ADVANCED CONTENT DELIVERY, STREAMING, AND CLOUD SERVICES

Edited by

Mukaddim Pathan
Telstra Corporation Ltd., Australia

Ramesh K. Sitaraman
University of Massachusetts, Amherst and
Akamai Technologies, USA

Dom Robinson
id3as-company Ltd., UK

WILEY

Cover Image: iStockphoto © nadla
Cover Design: Wiley

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Advanced content delivery, streaming, and cloud services / editors, Mukaddim Pathan, Ramesh K. Sitaraman, Dom Robinson.
pages cm
Includes index.
ISBN 978-1-118-57521-5 (hardback)
1. Cloud computing. 2. Computer networks. I. Pathan, Mukaddim. II. Sitaraman, Ramesh Kumar, 1964- III. Robinson, Dom.
QA76.585.A377 2014
004.67′82–dc23

2014005235

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

13

DYNAMIC RECONFIGURATION FOR ADAPTIVE STREAMING

Norihiko Yoshida

Information Technology Center, Saitama University, Saitama, Japan

13.1 INTRODUCTION

When a website catches the attention of a large number of people, it gets an unexpected and overwhelming surge in traffic, usually causing network saturation and server malfunction, and consequently making the site temporarily unreachable. This is the *flash crowd* phenomenon on the Internet.

For a content delivery network (CDN) to be scalable in such dynamic situations, the CDN should be dynamically reconfigurable. The network would adaptively change its topology and volume according to the observed traffic load and pattern. For delivery and distribution of static contents such as texts and images, there have been a fair amount of studies and systems, some of which are already employed in industry. However, for delivery and distribution of streaming contents such as voices, music, and videos, further studies are still required and anticipated. The difficulty of a streaming CDN comes mostly from the continuity of the contents and the QoS (quality of service) requirement. A stream is delivered spending a certain period of time, meanwhile a load to the CDN would change. The CDN must adapt dynamically during the period by changing its network in the same way as static content delivery and distribution, and also by adjusting the QoS of the stream dynamically. One of the most promising solution to the issues

Advanced Content Delivery, Streaming, and Cloud Services, First Edition.
Edited by Mukaddim Pathan, Ramesh K. Sitaraman, and Dom Robinson.

is to divide a stream into several slices or segments. The CDN changes and adjusts per segment. However, in this case, the problem is how to manage the integrity of the whole set of segments as a single stream in the dynamic situation.

This chapter presents a dynamic streaming CDN. It is an extension to a dynamic CDN for static contents and incorporates with solutions to the issues mentioned earlier. This scheme copes with situations that a large and varying amount of clients request a sequential stream from a streaming server. The presentation partly uses some examples, FCAN (Flash Crowds Alleviation Network) as a CDN and HyperText Transfer Protocol (HTTP) live streaming for stream segmentation; however, it is supposed to be generally applicable.

Dynamic load distribution or load balancing is one of the most important issues in both CDNs for static and stream contents. To handle increase and decrease in surrogates, most systems use virtual machine replication and migration. This scheme incurs much overhead on transferring the image of a virtual machine. There is another promising technique for dynamic replication and migration using mobile threads which leads to more lightweight, low overhead. This chapter shows an overview of this technique as well.

This chapter is organized as follows. After summarizing backgrounds and some related works including an overview of FCAN in Section 13.2, Section 13.3 presents dynamic server deployment based on mobile threads. Then, Section 13.4 describes adaptive dynamic CDN for streaming. Section 13.5 gives some future research directions, and Section 13.6 contains concluding remarks.

13.2 BACKGROUND AND RELATED WORK

13.2.1 Flash Crowds and Adaptive CDNs

The term *flash crowd* was coined in 1973 by a science fiction writer Larry Niven in his short novel *Flash Crowd* [1]. In the novel, cheap and easy teleportation enabled tens of thousands of people worldwide to flock to the scene of anything interesting almost instantly, incurring disorder and confusion. The term was then applied to similar phenomena on the Internet in the late 1990s. When a website catches the attention of a large number of people, it gets an unexpected and overwhelming surge in traffic, usually causing network saturation and server malfunction, and consequently making the site temporarily unreachable. This is the *flash crowd* phenomenon on the Internet.

