FLASH CROWD MITIGATION SYSTEM

by

DONGCHUL KIM

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2007

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Sajal Das for patience, support, guidance and for providing me an opportunity to work under him. I am very grateful to Prof. Kalyan Basu for his valuable advice. I would also like to thank Dr. Hao Che for being in my committee and reviewing thesis.

I would also like to thank to my colleague, Sumantra Kundu and Hyun Lee for their advices in my thesis works. I would like to extend my sincere thanks to all the members of CReWMaN for providing a motivational environment for research.

I would like to convey my sincerest gratitude to my wife for all her constant encouragement.

Finally, I thank God, our father, for all the talents and opportunities which he gave me. I can do all things, only through Christ who strengthens me (Philippians 4:13)

May 7, 2007

ABSTRACT

FLASH CROWD MITIGATION SYSTEM

Publication No. _____

Dongchul Kim, M.S.

The University of Texas at Arlington, 2007

Supervising Professor: Sajal K. Das

Flash crowd means that a web server suffers a sudden surge of traffic since a large number of Internet users access the web server simultaneously. Once a flash crowd occurs in a web server, the response rate for the HTTP requests decreases rapidly, or the web server may even crash. To protect the web server from such flash crowds, in this paper, we propose the Flash Crowd Mitigation System (FCMS) which is based on the cooperation of the web servers and the redirection of the HTTP request. In FCMS, when a flash crowd is predicted by the traffic monitoring module in the router which is connected to the web server, other web servers replicate the hot server's contents, and

then the router redirects the HTTP request to other web servers in order to alleviate the effect of the flash crowd. The simulation with real-world flash crowd traces shows that the monitoring module can predict the flash crowd and release the system from the flash crowd mode effectively. Also, our experiment in our testbed demonstrates that an implemented FCMS and our mechanism can work correctly and mitigate the effect of flash crowds in a web server.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Flash Crowd

Flash crowd means a phenomenon that a web server suffers a sudden surge of traffic as a large number of users on the Internet access the web server simultaneously.

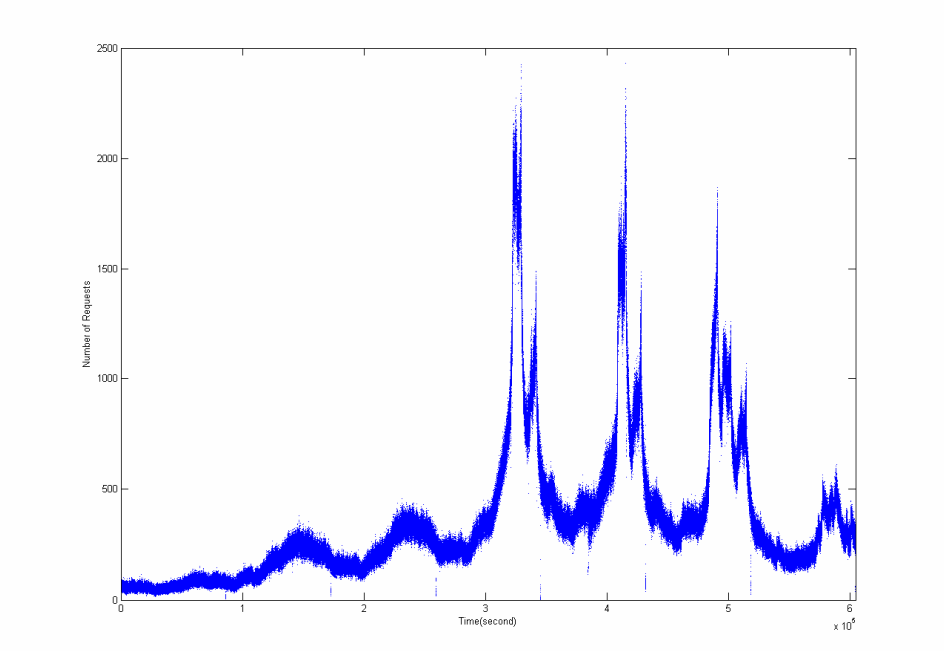

Figure 1.1 The example of flash crowd (1998 World-Cup Game homepage)

Generally, a flash crowd can be caused by the following. First, a number of people can access the related web site to get the information without any advanced warning when a specific event occurs. For example, when the 911 terror happened in United States, news media like www.cnn.com suffered a sharp increase in traffic and other sites experienced the similar situation also [1]. Second, if a server which has

relatively a low efficiency is linked to a web site which is remarkably popular, a flash crowd may happen. Generally, this case is called a slashdot effect. For instance, the web sites which were connected to the Google logo at www.google.com suffered flash crowds [2]. Third, a flash crowd would occur in a web site which puts an advertisement extensively in mass media without any advanced preparation for the high traffic. One example is a poorly planned marketing campaign by the company, Victoria's Secret [3]. Once a flash crowd occurs in a web server, the response rate for the Internet user's HTTP requests decreases rapidly, or the web server may crash.

1.2 Design Principles

In this paper, we propose a system for the mitigation of the flash crowd effect. Basically, when we designed the system, we have focused the three problems as follows. First, we have tried to find the solution for the small size servers and the "poor" servers which do not have the ability to prepare for the flash crowds with a pertinent budget unlike the commercial and popular web sites. In regard to this problem, initially, we have to distinguish the flash crowd which occurs in a short time from the overload which is shown continuously in a web server. In the case of the general overload, we know, the administrator of the server should deliberate a fundamental solution like the improvement of the related network resources or the employment of CDN like Akamai [4] rather than a temporary expedient. Contrary to the common overload in a web server, flash crowds rarely if ever happen, last for a short time relatively, or do not even happen even. Therefore, although we can apply the solutions mentioned above like using additional web servers having high performance, it would be wasteful for the poor

servers which do not have numerous traffic, much content, and abundant budget to use additional network resources for such flash crowds. For that reason, a solution for the poor servers is needed using only existing resources.

Second, we have considered how to define the point of time to decide the Flash Crowd, how to predict the Flash Crowd, and how to decide the point of time to release the system from the Flash Crowd. In this paper, we have applied the linear regression and moving average for the prediction and definition of the flash crowd as it is used in [5]. However, when we implement the real system using them, we recognized that the proper release time from the flash crowd mode is also important. Actually, through the simulation with real-world traces, when we used just the predefined threshold for the release point of time from the flash crowd mode, we recognized that the system repeats the unnecessary activation and deactivation of the system. Accordingly, the proper release time from the flash crowd mode is required in our system.

Third, HTTP redirection has been used to solve the surge of traffic in other research. In our work, we are also using redirection, but we are focusing the problem which can happen when the redirection is performed by the web server itself. To do HTTP redirection, TCP connection should be achieved in the transport layer. However, web servers have the limitation of number of TCP connections. Therefore, TCP connections may be failed first before getting the HTTP request due to the number of TCP connections and HTTP requests. To use the redirection correctly, this problem should be solved.

1.3 Flash Crowd Mitigation System

In this paper, we propose Flash Crowd Mitigation System (FCMS) which is focusing on the poor servers as we mentioned above. Basically, in FCMS, every server in the poor server group cooperates to rescue the web server which is suffering from overuse by the flash crowd. We call these poor web server the “member server”, and the web server which is suffering, the flash crowd, will be called the “target server” in this paper.

The basic operating flow of FCMS is as follows. If the end router connected to the member server predicts a flash crowd using linear regression and the threshold, then the router send the message in order to warn the flash crowd to the target server. After that, the target server informs the predicted flash crowd to other member server. If the member server receives the message from the target server, the member server can begin to replicate every object of the target server.

When replication in each member server is finished, the member server informs it to the target server. The router monitors the message between the member server and target server to know each server’s status. Once the router knows the member server has finished the replication, the router starts to redirect the HTTP request coming from the Internet. Finally, through these redirections, the Flash Crowd should be mitigated by reducing the number of requests for the target server.

1.4 Contributions

This work makes the following contribution. First, we designed FCMS which can be applied to the poor server without any kind of additional network resource.

Secondly, FCMS is using a more improved redirection mechanism using the virtual TCP connection which has an endless number of the connections. Also, through using more effective release time from the flash crowd mode, we can reduce unnecessary and repetitive activation of the system effectively. Lastly, this work involves the simulation of the monitoring module which is working in the router with real-world traces and the implementation of FCMS to evaluate its performance using the modified CLICK modular router, Java traffic generator, and several web servers as the member server.

CHAPTER 2

RELATED WORK

In this chapter, we categorize the related work and give an overview of the features of each solution.

2.1 Dropping

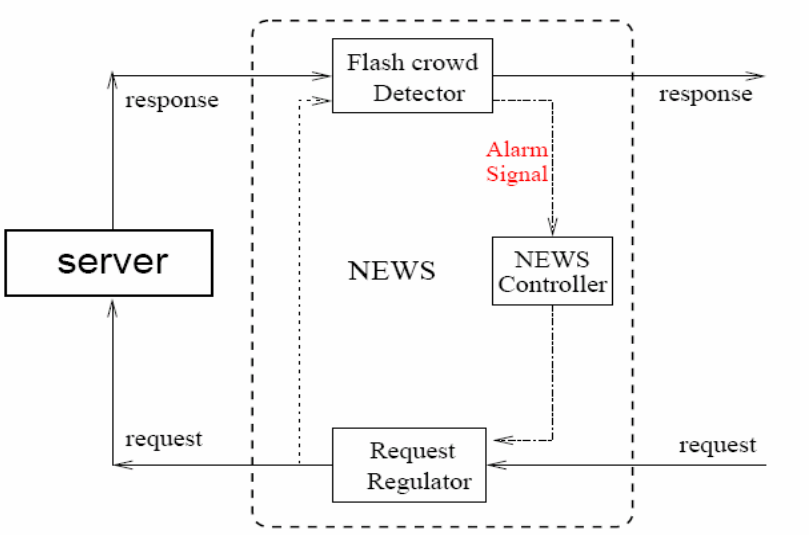

Figure 2.1 NEWS architecture [6]

Basically, these kinds of solutions [6, 7] allow the router to drop the HTTP request or SYN packet for the TCP connection which is coming from the Internet users in order to protect the web server from the flash crowd. In [6], they proposed the Network Early Warning System (NEWS), a router-based system. Once NEWS detects the flash crowd by discovering a decrease in response rate, a request regulator in Figure 2.1 controls the admitted request rate to mitigate the flash crowd. Although, however,

the server can survive the flash crowd, the service of the web server cannot be guaranteed, and the almost requests which are over the capacity of the web server will be dropped.

2.2 Peer to Peer

Recent studies for the flash crowd have tried to apply P2P schemes like [8, 9]. The basic idea of P2P schemes for the flash crowd is that, after a client retrieves the hot object from target server, the clients share the hot object without sending the request to the target server using client-side P2P overlay in order to alleviate the flash crowd in the target server. However, this solution requires the cooperation between Internet users. Practically, it is not feasible that every Internet user employ this P2P protocol for the flash crowd. In actuality, it is very questionable whether Internet users will provide their network resources to others for the flash crowd.

2.3 Contents Distributed Network (CDN), Cache Proxy, and HTTP Redirection

In [1], the author introduces adaptive CDNs. As shown in figure 2, DNS for adaptive CDNs distributes the request to the primary caches in adaptive CDNs. When the load on the primary cache reaches a dangerous level, the primary requests the DNS to start distributing requests to other members of the group called delegates. In [10], the authors proposed flash crowds alleviation network (FCAN) using cache proxy servers. Once the flash crowd happens, DNS returns the address of the cache proxy servers in balance. If the cache proxy server does not have the requested object, the server retrieves the object from the other proxy server or original server using the query on P2P overlay like figure 2.3.

