Autonomic Multi-Server Distribution in Flash Crowds Alleviation Network

Merdan Atajanov
Graduate School of Science and Engineering
Saitama University
Saitama 338-8570, Japan

Toshihiko Shimokawa
Faculty of Information Science
Kyushu Sangyo University
Fukuoka 813-8503, Japan

Norihiko Yoshida
Graduate School of Science and Engineering
Saitama University
Saitama 338-8570, Japan

Abstract

The Flash crowds are rapid increase in access to contents of web sites, which makes the web sites inaccessible, leaving the clients with unsatisfied requests. The major shortcoming of flash crowds researches is that they do not assist vital resizing feature of a cloud of the surrogates; the surrogates involved in the alleviation process do not change from the start to the end of flash crowds. Our system, FCAN (Flash Crowds Alleviation Network) is a system to provide resources to web sites to overcome flash crowds. A main feature of FCAN is its dynamically resizing feature, which can adapt to request load of flash crowds by enlarging or shrinking a cloud of surrogate servers used by the web sites. In this paper, we present a new feature of FCAN to support multiple servers which experience different flash crowds simultaneously, and show experiment results with real web log data provided by Live Eclipse 2006.

keywords: Internet Load Distribution, Content Distribution Networks, Flash Crowds

1 Introduction

Even though the Internet capacity and network bandwidth emerged very rapidly these days, in some cases clients still experience problems while accessing the

web sites. These problems are slightly different from the problems that were in the early days of the Internet. These new problems involve network congestion, traffic bottleneck on the server side, which maybe caused by the overwhelming number of users simultaneously accessing web contents. This is called "flash crowds" phenomenon [1]. The best way to provide decent replies to the clients is to disseminate the requested contents as near as possible to the clients. Most of the corporations disseminate their web contents by implementing geographically distributed network of surrogate servers or by using services of the companies such as Akamai [2]. These companies distribute the load of highly hit web sites across a geographically dispersed network in advance. Their solution mainly focuses on using proprietary networks and caching centers to intercept and serve clients requests before the flash crowds occur.

As a new solution, we already introduced a system called FCAN (Flash Crowds Alleviation Network) [3, 4]. It utilizes cache proxies in the Internet as surrogates to form an anti-flash crowds system. Its advantageous feature over other anti-flash crowds systems is that FCAN has a dynamic nature: with its help, a cloud of surrogates can grow or shrink adapting to the changes in traffic coming to the surrogates. Some small subset of cache proxies is involved at the beginning, and the subset can grow or shrink adapting to the load changes.

Our previous works were done considering just single server situation, where only one server can benefit from FCAN system. We extend our FCAN so that it can handle several member servers experiencing flash crowds. We focus our attention on addressing multi server situation, when several servers experience flash crowds simultaneously. Our system handles several flash crowds simultaneously by splitting a network of cache proxies to the servers that experience the flash crowds. The cache proxy can be involved in several flash crowds events, however the system tries to keep cache proxies involvement only in one flash crowds' event. In this paper, we use real flash crowds' data to investigate how FCAN behaves more realistically than in [4]. First we provide simulation results with artificial data and then show results with real flash crowds' data.

The rest of the paper is organized as follows: Section 2 is Related Work section; in Section 3, we describe FCAN design. The simulation data and results are in Section 4. Section 5 is the conclusion, including some considerations and future work.

2 Related Work

The flash crowds event is a very recent phenomenon in the Internet. Mostly flash crowds are infrequent and unpredictable events: you can predict that flash crowds could happen, but it is almost impossible to predict their magnitude and duration. Related researches against flash crowds can be divided into three categories: server-based solutions, client-based solutions, and intermediate solutions.

The CDN (Content Distribution Networks) is one approach in server-layer solutions. The main idea is to distribute load expected for one server to several surrogate servers. The server-layer solution's biggest disadvantage is overestimation of resources; these resources are only used in the flash crowds' event. In most cases, a flash crowds event is unexpected or unpredictable, it takes short time; therefore providing extra resources is not a good solution to anticipate flash crowds. In the client-side solutions client transparency is gone which in most cases requires client cooperation. Moreover the client-side solutions are very difficult to manage. Related work against the flash crowds includes CoralCDN [5] and P2P-based systems such as Backslash [6] and PROOFS [7]. Main

characteristics of flash crowds are extensively studied in [3], these characteristics were used as the basis for designing FCAN system.