Researches to alleviate flash crowds are divided into three categories: server-layer, intermediate-layer, and client-layer solutions, according to typical architectures of networks.

Server-Layer Solution. Systems in this category form delivery networks on server side similar to that of conventional CDNs. This is an expensive approach. The systems are inefficient on and difficult to deal with short-term Internet congestion. For example, CDN with Dynamic Delegation [2] and DotSlash [3] are in this category.

Intermediate-Layer Solution. Systems in this category let proxy servers work together for load balancing. Proxies in the system are mostly volunteers, and often less powerful than servers in the server-layer solution; however, caching techniques help to alleviate server load during flash crowds by filtering out repeated requests from groups of clients that share a proxy cache. Multilevel caching [4], BackSlash [5], and CoralCDN [6] are included in this category.

Client-Layer Solution. Systems in this category make clients help each other in sharing contents so as to distribute the load burden from a centralized server. Clients form peer-to-peer (P2P) overlay networks and use search mechanisms to locate resources. This is a costless approach. However, it is difficult to manage and control the clients, and to make the system reliable, secure, and transparent to users. CoopNet [7] and PROOFS [8] are included in this category.

13.2.2 FCAN

FCAN is an adaptive CDN that takes the form of client/server (C/S) or CDN depending on the amount of accesses from clients. Specifically, in the C/S mode, a server provides contents to clients as in a traditional C/S. In the CDN mode, when the server detects a coming of a flash crowd, volunteer cache proxies in the Internet construct a temporary P2P network and provide the content on behalf of the server. These volunteer proxies are recruited in advance out of providers and organizations. In case servers in such providers and organizations suffer from flash crowds, they will be helped by other volunteer proxies. FCAN is built on this mutually aiding policy. Figure 13.1 shows an overview of FCAN. FCAN is an intermediate-layer solution, which employs an Internet infrastructure of cache proxies to organize a temporal P2P-based proxy cloud for load balancing. However, FCAN has some extensions with some dynamic and adaptive features. Our FCAN studies achieved very promising results regarding static content delivery on the real Internet [9–11].

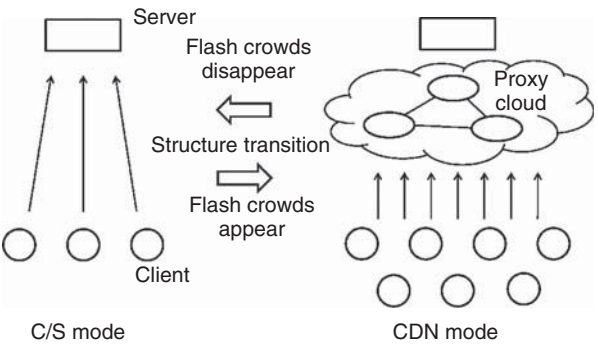

Figure 13.1 FCAN overview.

13.3 DYNAMIC SERVER DEPLOYMENT

To enable dynamic reconfiguration of networks, we must have some mechanism for dynamic server deployment. Without it, we must distribute the identical images of the server to all the hosts in the network beforehand even if the image would be actually not used after all. It would possibly cause wastes of computing and networking resources. There are two types of techniques for server deployment: virtual machine migration and mobile thread migration.

13.3.1 Virtual Machine Migration

One approach is migration and duplication of server virtual machines between hosts. Below are some examples.

vMatrix [12] focuses on distribution and replication of dynamic contents. vMatrix is a network of real machines running virtual machine monitor software, allowing server virtual machines to be moved among the physical machines. vMatrix can improve the response time perceived by end-users, overall availability and uptime, and on-demand replication to absorb flash crowd requests by optimizing the virtual server placement. vMatrix can also reduce the overall bandwidth consumed in the network in the same manner.