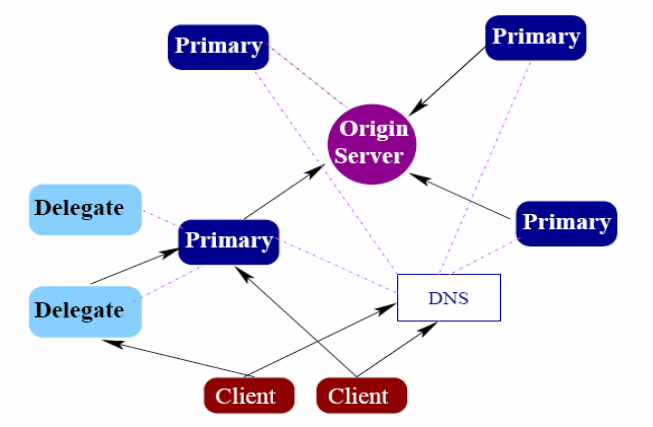

Figure 2.2 Architecture of an adaptive CDN [1]

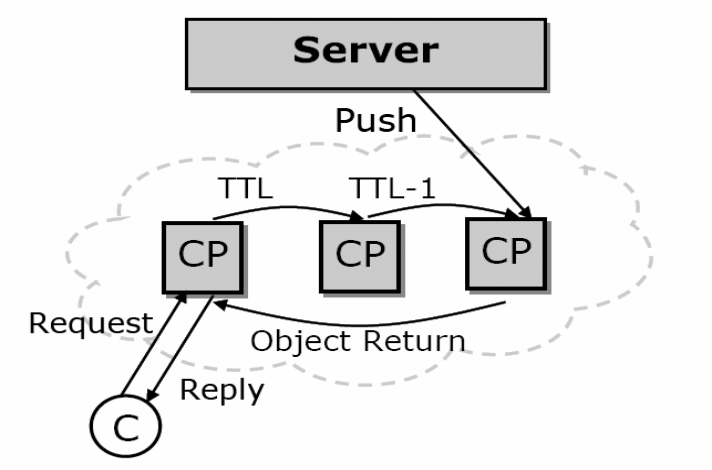

Figure 2.3 FCAN architecture [10]

In [11], what the author proposes is similar to our system using the redirection to the member server. As shown in figure 2.4, 2.5, they predefine the threshold to copy and redirect. Once the inter-arriving time of requests decreases below T_c (Threshold to copy), the web server asks other peer servers to pull the object in itself. If the traffic is over the redirection threshold, the web server can start to redirect the requests to peer servers.

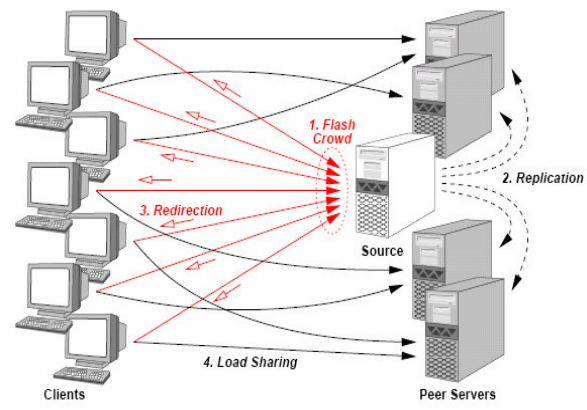

Figure 2.4 Principle of hot spot avoidance [11]

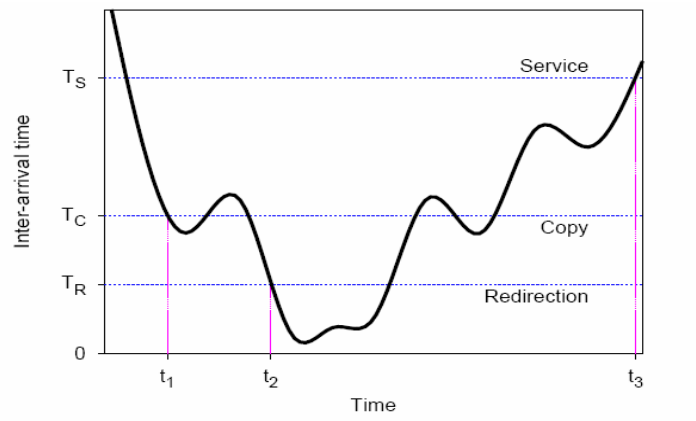

Figure 2.5 Configurable threshold [11]

However, potential problems exist in [11], and there are very important differences which can be represented by two features. First, although they have replicated mirror servers and the original server redirects the request to other peer servers, every request from users is still going to the original server. In this case, the web server should handle a large number of surge TCP connections and HTTP requests. In addition, the web server is more overloaded just before the flash crowd since the web server should perform the replication for the member server as well as traffic

monitoring to detect the flash crowd by itself. In FCMS, the traffic monitoring and the redirection are performed in the router to reduce the load in the target server and to protect the other end users from the flash crowds. Also, in FCMS, the proposed redirection which is working on the router has an endless number of TCP connections. Secondly, a point of time to replicate and redirect is not static like using the predefined threshold like figure 2.5. Of course, we are also using the threshold as the capacity of the web server for service in FCMS, but the time for copy and redirection is not static but adaptive through the prediction.

2.4 Flash Crowd Prediction

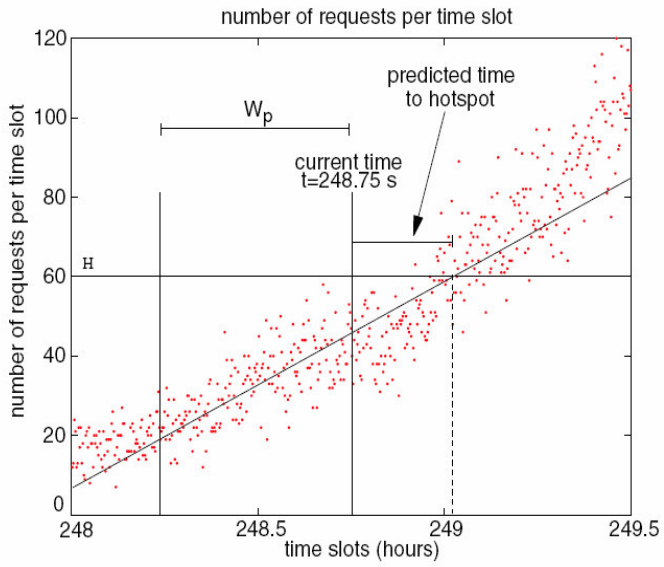

Figure 2.6 The definition of advance notice [5]

In [5], the authors apply linear regression to predict the flash crowd. As shown in figure 2.6, sample data between t – W_d (prediction window) and t is used for the linear regression, and the regression predicts the number of requests which will occur at

time $t + \tau$ as illustrated in figure 6. In FCMS, we also employ linear regression to predict the flash crowd, but our release time from the flash crowd mode is different so that FCMS can work more effectively. We will present this in detail in section 3.2.

CHAPTER 3

FLASH CROWD MITIGATION SYSTEM

In section 3.1, we will overview how FCMS is working and what is the preliminary assumption, and we will describe the monitoring module, object replication, and redirection as the main component of FCMS in 3.2, 3.3, and 3.4.

3.1 Overview

Basically, FCMS consists of a number of small size web servers as member servers which are located relatively close to each other, and each member server is connected to the end routers. Also, the application, Member Server Manager (MS-Manager), is working on each member server. MS-Manager existing in web server performs the communication, between the member servers, and manages the objects. The monitoring module and redirection module are embedded in each end router like figure 3.1.

The monitoring module observes the HTTP request traffic going to the member server in order to decide which the system has to set the flash crowd mode or the normal mode like figure 3.2. The goal of the redirection module is to redirect the HTTP request from the target server to other member servers using HTTP response code. We assume that we know the capacity of a web server as the value of the threshold in the monitoring module. In FCMS, the number of HTTP requests per second is used for that threshold. In addition, each member server has the object of one of other member

servers, and each member server knows what server has what server's object using the table like figure 3.14. In section 3.3, we will explain about this in detail.

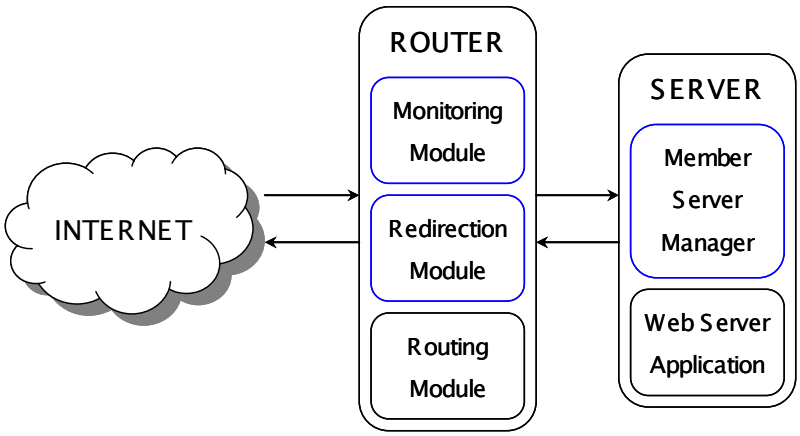

Figure 3.1 Monitoring module, redirection module, and member server manager

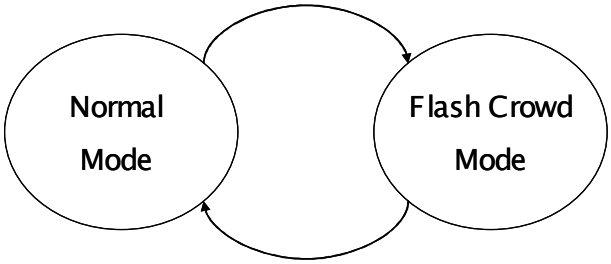

Figure 3.2 System modes

Figure 3.3, 3.4, and 3.5 shows how FCMS operates to mitigate the flash crowd effect. Initially, once a large number of Internet users sends the HTTP Request to one of the member servers simultaneously like figure 3.3, the router which is monitoring the traffic predict that the flash crowd will happen. After the prediction of the flash crowd, other member servers start to duplicate the objects of the target server from the member server which is keeping a copy of the objects. Then the router redirects the request to

other member servers which completes the duplication of the target server's contents like figure 3.4.

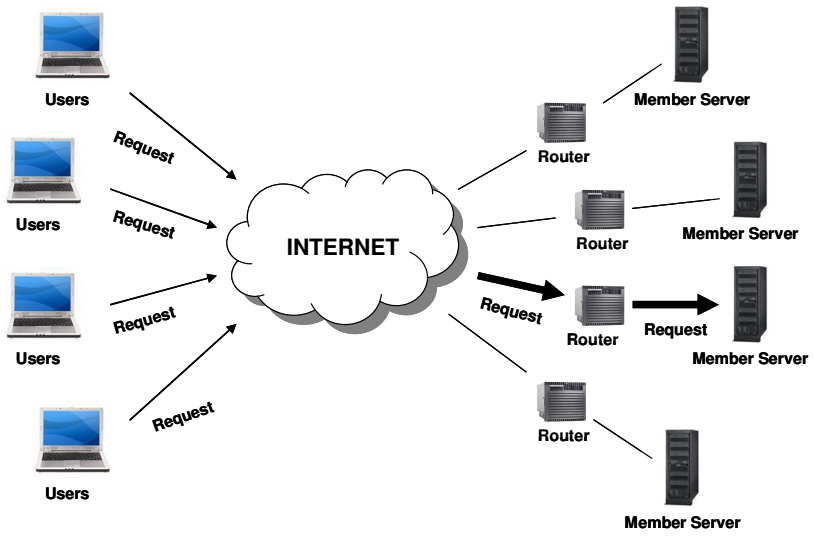

Figure 3.3 Surge of traffic

Each Internet user which received the redirection response sends the request again to the member server which is guided by the redirection response. As a result, through these redirections, the flash crowd in the target server can be mitigated like figure 3.5. Finally, when the traffic returns to the normal state, the router stops the redirection.