3 FCAN Design

3.1 Overview

FCAN is an intermediate layer solution, focusing on a CDN-like cloud of cache proxies where popular objects are cached in, and delivered to end users. Our proposal is to construct the CDN-like cloud of proxies where all proxies are static members. This cloud of CPs (cache proxies) is widely spread with lots of participants. When flash crowds occur, the member server chooses a subset of proxies to form its small cloud, which will be responsible for the popular objects. If the subset of CPs cannot handle the client requests, new proxies are invited and the cloud grows. When the flash crowds decrease, cloud of CPs shrinks.

The popular objects are stored in some kind of CDN-like cloud of reverse proxies. In these proxies, contents are stored for the duration that flash crowds last, and client requests that are supposed to go to the server, are delivered to these proxies serving as surrogates. In this way, the load on the main server is reduced, and more clients get satisfied replies. In the peaceful time, there is no need for the FCAN system to be active; it is only activated when flash crowds occur, and it helps servers until the flash crowds are over. DNS redirection is used to redirect client requests to the cache proxies.

FCAN consists of the below:

- A member server: is a server that suffers from the FC (flash crowds) event and wants to use system to overcome it.
- A FC object: is a main content that is requested in flash crowds' event.
- A Permanent proxy member: is a main proxy in the CDN-like cloud. It is divided into two subgroups:
  - A Core proxy member: is a proxy that is always responsible for a FC object, when the member server is in FC state.
  - A Free proxy member: is a proxy that dynamically joins and leaves the core part, and helps the core proxies when they are overloaded.

Every proxy can be a core or a free proxy, it can even be a core proxy for one member server and a free for another member server. The free proxies stay alert in case any of member servers begin to experience flash crowds. In that case some of the CPs switch to the core proxy state and help the member server.

3.2 Key Features of FCAN

The proxies generate and monitor such statistical information as request rates and loads for the FC objects. These statistics are passed to and processed by the member servers to monitor the overall state in alleviation procedure so that the member servers decide when to enlarge or shrink the cloud, and find out when the flash crowds are over.

All permanent proxies are defined beforehand and configured by the administrator of the system. These proxies form main CDN-like cloud of CPs, which is used by the member servers that suffer from flash crowds. Every permanent proxy should provide the following functions:

Table 1: Priority Table for the member servers

	SVR01	SVR02
CP0	70	90
CP1	20	50
CP2	10	100
CP3	30	30
CP4	90	20
CP5	60	40
CP6	100	10
CP7	50	80
CP8	40	70
CP9	80	60

- Change its state from a free proxy member to a core proxy member and vice versa
- Distribute permanent proxy database list
- Provide ability to store FC objects permanently
- Generate and monitor statistics of request rate and load
- Maintain cloud membership operations

The proxy has an ability to store FC objects permanently; these objects do not expire until FC event is over. The member server has the following functions:

- Disseminate FC object to the core proxies
- Manage own core set of proxies involved
- Trigger an update of DNS zone file
- Collect and process statistics generated by the core proxies

FCAN system uses a priority table in the selection mechanism of proxies. As it can be seen in the Table 1, the cache proxies are assigned different priorities for different member servers, where the least number has the highest priority. FCAN system assigns proxies to member servers according to these priorities.

The priorities are not sequential, but sparse. A new additional proxy can be easily inserted in between, or be removed from the table without a need to update priorities.

The core proxies' IP addresses are added to the DNS record of the member server, so that requests are redirected to the core proxies. While DNS update propagates, all the requests are still going to the member server. The member server can act beforehand, by triggering DNS update before state change of the FCAN system. This way DNS propagation will catch up sooner. Load distribution is done by round-robin DNS, or it may be done by some other advanced selection mechanism[8].