XenoServer [13] focuses on service deployment. A user first selects the number of XenoServers on which services are to be deployed, then configures the XenoServers so that the virtual machines accommodate the service components, and launches the service components.

In server proliferation [14,15], services (e.g., Web server, streaming server) are executed on virtual machines. When a new virtual machine is required, a disk image of the virtual machine is delivered from the distribution server (DS) to one of the execution servers (ESs), and the delivered virtual machine runs on the ES. In this architecture, DS may be multiplied so as to avoid load concentration.

13.3.2 Mobile Thread Migration

The other approach is migration and duplication of mobile threads between server hosts. It is more difficult to deploy, but would be much more efficient than the above.

A mobile thread, or sometimes called mobile agent, is a kind of thread that can move among computers over a network while keeping its program code and execution state [16]. Its idea came to help development and operation of large-scale network applications. Conventional network applications are designed based on communication. The idea of mobile threads separates computation from a physical platform and unifies computation and communication instead. Communication is now enclosed within computation, and this encapsulation is also expected to reduce network traffic practically. There are some mobile thread systems such as Aglets [17], MOBA [18], AgentSpace [19], and JavaGo [20].

Thread migration applied to dynamic network reconfiguration has been found mostly in grid computing based on distributed shared memory [21–23] and mobile threads [24], however not found in CDNs thus far.

13.4 FROM CONTENT DELIVERY TO STREAMING

13.4.1 HTTP Live Streaming

Conventional standard streaming protocols such as progressive download and real-time streaming do not allow switching of stream source servers on the client side dynamically. Therefore, massive accesses from clients concentrate on a particular site on the Internet. Accordingly, the server and its surrounding network choke up, and a flash crowd occurs.

In order to resolve this problem of conventional protocols, Apple has introduced a new protocol for video streaming, HTTP live streaming (also known as "HLS") [25]. It is defined in a standard draft for the Internet Engineering Task Force (IETF) [26]. Figure 13.2 shows an overview of this protocol. It would be beneficial for the readers to refer to Chapter 2 in this book as well for the details of HLS.

The server starts providing a video stream with the following procedure: (i) encodes an audio/video inputs; (ii) divides the encoded stream into a set of media segments and makes an index which refers them; (iii) delivers them to clients using HTTP on the Internet.

This protocol enables a client to switch the source server dynamically as opposed to the conventional streaming protocols. The delivery system archives load distribution easily with additional servers.

As another key feature, HLS supports "adaptive bitrate." The server provides alternative streams with different quality levels of bandwidths, so as to enable a client to optimize the video quality according to the network situation, as the load on the network and central processing unit (CPU), on both the server side and the client side, fluctuates on a frequent basis.

13.4.2 Adaptive Streaming on FCAN

Enhancing the HLS technology over FCAN, we built a dynamic CDN for both live and on-demand streaming [27]. The network changes its structure and configuration adapting to request loads. Its design, implementation, and experiment results are shown.

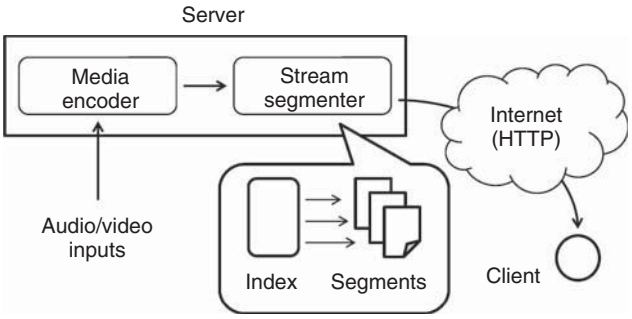

Figure 13.2 HTTP live streaming overview.

13.4.3 Structure Transition

The server and the cache proxies in the proxy network always monitor the amount of accesses they receive from clients and evaluate the load on the network. The system switches to the CDN mode if all nodes' loads are higher than a certain threshold, and switches back to the C/S mode if lower. Each cache proxy sends its own load information to the server periodically, and the server determines whether to perform structure transition.