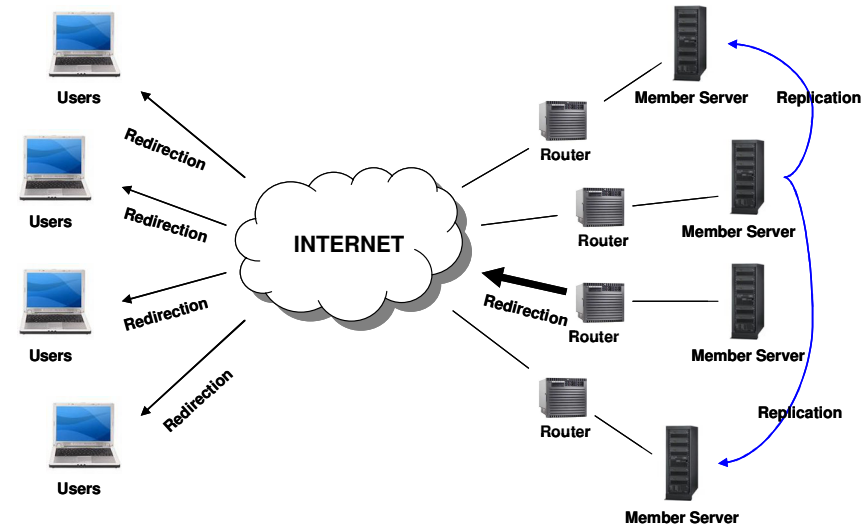

Figure 3.4 Replication and redirection

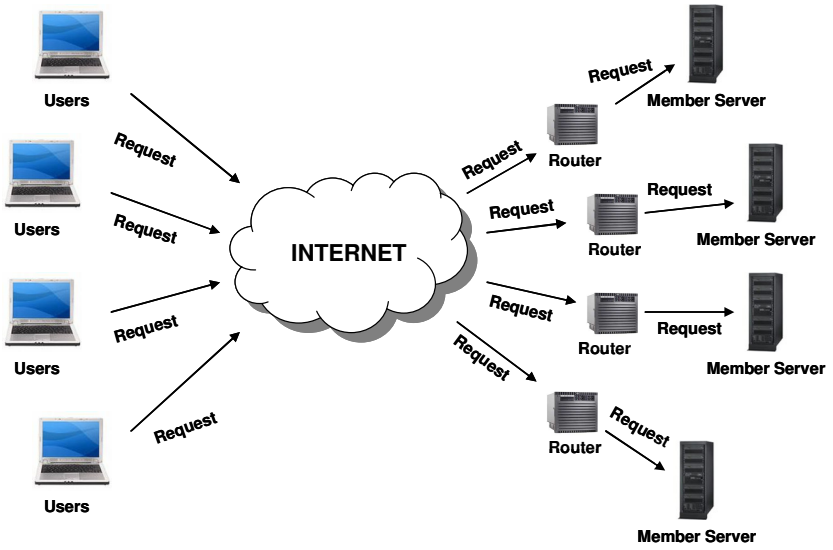

Figure 3.5 Distributed requests

3.2 Monitoring Module

3.2.1 Flash Crowd Prediction

As we mentioned in section 2, in [11], the authors suggested the linear regression for the prediction, and they defined the flash crowd status. Basically, we are

also using the linear regression to predict flash crowd and the definition of the flash crowd status. First, we define threshold H. This value is generally the maximum number of requests which can be processed by the web server. Simply, we can define the flash crowd when the number of request per second is more than H ($r_t \geq$ H). However, since there can be wide variation from sample to sample, the definition of the flash crowd require a smoothing of the data. Therefore, we determine if the current state is the flash crowd or not at time t using following condition [11]

$$\sum_{i \in [t-W_d, t]} r_i \geq H \times W_d$$

Also, we can use the average of the number of request during [t - W_d, t] $\geq$ H as the condition of the flash crowd. This means that the moving average has the term W_d and can be used to decide if the traffic at a particular time t is flash crowd or not like figure 3.6.

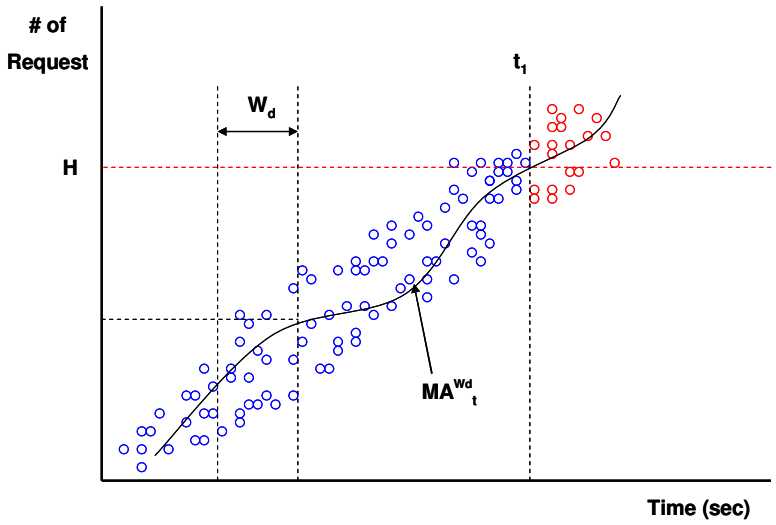

Figure 3.6 Moving average of traffic

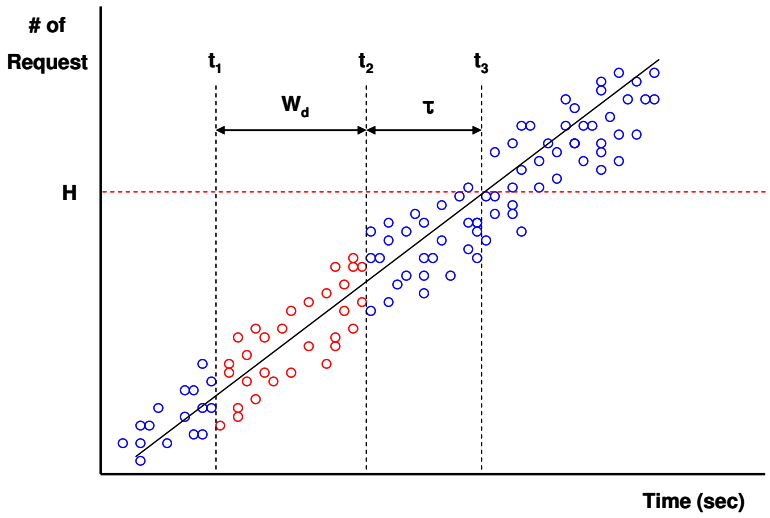

Figure 3.7 The definition of W_d and τ

As shown in figure 3.7, the time window W_d is a range of sample data for the prediction. The system predicts that the flash crowd will occur after time τ from the current time t_2. At current time t_2, to predict whether the traffic is over the threshold H or not, the prediction module in the router applies the linear regression to the sample data between t_1 and t_2. We can get the linear regression equation, $y = mx + b$, using the following:

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum(x^2) - (\sum x)^2} \qquad b = \frac{\sum y - m\sum x}{n}$$

And then, with value H, we can retrieve the value x as the predicted time when the flash crowd may happen. Finally, when this predicted time t_3 is the same as $t_2 + \tau$, the system sets the flash crowd mode. However, as [5] concludes, it is still very difficult to decide the two different parameters, the time window and the prediction time τ, since the nature of the Internet network traffic is different and flexible. In the case of FCMS,

we can use the time required to replicate whole objects for parameter τ , because the system should predict earlier considering the time for the duplication. We decide to use t x 2 as the constant window time after we did the experiment with a variety of different parameters. Although there is some error and it is not an optimal value, it should be enough to use for the FCMS, because we are not focusing on getting the parameter for linear regression in this paper.

3.2.2 Release from Flash Crowd Mode

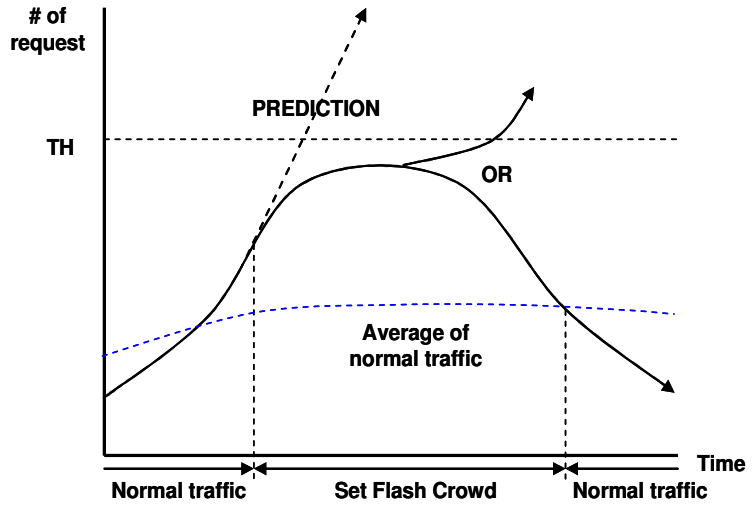

Figure 3.8 The traffic around threshold

Instead of finding the optimal parameters in linear regression, we have focused on the release time from the flash crowd mode since we met some problems while we simulated the prediction module which is using the release rules used in [5]. They release the flash crowd mode when the traffic is decreasing below the threshold or when the flash crowd predicted earlier does not occur after a fixed time τ max.

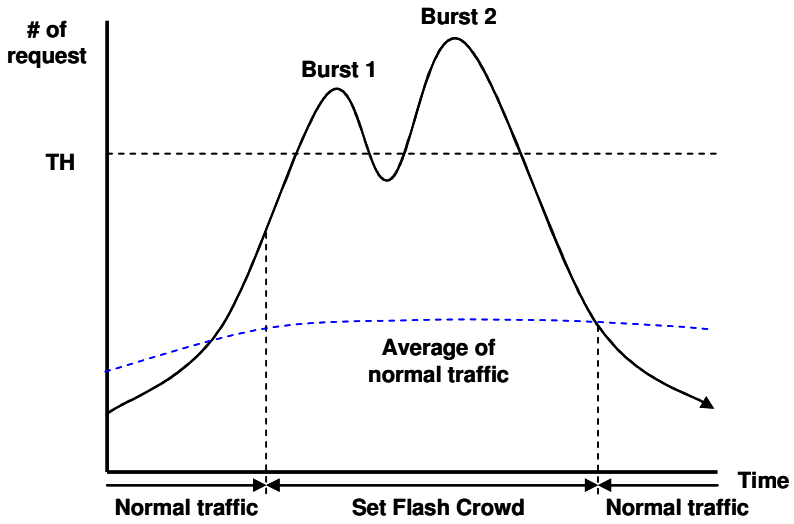

Figure 3.9 Two bursts in a flash crowd

In this case, there are two shortcomings. First, as shown in figure 3.8, once the system predicts that the flash crowd occurs after time t, the system set the flash crowd mode which means that the system is activated against the flash crowd. In this case, if flash crowd does not happen after time t, when should the system release the flash crowd mode? If the traffic stays the same near the threshold, the system will not need to release the flash crowd mode. Secondly, let's assume that the flash crowd mode is set by the prediction, and the traffic is increasing over the threshold. Some time later, the traffic will go down below the threshold. At that time, when should the system release the flash crowd mode? If the traffic increase again like figure 3.9, or the traffic is fluxuating around the threshold H, the system should start and stop the redirection. Therefore, only if the traffic decreases by a normal amount, the system should reset the flash crowd mode in order to avoid this superfluous repetition of the activation and deactivation. We define the traffic in the normal state as the average of the number of

requests which exclude the traffic occurring in the flash crowd like blue dotted line (the average of normal traffic) shown in figure 3.8, 3.9.