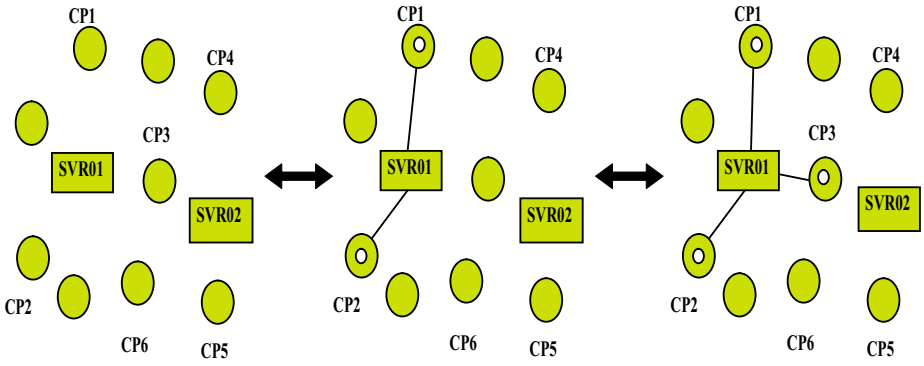

Figure 1: FCAN Outline

3.3 Overall View

At first, the system is in a peaceful state in which there are no flash crowds as it is shown in the left of Figure 1, and FCAN is in an inactive mode. When flash crowds come, a system forms a CDN-like cloud of core proxies, which act as surrogates for the member server as shown in the middle of Figure 1. If the cloud of surrogates cannot handle increasing amount of requests for the FC object, the member server invites more free proxies to participate in the cloud as shown in the right of Figure 1.

The member server selects potential core proxies among the proxies by sending check requests to these proxies. First, it probes a proxy which has the highest priority for the member server, and then a proxy with the second highest priority, and so on. Eliminating proxies which are already used in other servers' flash crowds alleviation procedure. This way it will find the most appropriate proxies that can participate in the alleviation procedure. When potential candidates are selected, the member server triggers DNS update to include IP addresses of newly added proxies, and disseminates FC object(s) to the selected proxies. In Figure 1, the initial core cloud consists of two proxies, CP1 and CP2.

3.4 The Cloud Growth and Shrinkage

There are two thresholds used in the FCAN system: T_{high} and T_{low}.

- T_{high}: the request rate is close to critical or soon will be above the acceptable rate, so that the system switches its state to the CDN-like cloud of the cache proxies.
- T_{low}: request rate is low, so that the system switches its state back to the peaceful client/server state.

When a member server detects that the load exceeds T_{high}, the system becomes active. When the initial core cloud is not enough, a new proxy (CP3 in Figure 1) is added to already involved member proxies (CP1 and CP2), and CP3 becomes core proxy for SVR01. When the average load on the core proxies is below T_{low} for predefined duration, the member server concludes that FC event is over. So, at first, it dismisses only some of the core proxies, and updates DNS.

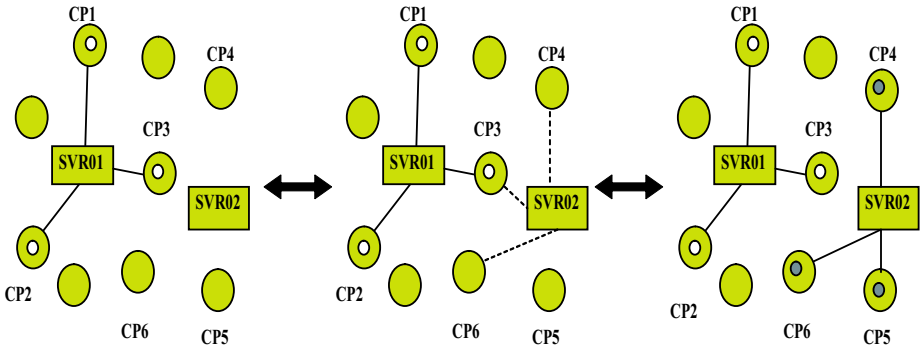

Figure 2: Multiple Member Servers in FCAN

The proxies with the lowest priority are dismissed first. Then, if the request rate at proxies is still under T_{low}, the member server dismisses some other core proxies. This process continues until all core proxies are dismissed. Firstly, the DNS records are updated, and the proxies wait for certain period until DNS propagation is supposedly finished. Then they change their state from core to free. The DNS redirection is done before dismiss, so as to prevent client requests being redirected to the proxy which does not have the flash crowds content, or which already changes its state to a normal forward proxy.