In peaceful times, the conventional C/S architecture satisfies most of the client requests. A server and cache proxies, both of which comprise FCAN, do little more than what normal ones do. When a flash crowd comes, the server detects the increase in traffic load. It triggers a subset of the proxies to form an overlay, through which all requests are conducted. All subsequent client requests are routed to this overlay.

The server-side procedure is outlined as follows: (i) selects a subset of proxies to form a CDN-like overlay of surrogates and builds a distribution tree; (ii) pushes the index file and stream segments to the node of the distribution tree, so as to meet the real-time constraint of the video streaming; (iii) prepares to collect and evaluate statistics for the object from the involved proxies, so as to determine dynamic reorganization and release of the overlay.

The proxy-side procedure is outlined as follows: (i) changes its mode from a proxy to a surrogate (or, in the strict sense, a mixed mode of a forward proxy and a surrogate); (ii) stores flash-crowd objects (except the index file) permanently, which should not expire until the flash crowd is over; (iii) begins monitoring the statistics of request rate and load and reporting them to the server periodically.

In the live streaming, the index file is updated periodically; therefore, the server monitors its composition and pushes the segments at an appropriate time. Meanwhile, when the proxy is released by the server, it discards the index so as to keep the consistency among the nodes.

When the member server detects the leaving of the flash crowd, the involved proxies are dismissed one by one with the following procedure: (i) the server notifies the proxy to be dismissed; (ii) the server requests the related proxies to modify the relation of connection; (iii) the proxy changes its mode from a surrogate to a proxy.

The CDN-like overlay transits back to the normal C/S mode when all the proxies are dismissed. They are not all dismissed at once, since the low load may be just temporary, and the system should therefore remain in the anti-flash-crowd mode for a while.

13.4.4 Dynamic Resizing and Quality Restriction

The proxy network is a pure P2P network. Therefore, it is highly fault tolerant and scalable. Contrary to traditional P2P systems, it does not include clients into the network itself in order to assure reliability and security.

FCAN resizes a scale of the proxy network depending on a load fluctuation adaptively in order to avoid troubles such as server down by massive access concentration. Figure 13.3 shows how the system works in the CDN mode.

When the server detects a coming of flash crowds, it forms a temporary proxy network as shown in the left-hand side of Figure 13.3. If the initial network cannot handle

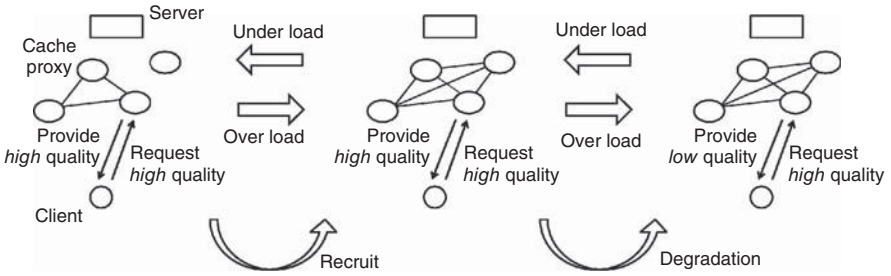

Figure 13.3 Adaptation to load change in the CDN mode.

increasing an amount of accesses, the server recruits a new member proxy one by one as shown in the middle of Figure 13.3.

If the server cannot recruit temporary proxies any more, it degrades the quality of the video stream as shown in the right-hand side of Figure 13.3. For example, in the situation that the system provides video streams of two different qualities, high and low, it delivers the low quality content as substitute for the high quality content under this quality restriction. The network occupancy per client decreases so that the server can alleviate the load of whole delivery network.

If the proxy network can easily handle all the incoming loads, the server may lift the restriction of the stream quality at first. After the derestriction, it releases temporarily recruited proxies one by one until all proxies are dismissed. Finally, the system all turns back to the normal condition.