Like figure 3.8, 3.9, when the traffic decreases down below the average line, the flash crowd mode is released. In chapter 4, the experiment will show how well the system can detect the flash crowd with parameter $\tau \times 2$ and τ and how effectively the system can work to reduce the unnecessary activation of the system by using the average of normal traffic.

3.3 Object Replication

3.3.1 Member Server Manager (MS-Manager)

In FCMS, the application, MS-Manager, is operated in each member server in order to communicate between each member server and its adjacent router and also to perform the object replication. Basically, MS-Manager is Java network program using TCP socket. To inform the status of each member server, it is using four digit numbers for the protocol in the application level. In addition, when the monitoring module in the router sends the message to the member server, this protocol is used also. The details of protocol are as follows.

Table 3.1 MS-Manager Protocol

Protocol ID(last 2 digit)	**Status**
00	Normal State/Release FC
01	Flash Crowd
02	Copy
03	Complete the copy

3.3.1.1 MS-Manager Protocol

The message to communicate between MS-Managers consists of four digit number. The first two digits means member server ID as the source of the message and last two digits means the status of member server like Table 3.1. A member server sends the message to each member server once a second, and each member server and router maintain the table of status.

Figure 3.10 shows the MS-Manager protocol in the normal mode. Each member server sends xx00 (xx is member server's ID) to the other server, and nothing happen in this time. In Figure 3.10 MS (Member Server) 01 is sending 0100 to MS04.

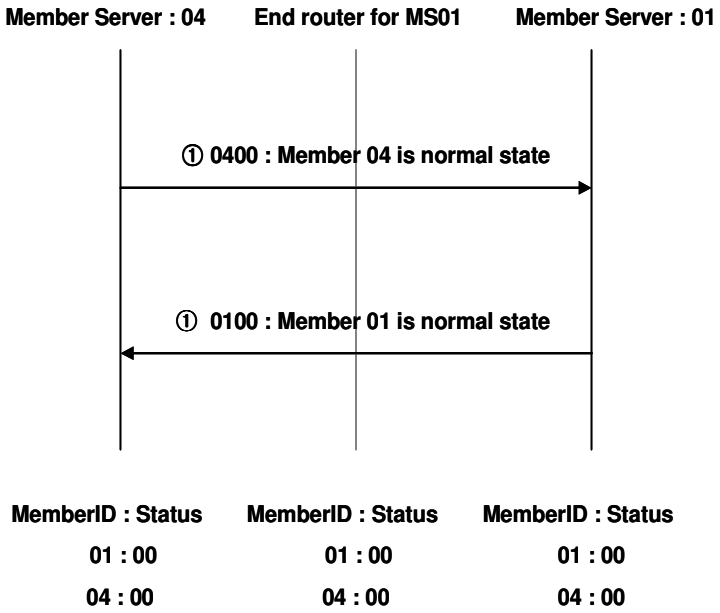

Figure 3.10 Normal state

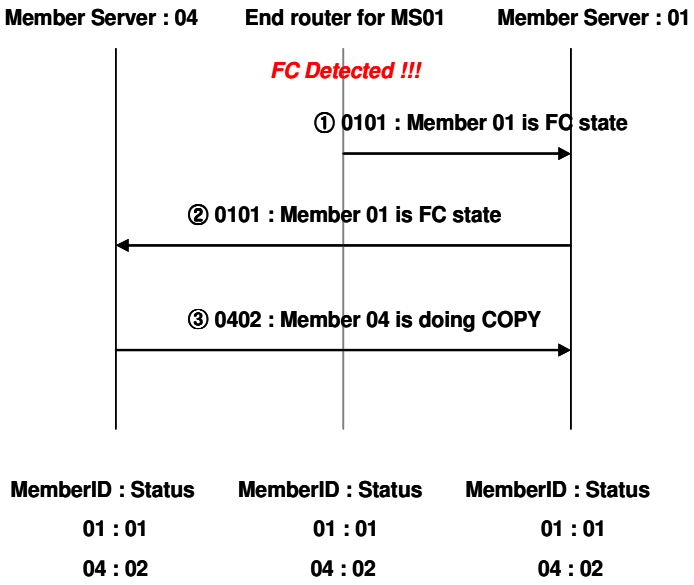

Figure 3.11 Flash crowd and replication

Figure 3.11 shows the MS-Manager protocol in the flash crowd mode. Once the router for MS01 detects the symptoms that the flash crowd will occur, the router sends 0101 to MS01. Through this message, MS01 can recognize the occurred FC in itself, and informs this to the other member server sending 0101. MS01 will send 0101 to others continually until flash crowd mode is released. Once the other MS receive the 0101, they will immediately start to replicate the objects and send message xx02 which means MSxx is copying now like figure 3.11. After each MS finish duplicating the MS01's objects, they will return the message xx03 until FC is released. Through the message xx03 from other MS, the router can know which MS is ready to receive the redirection. Therefore, the router executes the redirection for the request as soon as more than one MS send xx03 like figure 3.12.

Figure 3.13 shows the MS-Manager protocol for the release from flash crowd mode. Once the router decides the release from the flash crowd mode, the router send 0100 (normal state) to MS01 immediately. The MS01 stops to send 0101, and then sends 0100 to the other MS which are sending xx03 to MS01. Naturally, the other MS that received 0100 send xx00 to MS01.

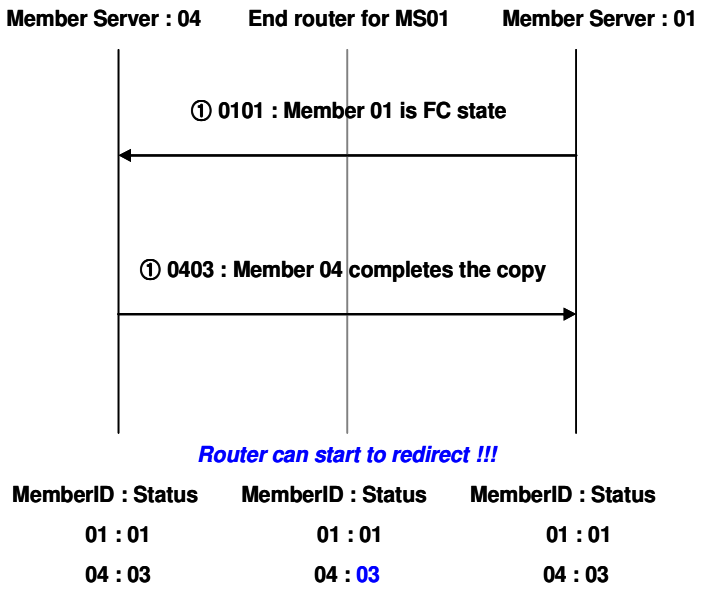

Figure 3.12 Completion of replication

Figure 3.13 shows the MS-Manager protocol for release from flash crowd mode. Once the router decides the release from the flash crowd mode, the router send 0100 (normal state) to MS01 immediately. The MS01 stops to send 0101, and then sends 0100 to other MS which is sending xx03 to MS01. Naturally, other MS received 0100 sends xx00 to MS01.

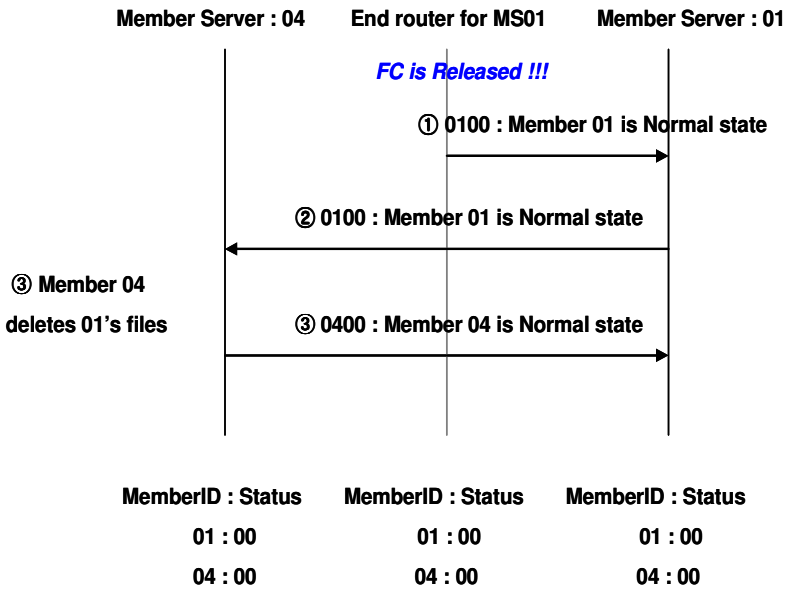

Figure 3.13 Release the system from flash crowd mode

3.3.1.2 Replication

As we mentioned in section 3.1, each member server should keep a ②copy of one of other member server's objects. Also, MS-Manager maintains the ③table indicating which server has which server's objects like figure 3.14. When the replication is started initially after the prediction, the member servers do not duplicate the target server's object from the target server directly but they duplicate the objects from other corresponding member server according to the table. The reason for this is that the direct replication can give another load to the target server. Also, the number of objects is not relatively sizable since we are focusing on small-size servers.

