofmann, and L. R. Beaumont, *Content Networking: Architecture, Protocols, and Practice*, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 132-134, 2005.

[5] N. Bartolini, E. Casalicchio, and S. Tucci, "A Walk Through Content Delivery Networks," in *Proceedings of MASCOTS 2003*, LNCS Vol. 2965/2004, pp. 1-25, April 2004.

[6] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally Distributed Content Delivery," *IEEE Internet Computing*, pp. 50-58, September/October 2002.

[7] Akamai Technologies, Inc., www.akamai.com, 2007

[8] Z. Xu, Y. Hu,L. Bhuyan ,"Efficient Server Cooperation Mechanism in Content Delivery Network", in *Proceedings of the 25th IEEE International Conference on Performance, Computing, and Communications (IPCCC 2006)*, pp.8 pp.-440, 10-12 April 2006

[9] Y. Chen, R.H. Katz, J.D. Kubiatowicz, "Dynamic Replica Placement for Scalable Content Delivery", in *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS 02)*, LNCS 2429, Springer-Verlag, pp. 306-318, 2002.

[10] Y. Kamiya, T. Shimokawa, F. Tanizaki, N. Yoshida. "Scalable Contents Delivery System with Dynamic Server Deployment", *International Journal of Computer Science Issues*, Vol. 7, Issue 6, November 2010.

[11] K. Stamos, G. Pallis, A. Vakali, D. Katsaros, A. Sidiropoulos, and Y. Manolopoulos, "Cdnsim: A simulation tool for content distribution networks," ACM Transactions on Modeling and Computer Simulation, vol. 20, no. 2, 2010.