13.4.5 Results

We conducted some preliminary experiments on a real network with a prototype of the system. Figure 13.4 shows an overview of the experiments. In our experiments, we use some hosts in Saitama University and Kyushu Sangyo University for a server, proxies, a pseudo client, and a client node. We use Apple's stream segmenter (mediastream-segmenter) for the segmenter and QuickTime Player for QuickTime X in the client.

The pseudo client is to trigger the FCAN's functions against flash crowds. It submits requests for randomly chosen segments to the server following the pattern shown in Figure 13.5. In the rest of this section, CP1, CP2, and CP3 are the proxies shown in Figure 13.4.

We made two experiments with two delivery methods, live and on-demand. In the on-demand streaming experiment, the server provides high and low quality contents with adaptive bitrates. We use segment samples of HLS in the Apple Developer's site [28]. The client software, the client has a master index file indicating these two quality contents, and the QuickTime Player requests segments of an adequate quality depending on load fluctuation following the master index. On the other hand, in the live streaming experiment, the server provides contents in a single quality in real time.

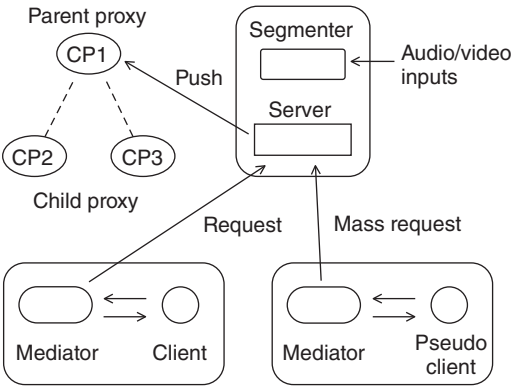

Figure 13.4 Experimental setup.

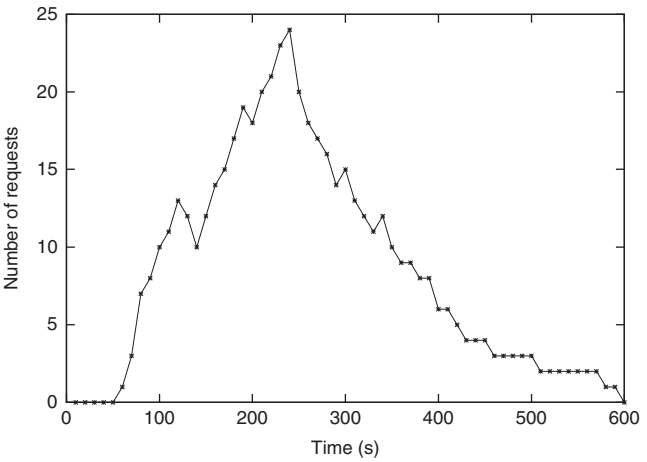

Figure 13.5 Sample request pattern of clients.

The server computes a load value regarding the size of requested segments. In the experiments, thresholds for load detection are defined beforehand based on some experiences. Workloads on the real Internet vary, and automatic and dynamic configuration of the thresholds is difficult. We suppose they may be configured based on the server capacity and the network bandwidth around the server.

Figure 13.6 shows the load transitions of the server with FCAN and without FCAN in the on-demand experiment. The case of the server with FCAN shows the average loads of member nodes in the distribution network. The first peak at the 40th second shows that "buffering" was done when the client started the playback as mentioned earlier. The load on the server exceeded the higher threshold at the 130th second, and then structure