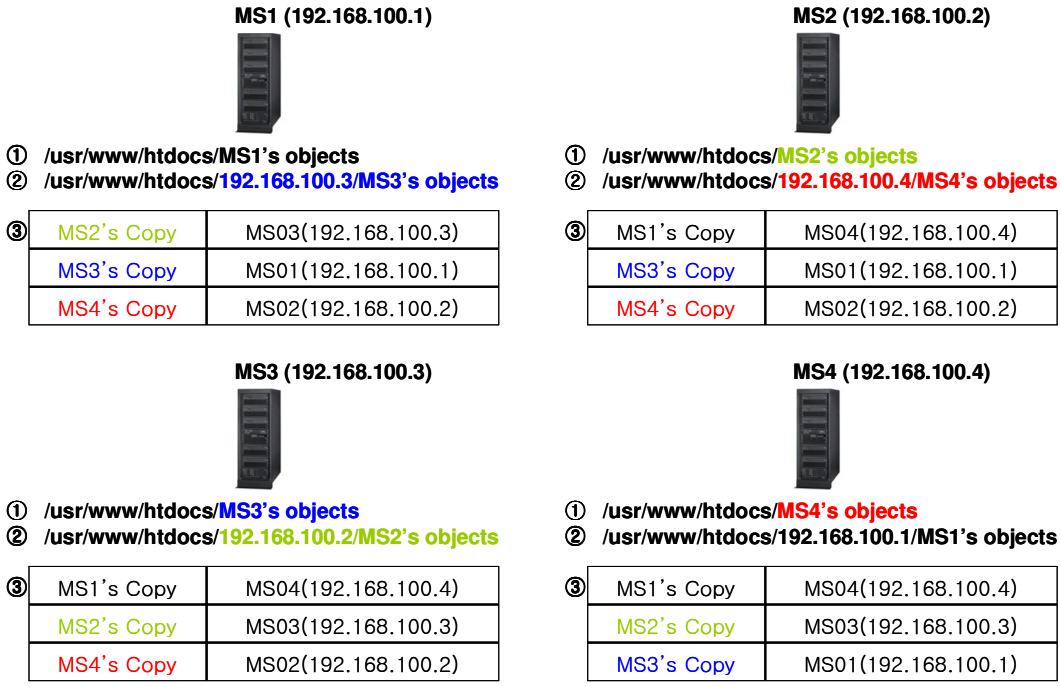

Figure 3.14 Table for replication

As shown in figure 3.15, 3.16, 3.17, and 3.18, the copy of the object is maintained under the directory which is named as owner's IP. If the original objects are changed, the administrator of the object should update the corresponding copy in other member server by himself. The list of the objects is recorded in the text file named as MAP. The member servers retrieve the MAP file first, and then they start the replication referencing the MAP file. In addition, once the replication is performed, every copied object is saved in TEMP directory. MS-Manager performs the replication with the following phases.

Phase 1 (Figure 3.15): Initially, once the target server sends the message 01 (flash crowd happened), the member server which has the copy of the target server changes the directory name to TEMP and broadcasts the message 03 (copy completed).

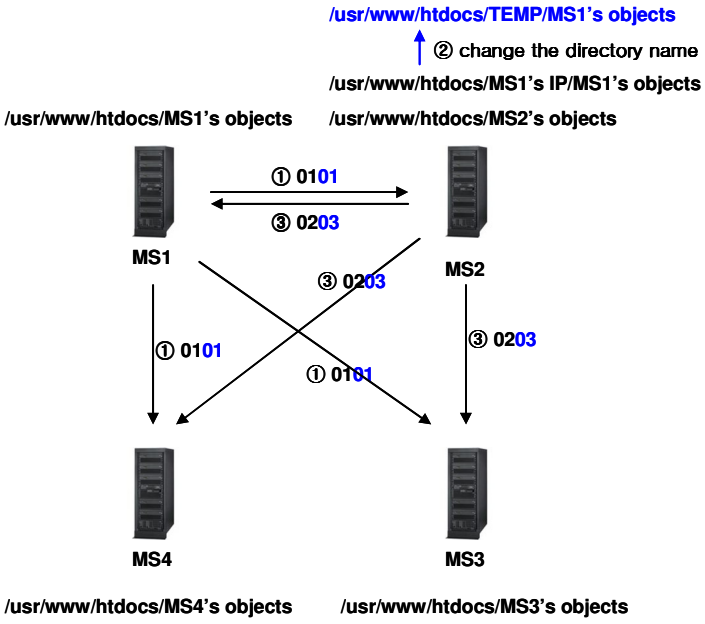

Figure 3.15 Changing directory name

Phase 2 (Figure3.16): The other member servers received the message 03 retrieve the MAP from the member server sending the message 03.

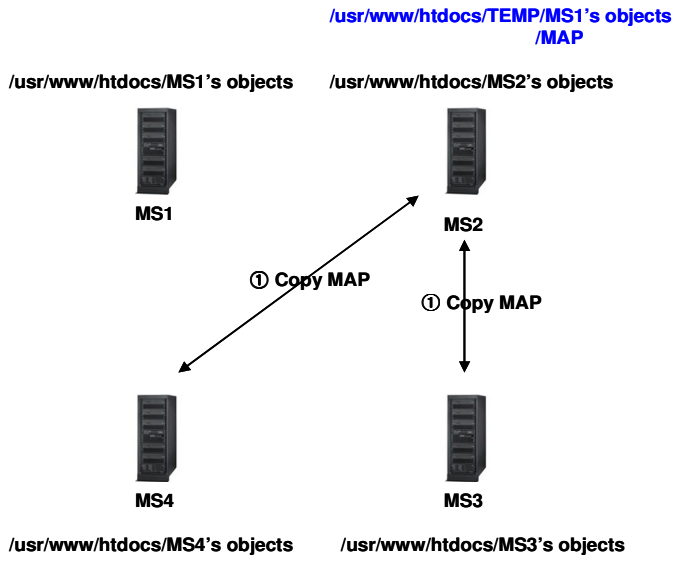

Figure 3.16 Retrieving MAP file

Phase 3 (Figure 3.17): The member servers start the replication with MAP and save the objects into TEMP directory.

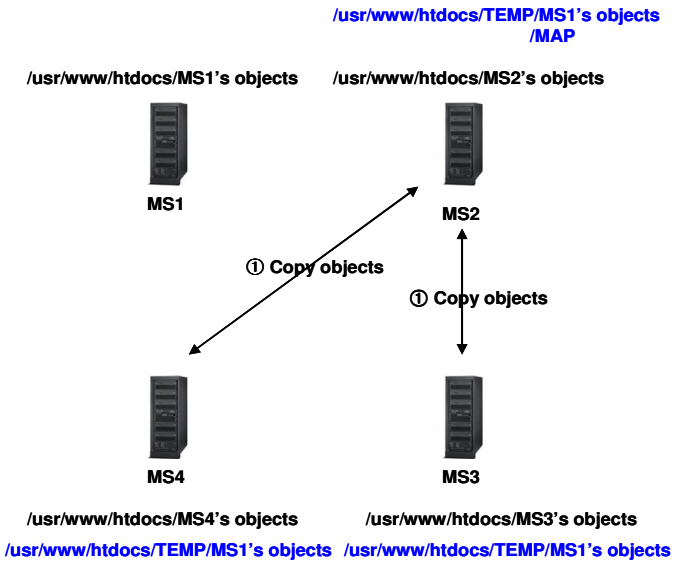

Figure 3.17 Creating TEMP directory

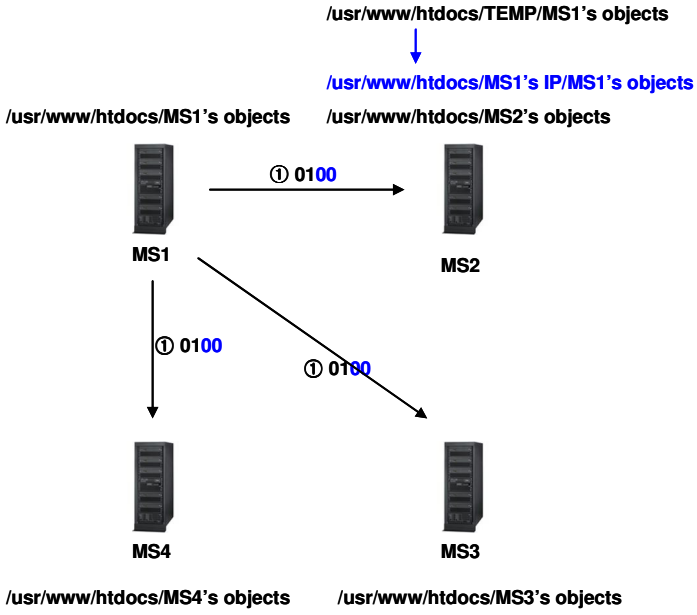

Figure 3.18 Changing directory name and removing temp directory

Phase 4 (Figure 3.18): If the member server gets the message 00 from the target server, the TEMP directory will be deleted and the member server which had kept the copy changes the name of the directory to the target server's IP again.

3.4 Redirection Module

If the web server redirects the huge number of HTTP request like [11] by itself, there can be the problem in TCP connection because the number of simultaneous TCP connection, which is provide by the web server, is limited. To solve this problem, we propose Virtual TCP Connection (VTC). Also, we use the HTTP response code 302 to redirect the HTTP request.

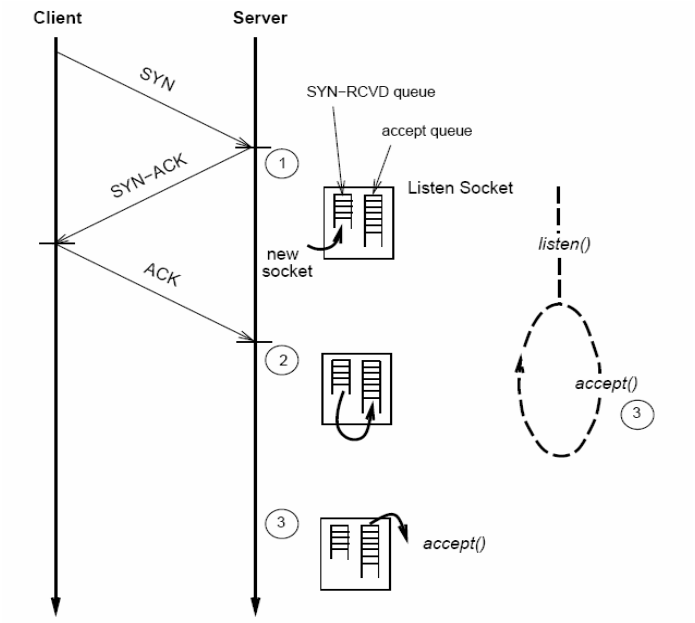

Figure 3.19 HTTP Connection Establishment Timeline [12]

3.4.1 The Limitation of TCP Connection in Web Server

Once an Internet user accesses the web server, the web browser establishes a new TCP connection, and then exchanges the HTTP request and response with that

connection. A web server executes the listen socket for the TCP connection, and the listen socket consists of SYN-RCVD queue and Accept queue. As shown in figure 3.19, when the SYN packet for a new connection comes from the browser, the server creates a new socket, returns SYN-ACK packet to the browser, and places the socket in SYN-RCVD queue. When the client sends the ACK packet for SYN-ACK sent by the server, the socket in SYN-RCVD queue is removed and placed in the accept queue. When web server process executes the accept() system call, the first socket in the accept queue of the listen socket is removed. After accepting a connection, the web server gets the request and sends back a response. In most Unix-based TCP/IP implementation, the kernel variable *somaxcom* defines the upper bound on the sum of the length of the SYN-RCVD queue and accept queue. The server drops incoming SYN packets whenever this sum exceeds a value of 1.5 times the variable. [12]

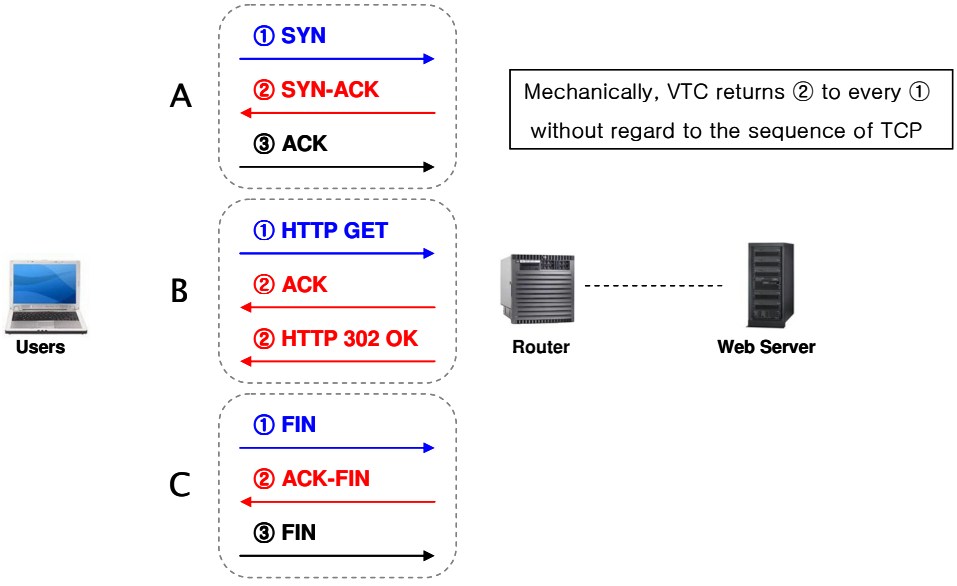

Figure 3.20 TCP connection and HTTP request in VTC

3.4.2 Virtual TCP Connection (VTC)

As we present above, the web server has the limitation to deal with a large number of TCP connection at the same time. To solve this problem, we have designed Virtual TCP Connections. Basically, VTC does not use any queue or management module like SYN-RCVD queue and the accept queue which can be found in a general web server. However, since VTC is designed for only flash crowd, the reliability, which is one of main features of TCP, is not guaranteed.

