Workload Scheduler with Fault Tolerance for MMSC*

Juhee Hong, Hocheol Sung, Hoyoung Lee, Keecheon Kim, Sunyoung Han

Department of Computer Science and Engineering , Konkuk University,
1 Hwayangdong, Kwangin-gu, Seoul, 143-701, Korea
{jhhong, bullyboy, hylee, kckim, syhan}@kkucc.konkuk.ac.kr

Abstract. Media server of the MMSC(Multimedia Message Service Center) offers a function that converts sound and image for various types of handsets. MMS Relay requests a media type and format conversion and the media server handles this conversion. When MMS Relay requests a media conversion, a workload scheduler makes the job distributed properly and processed by several media servers. Since the whole system load can be distributed evenly, the performance of MMS system can be improved. The stability and the reliability are provided. In this paper, we propose and implement a workload scheduler in which jobs are processed distributively by monitoring the weights of the jobs in each media server. The weights are assigned to the jobs according to the conversion time. Also, we guarantee a fault tolerance capability of media server to retransmit the jobs fast when fault happens by monitoring the active/idle states and executing the processes.

1 Introduction

With the development of wireless communication and high-speed network, there is a clear need for a more advanced messaging service that allows mobile users to send and receive longer messages with richer contents. In mobile a message service, SMS (Short Message Service) allowing only plain text has been evolved to MMS (Multimedia Messaging Service) that accepts a mixture of different media types including text, image, sound and video. This new messaging service called MMS can be in the future mobile networks such as GPRS and UMTS [1][2][10]. In the multimedia messages, it is possible to perform a format conversion based on the characteristics of the handsets. It is a media server that takes the responsibility of the conversion when a MMS Relay requests a conversion. Since a multimedia message takes longer to convert than text, multiple media servers are needed for faster processing [1][3].

3GPP and WAP Forum specify the necessity of media server for this type conversion, however they don't specify the performance improvement through fast conversion of multimedia message explicitly [2][3]. Therefore, load balancing is necessary

* This work is supported by NCA(National Computerization Agency) of Korea for Next Generation Internet Infrastructure Development Project 2002.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 643–652, 2002.

to prevent a overload in a specific media server when multiple media servers convert media formats.

In this paper, we propose and implement a workload scheduler that distributes jobs by monitoring the weight of the jobs in each media server and the scheduler runs the jobs according to weight assigned by the conversion time. The performance and the reliability can be improved in delivery the message. Also, we guarantee fault tolerance capability by retransmitting the job fast when a fault happens by monitoring active/idle states and executing the processes. We show enhanced performance in multiple media servers through our simulation tests.

The rest of this paper is organized as follows. In section 2, we introduce the overall MMSC component and architecture. In section 3, we present a design of workload scheduler supporting load balancing and fault tolerance. In section 4, we present our implementation, and in section 5, we compare and estate test results through the performance evaluation. Finally, section 6 gives our concluding remarks.

2 MMSC(Multimedia Message Service Center)

MMSC is a system for MMS as defined in our testing system based on 3GPP and WAP Forum. MMS is intended to provide a rich set of content to mobile subscribers in their messages, it supports both sending and receiving of such messages and assumes that messaging transmission and retrieval is supported by their handsets [1][3][10]. There are four basic types of equipment within MMSC, these are MMS Relay, MMS Server, Media Server, and Operation console [3][4].

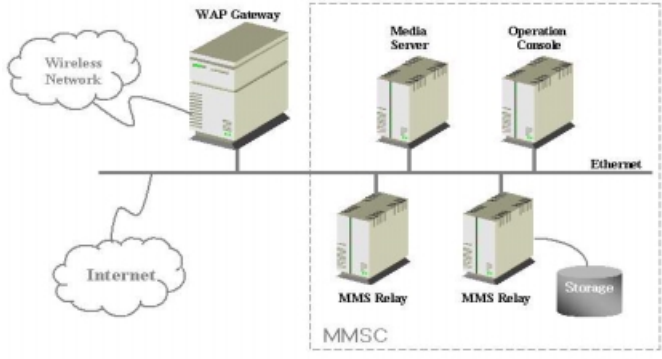

Fig. 1. MMSC(MMSC(Multimedia Message Service Center)

Fig. 1 shows an external configuration of our MMSC. The following explains each elements of Fig. 1.

- MMS Server

  This element is responsible for storing and handling the incoming and outgoing messages.