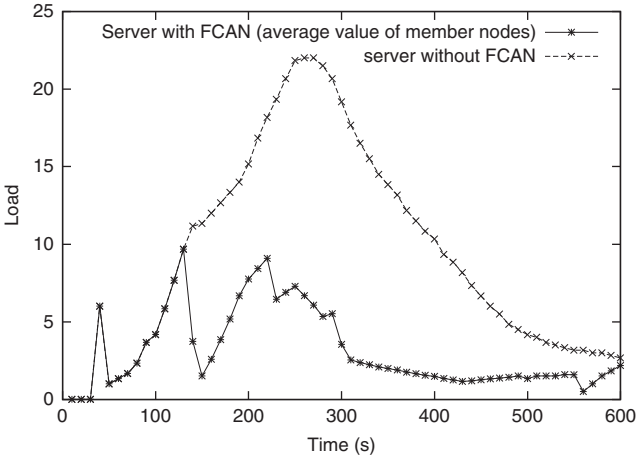

Figure 13.6 Comparison of load transitions in on-demand streaming.

transition to the CDN mode occurred so as to alleviate load concentration. However, the load on the member nodes continued to increase even after the transition, a new member proxy was recruited at the 220th second, and in addition the quality of segments was degraded at the 250th second, and consequently the server withstood the heavy load condition.

On the contrary, in the case of the server without FCAN, we observed that load values were far exceeding values of the server with FCAN consistently. We, therefore, confirmed that FCAN's features against flash crowds were performed as a result of the increase in client requests, and FCAN archived dynamic load balancing. We obtained equivalent results also in the live streaming experiment.

13.5 FUTURE RESEARCH DIRECTIONS

In order to fully deploy dynamic CDN for stream delivery and distribution, this chapter presents (i) dynamic network reorganization, (ii) load distribution and balancing, (iii) stream segmentation, and (iv) QoS assurance.

1. *Dynamic Network Reorganization*. There are two sides in network reorganization: the technical side on how to reorganize, and the management side on whom to reorganize with. Issues related to the former are studied intensively in the area of P2P networks. In large CDNs or global CDNs where voluntary hosts or providers are sometimes involved, centralized management is not effective, and autonomous management of join and leave of surrogates would be preferable. Research results on P2P networks must be helpful for that.

2. *Load Distribution and Balancing*. As stated earlier, the mobile thread technique is promising from the overhead point of view. However, it has not been fully investigated in CDN-related studies compared to the virtual machine migration technique. It may be because the mobile thread technique is rather new and is more difficult to implement especially in heterogeneous environments. The virtual machine technique, VMware [29] and Xen [30] for example, has been well developed to cope with such environments. Large CDNs or global CDNs would be heterogeneous in OS, middleware, and so on, and techniques to integrate them are essential. The mobile thread technique is no exception. Implementation and deployment in heterogeneous environments must be studied for the fruitful future of the mobile thread technique not only in CDN contexts but also in general actually.
3. *Stream Segmentation*. HLS is defined in an IETF draft and now being vigorously discussed. As a possible extension, a segmentation method adaptive to network environments and QoS requirement should be interesting.
4. *QoS Assurance*. QoS in stream delivery and distribution has already been thoroughly investigated in many related research areas. However, QoS assurance assisted by network reorganization is a new challenge. It must have a close correspondence with the research topic, dynamic resilient streaming [31]. The word "resilience" is defined as the persistence of avoiding failures and malfunctioning in the presence of changes. In the case of QoS, the aim is at persistence of acceptable quality in the presence of changes. The technique presented in this chapter is a basic one, and various extensions are possible including multipath streaming, network coding, and underlay awareness.

   There are also some other issues that should be studied for dynamic streaming CDN, such as (5) load estimation and (6) request redirection.
5. *Load Estimation*. A dynamic CDN requires any estimation method for host load and network load to determine how to reorganize its network and how to distribute loads. The most accurate method to estimate throughput and latency would be to measure round trip time of a small dummy message. However, if all the clients applied this, it would cause an equivalent to distributed denial of service (DDoS) attacks to the server. A practical estimation measurement for host load is the number of connections, and a measurement for network load is the number of messages.
6. *Request Redirection*. The system should provide a function for clients or networks to switch the direction of requests to an appropriate surrogate server dynamically. It is because the number and allocation of surrogate servers, as well as the (estimated) load of each surrogate, change dynamically. Some of the practical methods for redirection are using browser cookies, switching DNS records at the client side, and applying anycast infrastructure.