Simply, figure 3.20 shows the redirection module using VTC mechanically returns a SYN-ACK packet for every SYN packet which destination port number is 80. Also, VTC returns ACK and HTTP 302 for every HTTP GET which destination IP is target server's IP and returns ACK-FIN for every FIN packet having 80 as a destination port number without regard to TCP sequence. This means that the redirection module does not resend a SYN-ACK packet, even if ACK for SYN-ACK does not return back within timeout interval. Also, the redirection module sends ACK and Redirection HTTP response for every HTTP request without the preliminary TCP three-handshake connection. To do this, as if the web server does, the router has to generate and send the packet including appropriated source IP, destination IP, sequence number, acknowledge number, and so on.

As shown table 3.2, the following cases are about each possible scenario for the packet lost.

Table 3.2 Cases of packet lost

Lost A-①SYN	After timeout, the browser will send SYN to the web server again.
Lost A-②SYN-ACK	After timeout, the browser will send SYN to the web server again, and then The redirection module sends SYN-ACK to the browser again.
Lost A-③ACK	The redirection module does not care whether SYN-ACK arrived in the user side correctly or not.
Lost B-①HTTP GET	After timeout, the browser will send HTTP Request to the web server again.
Lost B-②ACK	After timeout, the browser will send HTTP Request to the web server again, and then the redirection module sends both of ACK and HTTP 302(redirection) to the browser again, even though HTTP 302 got through.
Lost B-②HTTP 302	
Lost C-①FIN	After timeout, the browser will send FIN to the web server again.
Lost C-②ACK-FIN	
Lost C-③ACK	The redirection module does not care whether the browser returns ACK for AC K-FIN or not.

3.4.3 Redirection of HTTP Request

As mentioned in section 3.4.2, basically, the redirection module working on the router redirects every HTTP requests having the target server's IP as the destination IP with the exception of the request having the member server's IP as the source IP.

In addition, the redirection module monitors the message which MS-Manager communicates to each other in order to know which member server completes the replication in flash crowd mode. Through this monitoring, therefore, the redirection module maintains the current status list of the member servers, and the redirection

module confirms if the member server which completes the replication exists on the status list before performing the redirection.

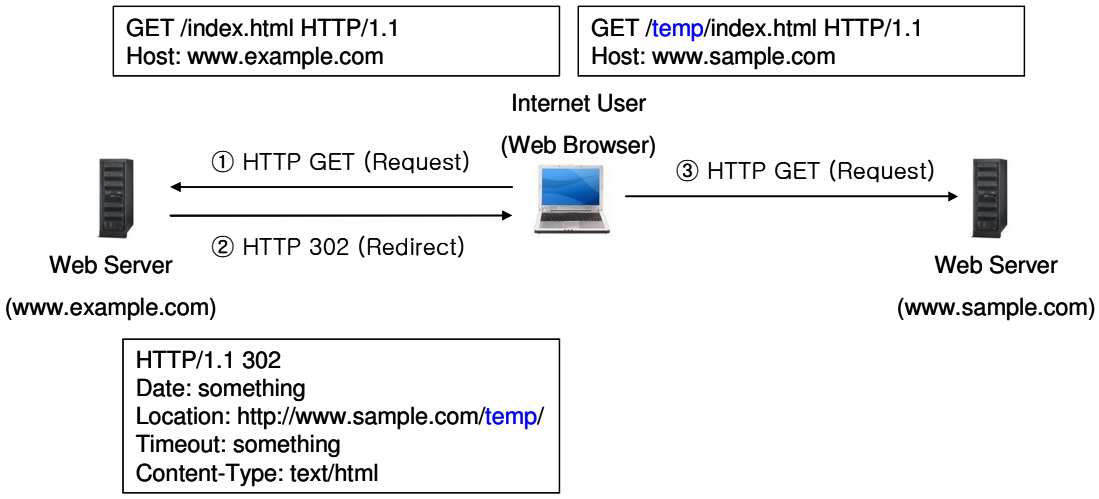

Figure 3.21 HTTP header for request and redirection

HTTP redirection is operated by using code 302 among the HTTP response code as in figure 3.21. According to HTTP protocol [RFC2616], response code 302 means that the requested resource resides temporarily under a different URI (Uniform Resource Identifiers).Therefore, if the web browser gets the response code 302, the web browser should send the HTTP request again to the location which the response message is indicating in order to get the objects.

When the flash crowd happens, the target server also processes 1/n (n: number of member servers) of the traffic like other member servers. Therefore, the redirection module does not redirect every HTTP request but passes 1/n of total requests. However, since the target server uses the origin TCP, unlike the redirection module which uses VTC, the redirection module should send the packets for the TCP connection which is

preprocessed for the HTTP request. To do this, therefore, the redirection module passes every packet which has the specific source port number defined by the modulus operator as illustrated in figure 3.22. For this way, we assume that the port number adopted by the Internet user's web browser is random.

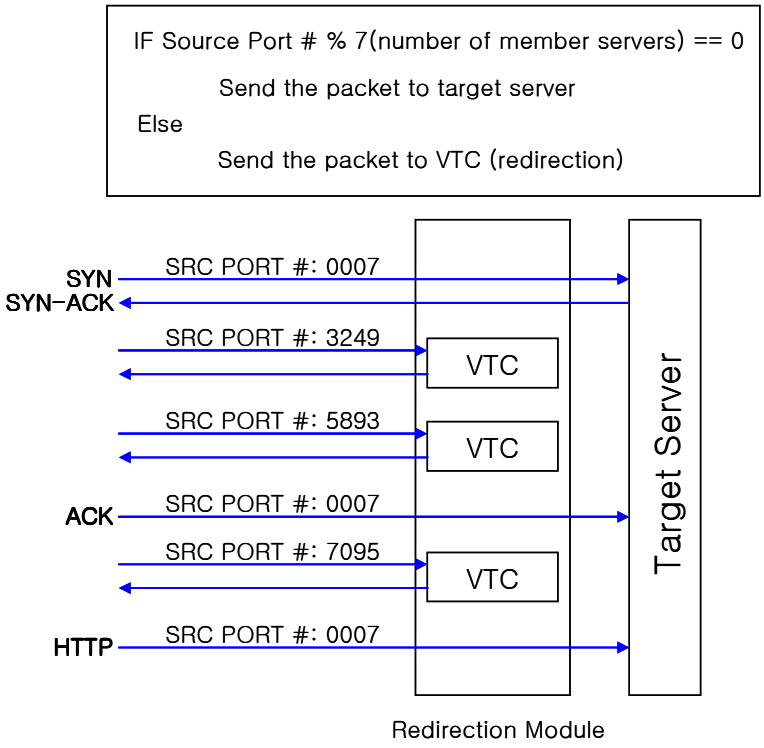

Figure 3.22 The exception of redirection

CHAPTER 4

SIMULATION FOR MONITORING MODULE

4.1 Methodology

4.1.1 World Cup

To evaluate the performance of the prediction and release in FCMS, we have done an experiment using the real-world traces for the access to 1998 World-Cup homepage [14]. Actually, we used the data for five weeks (42^{nd} ~ 76^{th} day of game period), in which the FC had occurred intensively, and we transformed the data set from the web access log file to the number of request per second. Although the World-Cup site is not small in size, the reason to use the World-Cup traces is that it is the most universal traces in FC research domain and it is more suitable to demonstrate the importance of release time because multiple peaks are shown like figure 1.1, compared to other flash crowd.

4.1.2 Criterion for Evaluation

The performance of prediction in FCMS depends on the time difference between the predicted time and the real flash crowd happened time. Ideally, if time T is a prediction time and flash crowd happens earlier than time T, the prediction will be a late alarm. In the other way, if the flash crowd happens later than time T, the prediction will be an early alarm. However, as it is very difficult to predict for the exact time, there should be a tolerance for the prediction time.

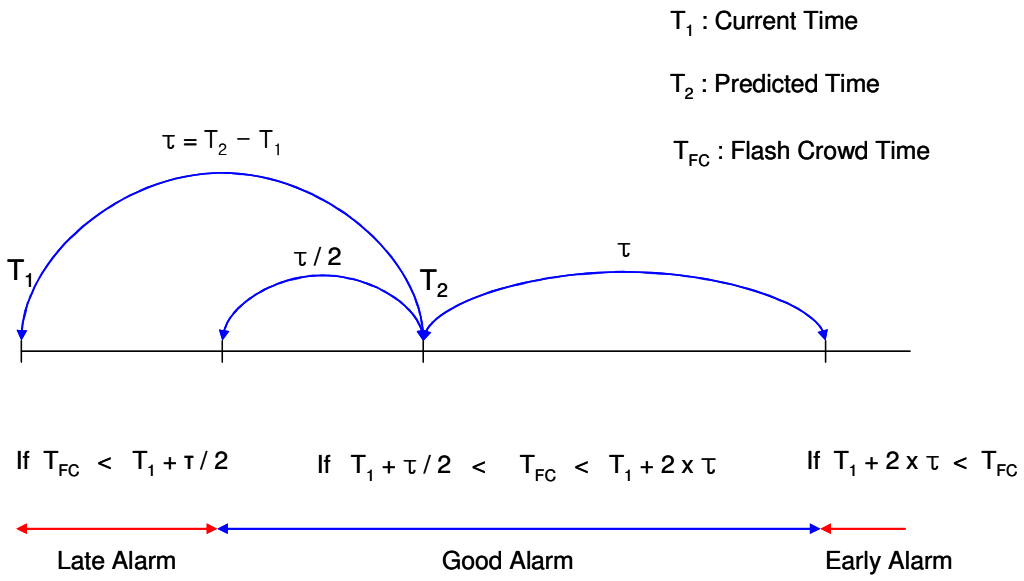

Figure 4.1 The definition of the alarm

Therefore, in our experiment, we define the allowable error as the following. Even though flash crowd happens earlier than time T_2 which is predicted at time T_1, if the flash crowd occurs after $T_1 + \tau / 2$, we regard it as a correct prediction. If the flash crowd occurs before $T_2 + \tau$, we also take it as a correct prediction.