- MMS Relay
  This element transfers messages between different messaging systems associated with the MMS Server. It should be able to generate a charging data (CDR) when receiving multimedia messages or when delivering multimedia messages to the MMS User Agent or to another MMS environment [1][2].

- User Databases
  This is the system comprised of one or more entities that contain user related information such as subscription and configuration (e.g. user profile, HLR).

- User
  It is a function of application layer to provide the users with the ability to view, compose and handle multimedia messages such as sending, receiving, and deleting.

Depending on the business model, the MMS Server and the MMS Relay may be combined, or distributed across different domains. In practice, MMS system may be integrated in a single physical place and we explain it as a set of components for better understanding.

3 Design of Workload Scheduler

Media server is important and independent server in MMSC, because it performs a format conversion for the multimedia messages based on the characteristics of the handsets [1]. If one MMS Relay and one media server have been consisted separately, routing and scheduling issues between MMS Relay and media server aren't important. But as the necessity of messaging service increases, more than two MMS Relays and media servers are needed. Thus, the routing between MMS Relays and media servers is very important, because it influences the system performance. In order to support this routing function, there must be a monitoring capability in the media servers. Monitoring detects the faults and prevents an overloading problem.

Fig. 2 shows the scheduler suggested in this paper and it performs a load balancing for fast contents conversion in multiple MMS Relays and Media Servers. Because of the distributive processing for a request, it should include fault tolerance to handle the faults happened in a specific Media Server or process.

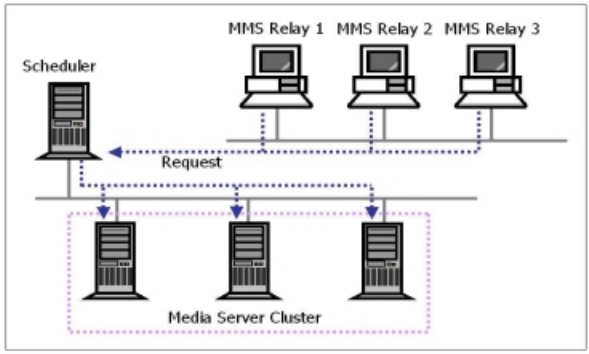

Fig. 2. System topology

We designed a workload scheduler based on the least-connection algorithm because the algorithm directs the network connections to server with the least number of established connections [6]. But the scheme suggested in this paper is based on the sum of the weight of jobs at work in each media server because the connections are within a local network. Weights are assigned for the fast media conversion because it takes different time for each media conversion. These weights are statistical values obtained through our tests for media conversion. Table 1 and 2 show the values.

Media server with the least of the weighted jobs is selected by the scheme below. If there are n jobs at work in a media server and each job has a weight $Wi(i = 1, \ldots, n)$, the weighted job in that media server is expressed by $\sum(1 / W_i)\ (i = 1, \ldots, n)$. Therefore, the selected media server has a value of $min\{\sum(1 / W_i)\}\ (i = 1, \ldots, n)$.

Table 1. Statistical value for image format type

(unit : count/second)

Source\Destination	GIF	JPEG	PNG	BMP	WBMP
GIF		5.1	5.0	5.2	4.2
JPEG	3.4		2.5	5.2	1.4
PNG	2.7	5.1		5.2	3.1
BMP	3.0	5.1	2.4		3.5
WBMP	0.2	5.1	5.0	5.2	

Table 2. Statistical value for audio format type

(unit : count/second)

Source\Destination	WAV	MP3	MID
WAV		3.9	2.9
MP3	5.0		2.0
MID	5.1	X	

A scheduler distributes the requests received from MMS Relay using the above scheme. Fig. 3 shows the whole processing diagram of the load balancing. Our system is consisted of two parts, a scheduler and a media server.

- Scheduler
  It is responsible for load balancing and fault tolerance, it creates a worker thread and processes the request.

- Media server
  It performs a media format conversion.

To select a media server with the smallest of weighted jobs, the scheduler retrieves media server list table and finds the states whether it is in active or idle for a candidate media server. This is repeated until the media server is selected to convert the request. When a media server is selected, weighted job field of the list table is increased. Monitoring the conversion state is made known by sending a signal to the media server. We have two signal types, one is a 'complete' signal when process finishes normally and the other is a 'fault' signals when an absurd end of process or other errors happen.