Recently, a technology named "Software Defined Networking" (SDN) [32] has emerged, and is now attracting much attention in the field of Internet routing. The technology introduces a programmable router that can change its routing behavior

dynamically according to its programs. Therefore, if we would install a program on an SDN router to change the destination of network packets different from the original destination, the packets could be redirected. Actually, it has been reported that SDN can be used to implement anycast in a more flexible manner than before [33]. SDN-based request redirection for dynamic CDN must be a promising research topic to explore as well.

Actually, there is no best method for estimation and redirection practically, and system designers must choose any appropriate method according to individual requirements and constraints.

13.6 CONCLUSION

This chapter presents a dynamic CDN for streaming. It is an extension to a CDN for static contents and optimizes its organization and size adaptively to a varying amount of requests from clients. The system comprises techniques for dynamic network reorganization and for load distribution and balancing to realize dynamicity, as well as techniques for stream segmentation and reconstruction and for QoS assurance. Live streaming and on-demand streaming require different treatments, which are also illustrated using a system prototype example. Although the example shows a success in adaptive optimization to the varying load, much space still remains to explore and investigate, as discussed in Section 13.5. In particular, HLS is the only established solution to stream segmentation and reconstruction at this moment, and any improvement and sophistication to the technology will bring a wider perspective to streaming CDNs. It is expected that continual advancement of the related technologies will address the future issues and pave the way for smooth deployment in real-world applications.

ACKNOWLEDGMENTS

This chapter is partially based on the author's joint works with Mr. Yuta Miyauchi, Mr. Masaya Miyashita, Prof. Noriko Matsumoto (Saitama University, Japan), Dr. Yuko Kamiya, and Prof. Toshihiko Shimokawa (Kyushu Sangyo University, Japan).

REFERENCES

1. Niven L. Flash Crowd. In: *The Flight of the Horse*. Ballantine Books; 1973. p 99–164.
2. Jung J, Krishnamurthy B, Rabinovich M. Flash crowds and denial of service attacks: Characterization and mplications for CDNs and web sites. Proceedings of 11th International World Wide Web Conference; 2002. p 252–262.
3. Zhao W, Schulzrinne H. DotSlash: A self-configuring and scalable rescue system for handling web hotspots effectively. Proceedings of International Workshop on Web Caching and Content Distribution; 2004. p 1–18.