Actually, τ depends on the time to replicate the objects because the system should be able to replicate all objects before the predicted flash crowd happens. By the way, every member server does not have same replication time because their network resources as well as physical location are different. In FCMS, we define τ as average replication time as in figure 4.1. In addition, the latter tolerance time (τ) is longer than the earlier tolerance time ($\tau/2$). The reason for this is that, although time T_2 ($T_1 + \tau$) is preconfigured as the average time of the replication, it takes more time to replicate in actual flash crowd. Therefore, we do not regard the prediction as an early alarm, even

though the predicted flash crowd occurs relatively much later ($T_2+\tau$) than predicted time T_2.

4.1.3 Parameters for Prediction

According to [13], a pertinent maximum capacity of a web server is doubled traffic in normal mode. Therefore, in this experiment, the threshold is 400, 500, and 600, since the average traffic in normal mode is between 200 and 300. The time τ for the replication is 5, 10, and 20 minutes and the prediction window is 10, 20, and 40 minutes for each prediction time (τ), as we have defined the prediction window as $\tau \times 2$.

4.2 Result

First, table 4.1 shows the result for the simulation of the prediction module with the release policy of [5]. Basically, the number of performed prediction is relatively high which means that the system releases the system mode from flash crowd mode very often. Also, averagely, false alarm is about 45% of the total prediction. The reason for the high percentage of false alarms is that, when the traffic continually stays near threshold, the prediction and release is reiterated. Early alarm is not shown because the system always releases the flash crowd mode after time τ_{max}, as we mentioned in section 3.2.2. However, the late alarm is shown (9%). As we describe in section 4.1.2, a late alarm indicates that flash crowd happens earlier than predicted time. This means that the system cannot complete the replication, and the number of incoming requests over threshold should not be processed. If the traffic goes down below the threshold, the system will release the flash crowd mode in [5]. In this case, however, if the traffic is increased rapidly again as soon as the system releases, the system cannot do an advance

alarm and this alarm should be a late alarm as shown in figure 4.2. The average of true positive is 46% of the total predictions

Table 4.1 The simulation for the prediction module with the release policy of [5]

Threshold	Prediction Window	Prediction time(τ)	False Alarm	False Positive	False Negative	True Positive	Total
				Early Alarm	Late Alarm	Good Alarm	
400	10min	5min	95 (63%)	0 (0%)	15 (10%)	40 (27%)	150
	20min	10min	45 (53%)	0 (0%)	11 (13%)	29 (34%)	85
	40min	20min	30 (51%)	0 (0%)	5 (8%)	24 (41%)	59
500	10min	5min	43 (55%)	0 (0%)	9 (12%)	26 (33%)	78
	20min	10min	27 (50%)	0 (0%)	8 (15%)	19 (35%)	54
	40min	20min	17 (44%)	0 (0%)	5 (13%)	17 (44%)	39
600	10min	5min	29 (51%)	0 (0%)	3 (5%)	25 (44%)	57
	20min	10min	9 (24%)	0 (0%)	1 (3%)	28 (74%)	38
	40min	20min	5 (15%)	0 (0%)	2 (6%)	26 (79%)	33

Secondly, the result for the simulation of the prediction module with the release policy of FCMS is shown in table 2. Averagely, false alarm is about 16% of the total prediction, and early alarm is about 37%. However, in regard to early alarm, although the flash crowd did not occur at the predicted time and the system is activated needlessly, it should not be the failed service for the request unlike the late alarm. In FCMS, a late alarm is not shown. As shown in figure 4.2, it means that the system predicts the flash crowd initially and the system does not make late alarm, although the traffic is decreased below the threshold temporarily. In FCMS, although the average of

true positive (47%) is almost same as the result for [5], the average of the sum of true positives and false positives (84%) is relatively very high.

Table 4.2 The simulation for the prediction module with the release policy of FCMS

Threshold	Prediction Window	Prediction time(τ)	False Alarm	False Positive	False Negative	True Positive	Total
				Early Alarm	Late Alarm	Good Alarm	
400	10min	5min	1 (8%)	7 (58%)	0 (0%)	4 (34%)	12
	20min	10min	1 (8%)	6 (46%)	0 (0%)	6 (46%)	13
	40min	20min	2 (15%)	6 (46%)	0 (0%)	5 (39%)	13
500	10min	5min	2 (13%)	10 (62%)	0 (0%)	4 (25%)	16
	20min	10min	3 (25%)	5 (42%)	0 (0%)	4 (33%)	12
	40min	20min	4 (27%)	3 (20%)	0 (0%)	8 (53%)	15
600	10min	5min	3 (17%)	6 (33%)	0 (0%)	9 (50%)	18
	20min	10min	3 (16%)	3 (16%)	0 (0%)	13 (68%)	19
	40min	20min	3 (17%)	1 (6%)	0 (0%)	14 (77%)	18

The graphs in figure 4.2 and 4.3 can visually show the difference of the release time of the flash crowd mode in between FCMS and [5].

This result is for a week (49^{th} ~ 55^{th} day) during the period of 1998 World Cup. Threshold is 600, prediction window (W_d) is 2400 seconds, and prediction time (τ) is 1200 seconds. The ordinate is the number of request, and the abscissa is time. Blue point is the number of request in normal state. Red point occurring in the traffic of flash crowd means that the current amount of traffic is over the capacity of web server. While the system is in flash crowd mode, the graph color of traffic is yellow. Therefore, if the

traffic is over the threshold and the system set flash crowd mode at the same time, two colors will overlap. In this figure, red is used for the overlap. In other words, the redirection by FCMS is executed during red and yellow traffic. In figure 4.2, the color is often changed from blue to yellow, and late alarm happens. On the other hand, in figure 4.3, the system keeps flash crowd mode until traffic is stable, and late alarm is not shown.

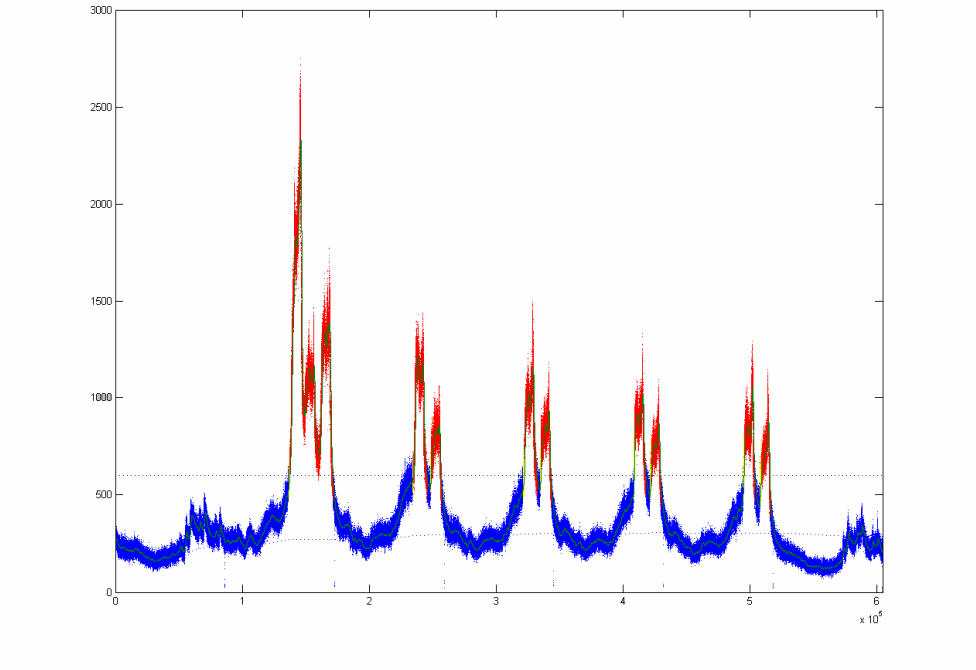

Figure 4.2 The simulation for the prediction module with the release policy of [5]

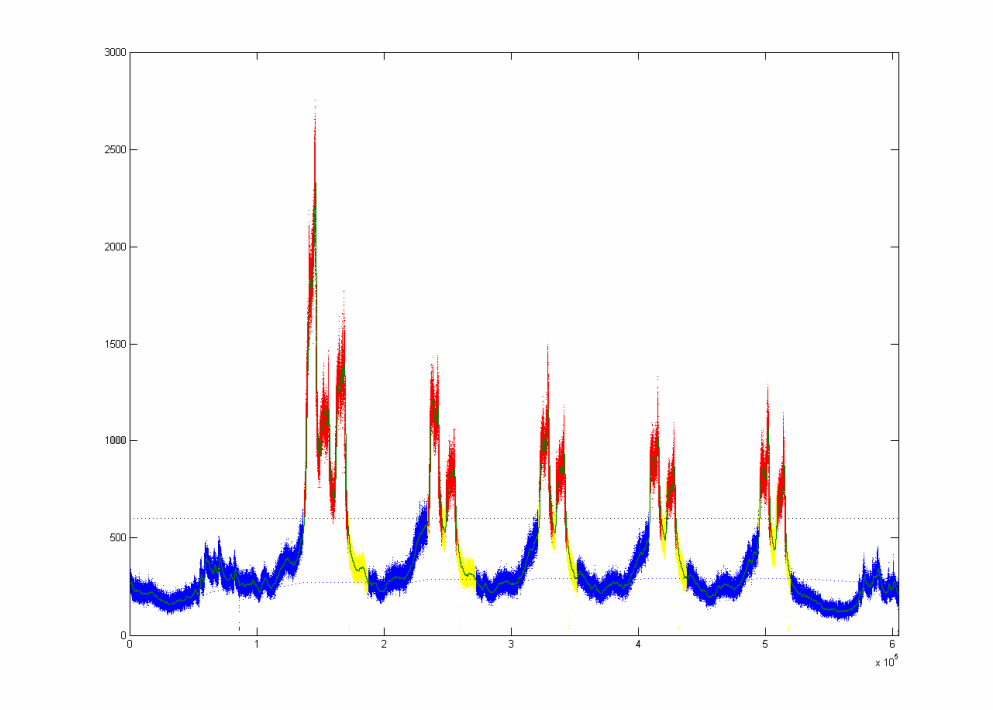

Figure 4.3 The simulation for the prediction module with the release policy of FCMS

CHAPTER 5

EXPERIMENT IN TESTBED

5.1 Experiment Setup

Figure 5.1 illustrates our testbed environment. We employ four machines as member server, one machine as end router, and one machine as traffic generator. Ideally, as we designed, every member server has each end router to monitor the traffic, but only one member server has a router in our experiment, since we generate the flash crowd for one member server, every router is not strictly required.