3.5 Multi-server Scenario

Cache proxies and member servers are independent from each other. FCAN is not designed for some specific member servers; it is designed to be used by several member servers in need at the same time. A member server can use a subset of proxies depending on the magnitude of flash crowds, and another member server can use another subset of the proxies. A proxy in the system can be a core proxy for one member server and a free proxy for another server. It is preferable that these subsets of the proxies used in different flash crowds do not overlap. To avoid overlapping, the system uses the priority table with predefined priorities for the member servers. When flash crowds comes, proxies are assigned to member servers according the priorities. If the proxy is already used in another flash crowds alleviation process, FCAN skips it and moves to a next proxy in the priority table.

In Figure 2, suppose that SVR02 also starts to experience the flash crowds event. It needs to construct its own cloud. Let assume that flash crowds magnitude of SVR02 is higher than SVR01, therefore the system invites three proxies at the beginning. From the table 1, we see that initially CP3 is overlapping between SVR01's core proxies cloud and SVR02's. However CP3 is already used in alleviation process for SVR01's flash crowds. So, when SVR02 probes CP3, it finds out that CP3 is busy, so it leaves CP3 and invites a next proxy from the priority table, which is CP5. After the initial core cloud is defined, SVR02 disseminates contents to its core members. These measures are done to avoid overlapping of the core proxy clouds.

In case where all proxies are used up in the alleviation procedure, some member server should share some of the proxies among each other. The proxies

Table 2: Simulation with Artificial Input

Servers	Start (sec)	End (sec)	CPs used	Initial Set	T_{high} (req/sec)	T_{low} (req/sec)
SVR01	55	250	8,3,2	8,3	40	5
SVR02	45	175	0,6,4	0	30	5

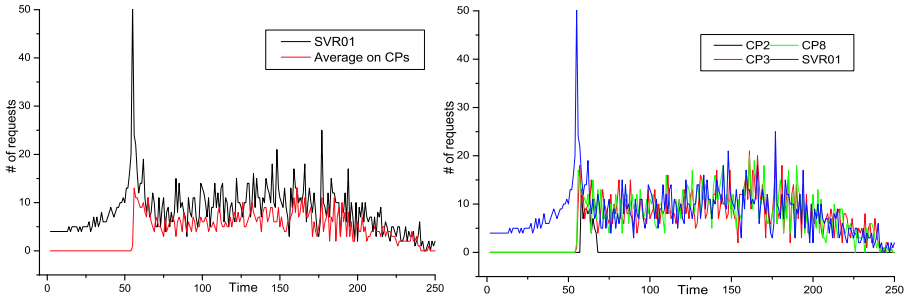

Figure 3: SVR01 with artificial input

that are shared among member servers will divide local capacity per server to be able to handle the requests for several flash crowds' objects.

4 Simulation and results

Our simulation has three roles: member servers, member proxies, and clients; each of them runs as independent thread. It is based on previous simulation of FCAN project. The network is built on an application layer with servers and proxies running continuously and concurrently. Clients' threads are created to send the requests, and then destroyed after getting the replies. Either a server or a proxy rejects the incoming requests if the average load exceeds its capacity. When the traffic is increasing, a member server switches the system to CDN-like cloud of proxies. If the initial core proxies are not enough in the alleviation procedure, the member server invites new proxies, until the load is stabilized. When the load is decreasing, the proxies begin to leave the cloud and a member server is switched back to the normal state.

Table 2 summarizes the configuration for the simulation of servers SVR01 and SVR02 with artificial input. Figures 3 and 4 present the results of this simulation. The left graphs in Figures 3 and 4 show average load, while the right graphs show individual loads on the proxies and member servers. The flash crowds for SVR01 start at 55th second and continue until the end of simulation. The flash crowds for SVR02 start at 45th second and end at 175th second. The magnitudes of two flash crowds are different; the magnitude of SVR01's flash crowds is bigger than the SVR02's. Three additional proxies CP8, CP3, CP2 are used till the end of the SVR01's flash crowds. On the other hand only CP0 is used till the end of the SVR02's flash crowds and CP6, CP4 is used just for 10 seconds at the beginning of the flash crowds.