4. Ari I, Hong B, Miller EL, Brandt SA, Long DE. Managing flash crowds on the Internet. Proceedings of 11th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems; 2003. p 246–249.
5. Stading T, Maniatis P, Baker M. Peer-to-peer caching schemes to address flash crowds. Proceedings of 1st International Workshop on Peer-to-Peer Systems; 2002. p 203–213.
6. Freedman MJ, Freudenthal E, Mazieres D. Democratizing content publication with coral. Proceedings of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation; 2004.
7. Padmanabhan VN, Sripanidkulchai K. The case for cooperative networking. Proceedings of 1st International Workshop on Peer-to-Peer Systems; 2002. p 178–190.
8. Stavrou A, Rubenstein D, Sahu S. A lightweight, robust P2P system to handle flash crowds. IEEE J Select Areas Commun 2002;22:6–17.
9. Pan C, Atajanov M, Hossain MB, Shimokawa T, Yoshida N. FCAN: Flash crowds alleviation network using adaptive P2P overlay of cache proxies. IEICE Tr Comm 2006;E89-B(4):1119–1126.
10. Yoshida N. Dynamic CDN against Flash Crowds. In: Buyya R, Pathan A-MK, Vakali A, editors. *Content Delivery Networks*. Springer; 2008. p 277–298.
11. Miyauchi Y, Matsumoto N, Yoshida N, Shimokawa T. Preliminary study on world-wide implementation of adaptive content distribution network. Proceedings of Workshop on Self-Organising, Adaptive, Context-Sensitive Distributive System; 2011. 11 pages.
12. Awadallah AA, Rosenblum M. The vMatrix: A network of virtual machine monitors for dynamic content distribution. Proceedings of 7th International Workshop on Web Content Caching and Distribution; 2002.
13. Kotsovinos E, Moreton T, Pratt I, Ross P, Fraser K, Hand S, Harris T. Global-scale service deployment in the XenoServer platform. Proceedings of 1st Workshop on Real, Large Distributive Systems; 2004.
14. Kamiya Y, Shimokawa T, Tanizaki F, Yoshida N. Scalable contents delivery system with dynamic server deployment. Int J Comp Sci Issues 2010;7(6):81–85.
15. Kamiya Y, Shimokawa T, Tanizaki F, Yoshida N. Dynamic wide area server deployment system with server deployment policies. Int J Comp Sci Netw Sec 2010;10(10):92–96.
16. Chess D, Harrison C, Kershenbaum A, Watson TJ. *Mobile Agents: Are They a Good Idea?, Mobile Object Systems Towards the Programmable Internet, LNCS 1222*. Springer; 1997.
17. Lange DB, Oshima M. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley; 1998.
18. Shudo K, Muraoka Y. Asynchronous migration of execution context in Java virtual machines. Fut Gen Comput Syst 2001;18(2):225–233.
19. Satoh I. A mobile agent-based framework for active networks. Proceedings of IEEE Conference on Systems, Man, and Cybernetics 1999; 1999; 2. p 71–76.
20. Sekiguchi T, Masuhara H, Yonezawa A. *A Simple Extension of Java Language for Controllable Transparent Migration and Its Portable Implementation, Coordination Models and Languages*. Springer; 1999. p 211–226.
21. Thitikamol K, Keleher P. Thread migration and load balancing in non-dedicated environments. Proceedings of 14th International Parallel and Distributed Processing Symposium; 2000. p 583–588.
22. Hai J, Chaudhary V. MigThread: Thread migration in DSM systems. Proceedings of International Conference on Parallel Processing; 2002. p 581–588.

23. Chen P, Chang J, Liang T, Shieh C, Zhuang Y. A multi-layer resource reconfiguration framework for grid computing. Proceedings of 4th International Workshop on Middleware for Grid Computing; 2006. 13 pages.
24. Miyashita M, Haque ME, Matsumoto N, Yoshida N. Dynamic load distribution in grid using mobile threads. Proceedings of IEEE 3rd International Workshop on Internet and Distributive Computing Systems; 2010. p 629–634.
25. Apple Developer. 2011. HTTP live streaming. Available at http://developer.apple.com/resources/http-streaming/. Accessed 18 August 2013.
26. Pantos R. 2011. IETF Internet draft: HTTP live streaming. Available at http://tools.ietf.org/html/draft-pantos-http-live-streaming. Accessed 17 August 2013.
27. Miyauchi Y, Matsumoto N, Yoshida N, Kamiya Y, Shimokawa T. Adaptive content distribution network for live and on-demand streaming. Proceedings of Workshop on Architecture for Self-Organizing Private IT-Spheres; 2012. p 27–37.
28. Apple Developer. 2011. Bip Bop All. Available at http://devimages.apple.com/iphone/samples/bipbopall.html. Accessed 18 August 2013.
29. VMware. 2013. Available at http://www.vmware.com/. Accessed 17 August 2013.
30. Xen Project. 2013. Available at http://www.xenproject.org/. Accessed 17 August 2013.
31. Abboud O, Pussep K, Kovacevic A, Mohr K, Kaune S, Steinmetz R. Enabling resilient P2P video streaming: Survey and analysis. Multimedia Syst 2011 Springer;17(3):177–197.
32. Software Defined Networking. 2013. Available at https://www.opennetworking.org/. Accessed 18 August 2013.
33. Othman OMM, Okamura K. Design and implementation of application based routing using OpenFlow. Proceedings of ACM 5th International Conference on Future Internet Technologies; 2010. p 60–67.