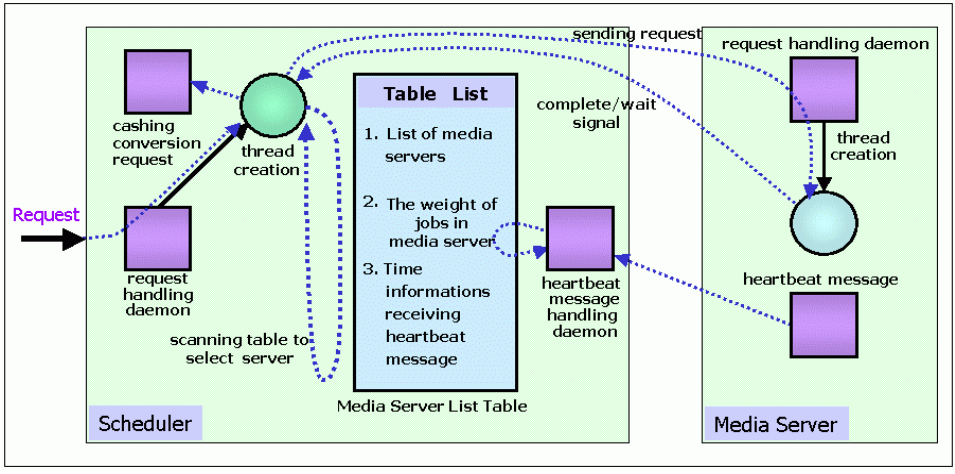

Fig. 3. Media server processing diagram

The current media server receives these signals and knows the conversion state. When a scheduler does not receive a signal in a given time, it regards it as fault and retrieves the next media server. When a complete signal comes to an end, job field in the list table decreases.

Conversion state of media server is reported to the scheduler through the periodic signal. A heartbeat message notifies that the server is active, and the scheduler updates time field of the media server in the list table. Updated time value is calculated from the receiving heartbeat message time plus(+) the given time intervals. If a

heartbeat message does not arrive in time, the media server is regarded as fault and retransmits the requests to another server.

4 Implementation

4.1 Modules

The system consists of two parts, a scheduler and a media server. A scheduler forwards the requests to a media server using load balancing and fault tolerance mechanism. It consists of a request-handling module, a media server list table, and a heartbeat message-handling module. Media server appends a heartbeat message-sending module and a complete signal-sending module to monitor the state conversion process.

The request-handling module distributes the conversion requests to media server evenly. It is required to listen a request from MMS Relay and forward appropriately to a media server by the load balancing algorithm. It creates a new thread whenever a request comes in and destroys the thread in its completion.

The request-handling thread needs an algorithm to check the state for sending request to a media server. First, it should have a capability of retrieving a media server with the smallest total weight and checks the state, whether it is in active or idle. If it is selected, a media server will receive and execute the conversion request.
Then the request-handling thread increases the weight of job in the media server list table, and it starts monitoring the job. If a fault occurs, a new media server will be selected again. Otherwise, the thread decreases the weight of the jobs and it is destroyed automatically.

4.2 Structure

A media server list table is a table to retrieve a media server with the smallest of weighted jobs by the load balancing algorithm. It consists of Server ID, Server Address, Job, Time and Critical Section for synchronization.

Server ID is an unique number for identifying each media server and the Server Address is IP address. Job is the weighted value of the job in execution in each media server and Time is the expected time that the media server will be active for some times. The value of the Time is calculated from the incoming time value of heartbeat message added to the predefined interval value. During that time, a scheduler considers that media server is active. Otherwise it is idle. Critical Section is used for locking when the table is being updated.

Table 3. Media server List Table

Server ID	Server Address	Job	Time	Critical Section
Unique identification of each media server	IP address of each media server	The weighted value of the job in execution in each media server	The value of incoming heartbeat + predefined interval value	The variable for synchronization

Table 4. Media server List Table structure

```
typedef struct _ServerEnt {
          int    ServerId;
          SOCKADDR_IN       ServerAddr;
          unsigned int      uWgtOfJob;
          DWORD    dwTime;
          CRITICAL_SECTION        critical_sec;
 } ServerEnt;
```

4.3 Fault Tolerance Mechanism

When a media server starts a service, it creates a thread for sending a heartbeat message and forwards it to a scheduler. If a media server succeeds in UDP connection to a scheduler, it starts sending a heartbeat message. Then, the request-handling thread executes UDP listening daemon to receive the message and updates the time field in the media server list table whenever it receives a message. If a heartbeat message does not arrive in time, the media server is regarded as fault, and retransmits the requests to another server.

A media server sends a complete-signal to Process Monitoring Daemon in a scheduler when the job terminates normally, otherwise it sends a fault-signal. We use '1' for a complete-signal and '0' for a fault-signal in this paper.