Next, we investigate simulation results with real access logs, kindly provided

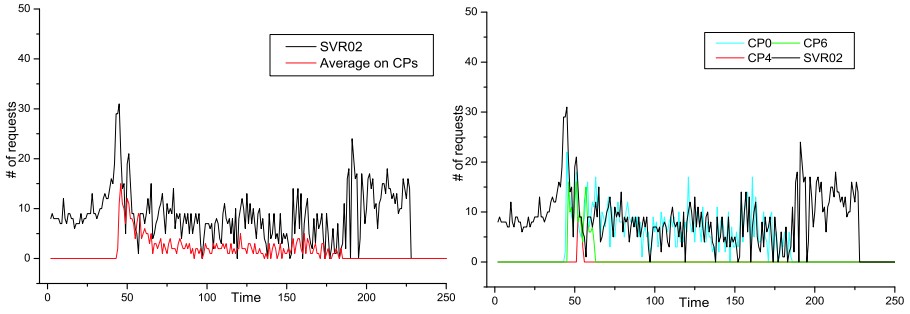

Figure 4: SVR02 with artificial input

Table 3: Simulation Configuration

Servers involved	2
The number of proxies	10
Proxy's priority order for SVR01	8,3,2,5,7,1,9,4,6,0
Initial set for SVR01	8,3
Proxy's priority order for SVR02	0,6,4,9,1,7,5,2,3,8
Initial set for SVR02	0
Threshold T_{high} (req/sec) for SVR01	30
Threshold T_{low} (req/sec) for SVR01	5
Threshold T_{high} (req/sec) for SVR02	20
Threshold T_{low} (req/sec) for SVR02	5
Server Capacity (req/sec) SVR01	40
Server Capacity (req/sec) SVR02	30

by "Live Eclipse" web site [9]. These live web logs are used as the input data for simulation with multi-server scenario. In March 2006, Live Eclipse delivered web streaming for the Eclipse that took place in Turkey, Libya and Egypt. Live Universe provided two different streaming sites:

- http://www.live-eclipse.org
- http://www.nishoku.jp

The first one was used by all the clients in the world, while the second one was just by Japanese clients. Therefore, there is a difference in access graphs for these two sites, as expected access rates for the live-eclipse is much higher than for the nishoku site. Figure 5 shows access rate patterns for these two sites at the time eclipse took place.

The logs for these two servers are scaled down so we can feed them into the simulation. The log of Live-Eclipse site is scaled down by 30, and log for Nishoku site by 10. Every simulation second corresponds to one minute of real time.

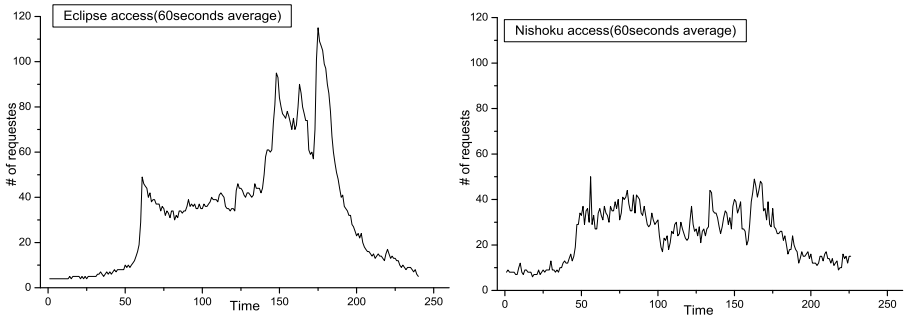

Figure 5: Access Pattern for "Live-Eclipse" and "Nishoku" Sites

In this experiment, two different member are used servers one for live-eclipse, another for nishoku and also ten permanent proxies are used for alleviation procedure. Table 3 summarizes the configuration of ten proxies and two independent member servers. The SVR01 and SVR02 experience flash crowds at the same time: SVR01 is fed the live-eclipse log, and SVR02 is fed the nishoku log. The priorities for these member servers are different, and initial core sets of proxies are different between the member servers according to their priority settings for the proxies and magnitudes of flash crowds. Before inviting free proxy to join the core cloud, a member server checks if free proxy is available, not used in other member server's flash crowds event. If it is already used by another member server, then a next free proxy in the priority table is probed for availability.