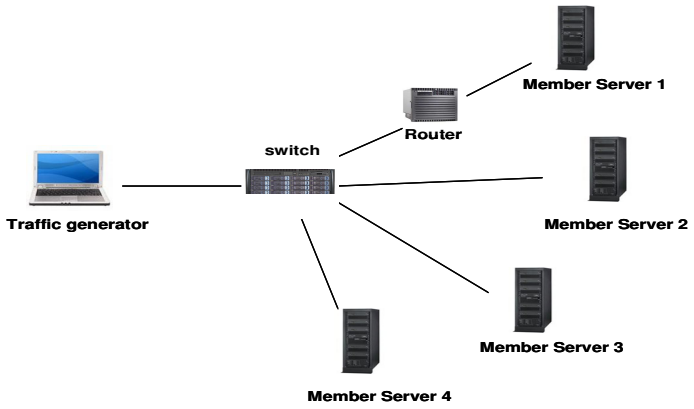

Figure 5.1 Testbed architecture

Member server 1 as the target server has 2.4GHz Pentium4 CPU with 1GB RAM. Member server 2 has 3.0GHz Pentium4 CPU with 1GB RAM. Member server 3 has 930MHz Pentium3 CPU with 256MB RAM. Member server 4 has 3.0GHz Pentium4 CPU with 1GB RAM. The router machine has 3.0GHz Pentium4 CPU with

1GB RAM. Traffic Generator has 3.2GHz Pentium4 CPU with 2GB RAM. Suse Linux 10.2 is used for every member server. Router is using Suse Linux 10.1. Traffic generator uses Windows XP. All member servers employ the Apache2 as a web server application. To embed the prediction and monitoring module into the router, we have used CLICK modular router which is able to work on kernel level. Traffic generator is a Java program using HttpUrlConnection class. All machines have Gigabit Ethernet network interfaces. Traffic generator, end router, and three member servers are connected to the same Gigabit switch. For the experiment, we assume that the capacity of the web server is 30 requests per second because we are not focusing on getting the capacity of web server. Basically, traffic generator generates about 10 requests for 20 seconds, and then traffic generator generates the HTTP request on the increase from 10 to 70 for 60 seconds. Traffic generator reduces the number of request from 70 to 10 for 60 seconds. After that, it keeps generating about 10 requests for 20 seconds again. However, we give some variation per second to ensure that the traffic is not generated as linear. Java random function and modulus operator with 4 is used for the variation. When the number of requests is over 30, the router returns the 404 error response code to drop the requests. The target server has a total of 60 Mbytes data and the number of data is about 400 objects. (100 document and 300 pictures) If the web site has more objects, the system should predict earlier with longer prediction time (τ).

5.2 Flash Crowd Mitigation

Figure 5.2 shows the experiment without FCMS. Blue dotted line is the number of requests, and red dotted line is the number of response. Initially, the response rate is

same as the request rate by the threshold. However, the response rate cannot follow the request rate over the threshold. As several tens experiment, the total number of request is about 3500, and the total number of responses is about 2500. Therefore, about 1000 requests are dropped which means the response rate is about 70%.

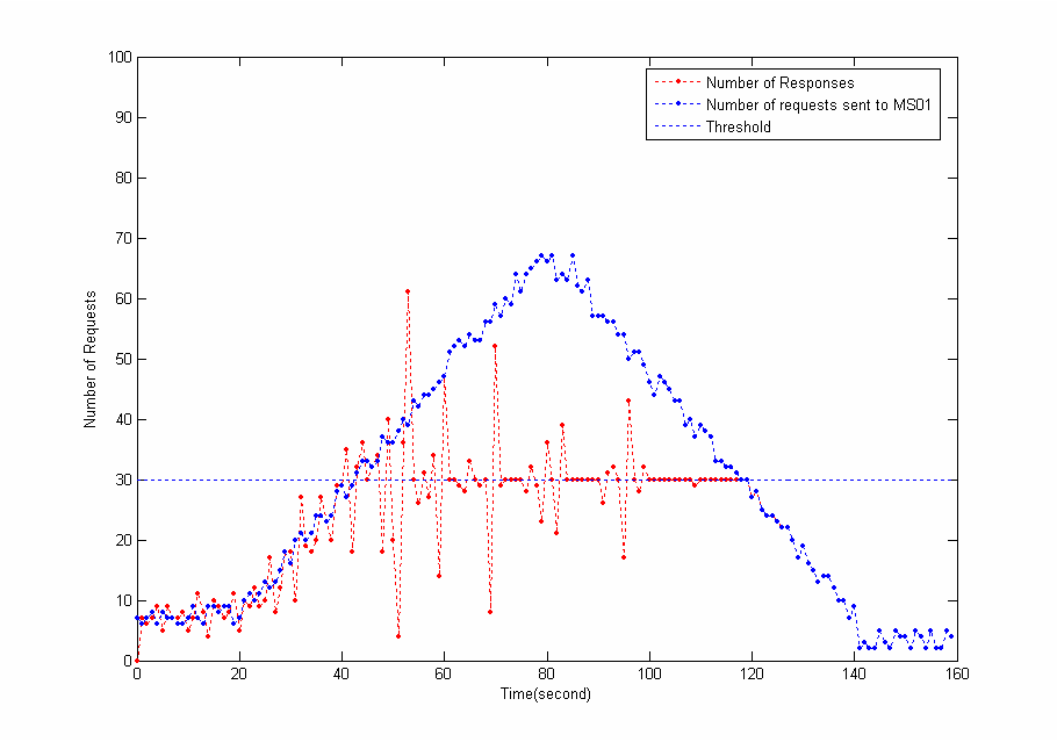

Figure 5.2 Response rate without FCMS

Figure 5.3 shows the experiment with FCMS. The parameter W_d for the prediction is 20 and τ is 10. Black solid line means the total number of requests sent by the traffic generator. Blue dotted line means the number of requests in the target server (MS01) and red dotted line indicates the total number of response from all member servers. The other three dotted lines are the number of requests which distributed to other member servers.

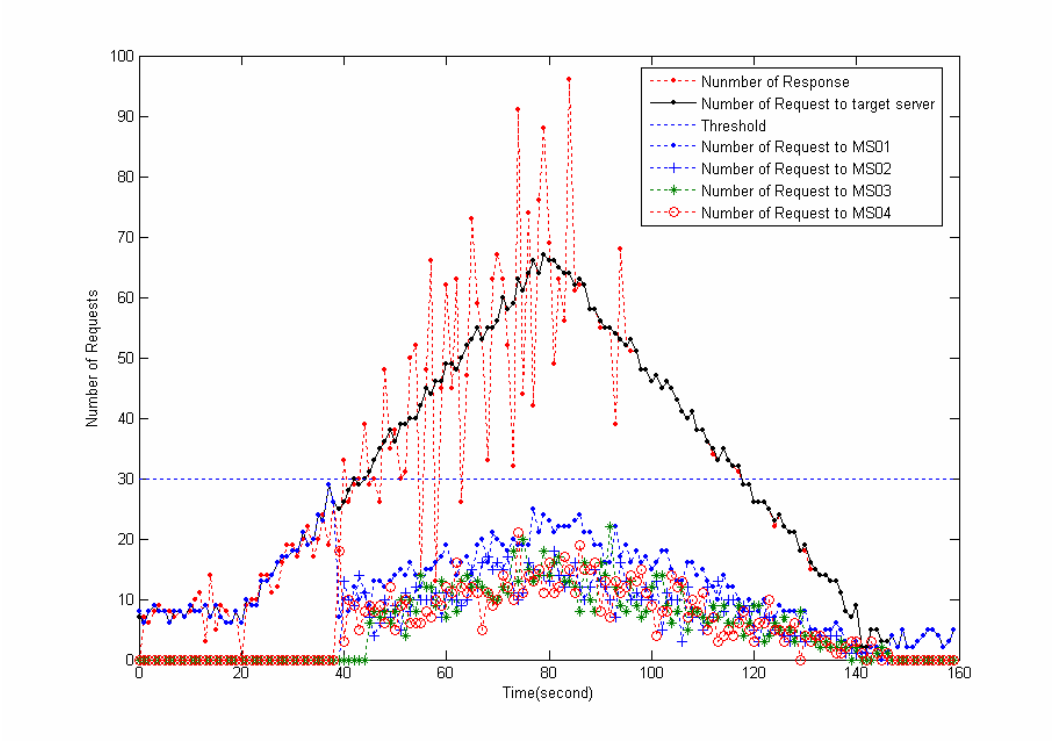

Figure 5.3 Response rate with FCMS and distributed requests

At 28 second, the system predicts that the flash crowd will happen, and the redirection is started in 38 second. This means that it takes 10 seconds to exchange the message and perform the redirection. Actual flash crowd happens at 44 second.

After 38 second, the number of requests which is sent to each member servers is increased rapidly and the number of requests to the target server is decreased quickly. Totally, about 3500 requests are generated, and total response rate is 99%. This result means that the redirection module is working correctly to mitigate the effect of flash crowd in the target server.

CHAPTER 6

CONCLUSION

In this paper, we have proposed Flash Crowd Mitigation System performing with the router embedded with the monitoring and redirection modules, member-server manager program in java, the protocol for the communication between the member servers. FCMS is designed for the "poor" server which does not have the ability to use an additional network resource for the flash crowd. Therefore, FCMS is operated on the basis of the cooperation of the member servers without any additional cost. The monitoring module uses the linear regression for the prediction and the proper release point of time from the flash crowd for the effective activation of the system. When the flash crowd is predicted, the member servers duplicate the target server's object, and then the redirection module in the router redirects the requests to the member server and target server on balance using the virtual TCP connection which can redirect without any receiving queue. The simulation with World-Cup traces shows that the monitoring module can predict and release the flash crowd effectively. Also, our experiment using testbed demonstrates that implemented FCMS and our mechanism work correctly and mitigate the effect of flash crowd in target web server.

REFERENCES

[1] Jung, Jaeyeon and Krishnamurthy, Balachander and Rabinovich, Michael Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In Proceedings International WWW Conference, Honolulu, Hawaii, USA. (2002)

[2] http://local.wasp.uwa.edu.au/~pbourke/fractals /quatjulia/google.html

[3] http://www.cnn.com/TECH/computing/ 9902/05/vicweb.idg/

[4] Tyron Stading, Petros Maniatis, Mary Baker, Peer-to-Peer Caching Schemes to Address Flash Crowds, 1st International Peer To Peer Systems Workshop (IPTPS 2002)

[5] Baryshnikov, Y. Coffman, E. Pierre, G. Rubenstein, D. Squillante, M. Yimwadsana, T., Predictability of Web-server traffic congestion, Web Content Caching and Distribution, 2005. WCW 2005. 10th International Workshop

[6] Xuan Chen, John Heidemann, Flash Crowd Mitigation via Adaptive Admission Control based on Application-level Observations, ACM Transactions on Internet Technology, Vol 5, No. 3, August 2005.

[7] Hani Jamjoom, Kang G. Shin, On the role and controllability of persistent clients in traffic aggregates IEEE/ACM Transactions on Networking (TON), April 2006

[8] Stavrou, A. Rubenstein, D. Sahu, S. A lightweight, robust P2P system to handle flash crowds, Selected Areas in Communications, IEEE Journal on Jan. 2004

[9] Hui, K.Y.K. Lui, J.C.S. Yau, D.K.Y., Small world overlay P2P networks, Quality of Service, IWQOS2004. Twelfth IEEE International Workshop on, 2004.

[10] Chenyu Pan, Merdan Atajanov, Mohammad B. Hossain, Toshikiko Shimokawa, Norihiko Yoshida, FCAN: Flash Crowds Alleviation Network, Symposium on Applied Computing, Proceedings of the 2006 ACM symposium on Applied computing

[11] Pascal Felber, Tim Kaldewey, Stefan Weiss, Proactive Hot Spot Avoidance for Web Server Dependability Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS2004)

[12] Gaurav Banga, Peter Druschel Measuring the Capacity of a Web Server (1997) USENIX Symposium on Internet Technologies and Systems

[13] http://www.soft.com/eValid/Technology/White.Papers/website.loading.html

[14] http://ita.ee.lbl.gov/html/contrib/WorldCup.html

BIOGRAPHICAL INFORMATION

Dongchul Kim has graduated the department of computer science in Kyungsung University, Busan, Korea, in the year 2003, and he received his Bachelor of Science degree. He began his graduate studies in 2004 at the Computer Science and Engineering department in the University of Texas at Arlington. Since 2005 he has been a member of the Center for Research in Wireless Mobility and Networking (CReWMaN), and he received his Master of Science in Computer Science and Engineering from the University of Texas at Arlington in August 2007. His current research interests include flash crowd and wireless sensor networks.