Figure 6 and 7 shows the results of the simulation with real data, where Figure 6 shows SVR01 "Live Eclipse" server's cloud and Figure 7 shows SVR02 "Nishoku" server's cloud. The left graphs in Figures 6 and 7 show average load, while the right graphs show individual loads.

For SVR01, seven proxies are used: two core proxies and five additional proxies. Initial set of the core proxies consists of two proxies CP8 and CP3. For the first 60 seconds the member server can handle the client requests itself. The SVR01's flash crowds start around 60th second. The member server invites initial core set of proxies to join in the alleviation process. At the beginning the requests grow so rapidly that initially assigned two proxies are not enough. Therefore third proxy CP2 is added to the system immediately. In this situation where three proxies and member server are involved, the system handles the load until the next rapid increase of flash crowds starting around 150th second. Four more proxies are added to the system in the following order: CP5, CP7, CP1, and CP9. With the help of these new additional proxies to the core part, the average load on the system is kept under the threshold Table 3. After 180th second, the client requests start decreasing very rapidly. At this moment, the system waits short duration to check if it is temporary change in the client requests. Since requests are decreasing at the steady rate, the system dismisses the proxies one by one until the system is switched back to the client/server mode. The state change occurs around 200th second.

In Figure 7, SVR02 uses two proxies, one is core proxy and the other is additional free proxy. Initial set consists of just one core proxy CP0. The SVR02's flash crowds start around 50th second, at this point, the highest peak of client requests is reached. Initially CP0 is added to the system, then immediately

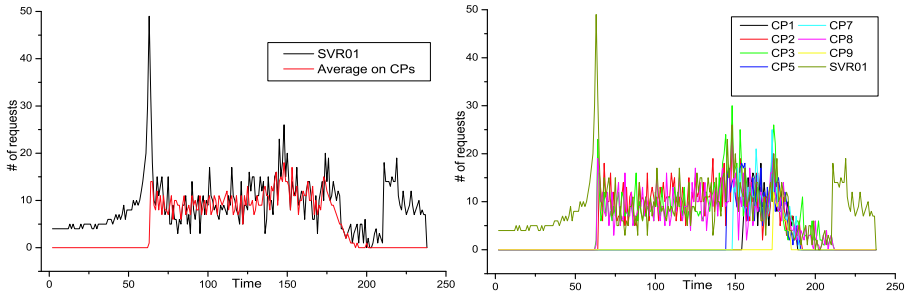

Figure 6: Access Log of "Live Eclipse" Proxy Cloud

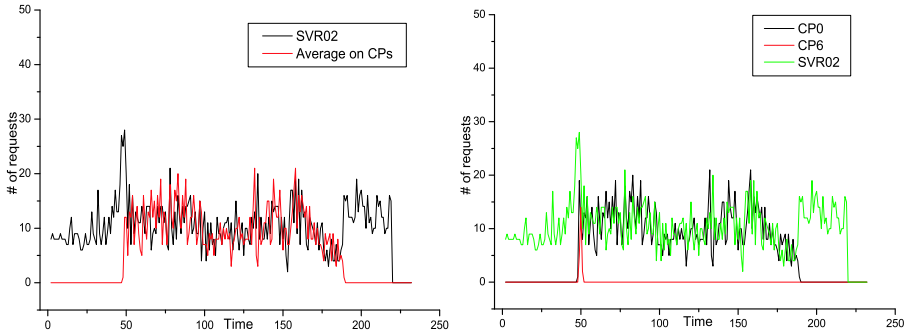

Figure 7: Access Log of "Nishoku" Proxy Cloud

CP6 is invited for 3 seconds. The CP6 is dismissed since the load is not high and can be handled by one core proxy and member server. In this situation where just one core proxy and member server are involved, the system handles all client requests until the end of the flash crowds. The SVR02's flash crowds ends around the 195th second. The member server dismisses core proxy CP0 and switches back to normal client/server mode.

5 Discussions

The flash crowds are very unpredictable events. Therefore is it very difficult to determine T_{low} and especially T_{high} thresholds. T_{low} should be as low as average load of the server in peaceful time. But the system will not switch itself to C/S mode, just by load dropping under T_{low}. The load should persist under T_{low} for predefined duration of time. T_{high} is defined as the rate that is reached in a short duration, and which cannot be handled by the member server itself. At the moment it is planned that CPs are volunteer proxy servers that are already functioning on the Internet. The priority assignment to individual CPs is done beforehand; priorities are in the way so that a proxy is not assigned high values

for serveral member servers. This way we reduce overlapping of the proxies involved in the alleviation procedure. The selection of CPs for a member server depends on the priorities assigned to proxies for that member server. Before assigning the proxy as a core proxy, the member server probes the proxy for availability and checks if it is not used by any other member server. In case the proxy is involved in another alleviation procedure, the member server skips to the next proxy in the priority table.

6 Conclusions

The strong point of FCAN is its dynamically resizing feature for the cloud of surrogates. This cloud can grow or shrink according to the load of the system. In this paper, we present FCAN's support for multiple servers simultaneously experiencing independent flash crowds. We investigate efficiency of this feature using real data which were kindly provided by Live Eclipse project. The simulation results showed that FCAN system is capable of handling multiple flash crowds at the same time.

In the Internet, dynamically generated contents have become more and more popular. We are planning to concentrate our attention on dynamic contents in flash crowds. The dynamic contents can be divided into two categories:

- Dynamically generated objects (using a backend database and scripts)
- Frequently updated contents

It is important to reduce the number of messages and amount of data exchanged in the network to reduce network congestion. In the flash crowds event, the network already will be overwhelmed by flash crowds object.

Now we are applying some techniques originally developed for "distributed shared memory", especially the "lazy release consistency" technique which was proved one of the most efficient [10].

Acknowledgments

This research was supported in part by MEXT in Japan under Grants-in-Aid for Scientific Research on Priority Area 18049009, and by JSPS in Japan under Grants-in-Aid for Scientific Research (B) 17300012.

References

[1] Flash Crowd phenomenon, http://en.wikipedia.org/wiki/Flash_Crowd

[2] Akamai, http://www.akamai.com

[3] C. Pan, M. Atajanov, M. B. Hossain, T. Shimokawa, and N. Yoshida, "FCAN: Flash Crowds Alleviation Network Using Adaptive P2P Overlay of Cache Proxies", IEICE Trans. on Communications, Vol.E89-B, No.4, pp.1119–1126, 2006.

[4] M. Atajanov, C. Pan, T. Shimokawa, and N. Yoshida, "Scalable Cloud of Cache Proxies for Flash Crowds Alleviation Network", Int'l Trans. on Communication and Signal Processing, Vol.8, No.1, pp.59–70, 2006.

[5] M. J. Freedman, E. Freudenthal, and D. Mazieres, "Democratizing Content Publication with Coral", Proc. 1st USENIX/ACM Symp. on Networked Systems Design and Implementation, 2004.

[6] T. Standing, P. Maniatis, and M. Baker, "Peer-to-Peer Caching Schemes to Address Flash Crowds", Proc. 1st Int'l Workshop on Peer-to-Peer Systems, pp.203–213, 2002.

[7] A. Starvrou, D. Rubenstein and S. Sahu, "A lighweight, robust P2P system to handle flash crowds", IEEE Journal on Selected Areas in Communications, vol.22, no.1 Jan. 2004.

[8] T. Shimokawa, N. Yoshida and K. Ushijima, "DNS-based Mechanism for Policy-added Server Selection", SSGRR2000: Int'l Conf. on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet July 2000

[9] Live Eclipse 2006, http://www.live-eclipse.org/index_e.html

[10] P. Keleher, A. L. Cox, and W. Zwaenepoel, "Lazy Release Consistency for Software Distributed Shared Memory", Proc. 19th Annual Int'l Symp. on Computer Architecture, 1992.