5 Test Results

We built a test bed in order to compare the performance with the legacy MMS system. So, we prepare five media servers, a scheduler and a client application for playing parts in MMS Relay. We used a Pentium server for media servers and a scheduler. We used windows 2000 compatible OS. Each media server and scheduler was located in a local network. Also, we had to prepare MMS storages for various media files. Each media server connected to MMS storage with a network drive of supporting OS. Therefore, each media server operates as if the source media file for conversion is in the same local drive. And we killed all the unnecessary processes in the media server and scheduler for an accurate test. In order to measures the performance, we have sent between 10 and 100 requests at a time to a media server for message conversion, and then we calculated the average processing completion times. The graph shows the results of this implementation.

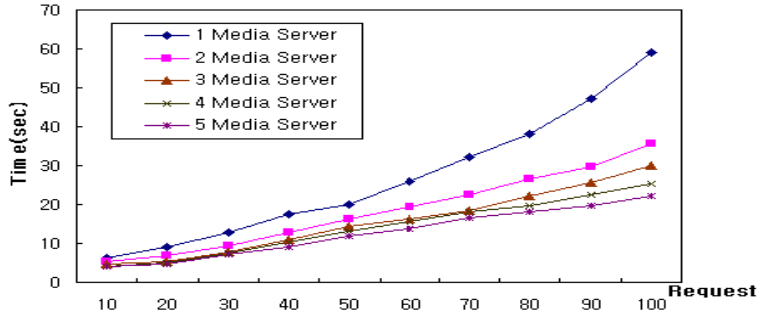

Fig. 4. Using with 1Mbyte wav file

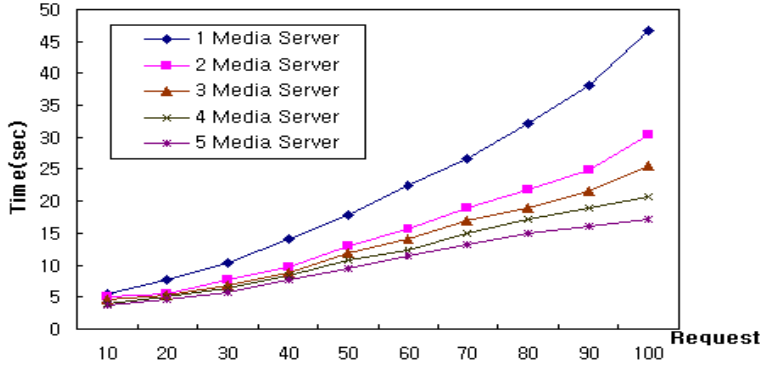

Fig. 5. Using with 500Kbyte wav file

We measured the processing time by increasing the number of media servers in order to compare the performance improvement resulting from our system. As for the various performance tests, we calculated the processing time with changing the file size of different media types. Figures, 4 through 7, show the result of the test. Fig. 4 shows the test result with 1Mbyte audio file. Fig. 5 to Fig. 7 shows the test result with 500Kbyte, 150Kbyte and 50Kbyte respectively. Clearly, the conversion time is different as the number of media servers, and the media server can convert very fast in case of small size of media files. In case of sending 10 requests, there are not different between single media server and five media servers as the graphs show. It spent more time for reaching requests rather than for converting a media file. The increased performance can be seen as the converting requests increased. And the bigger a media file size is, the greater the difference in converting time between a single media server and five media servers. It seems to be obvious that the same result applies to the image files and other audio format files.

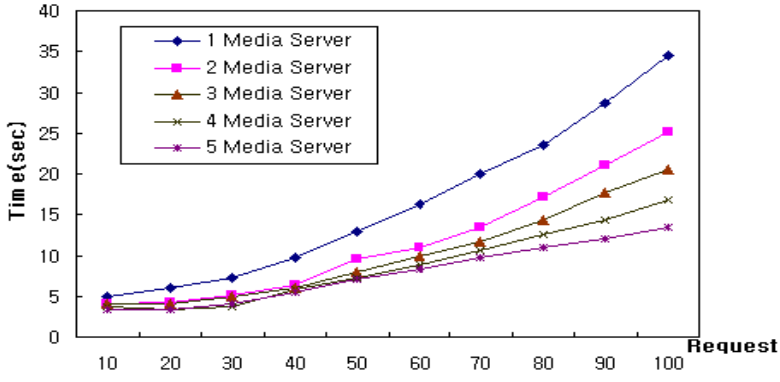

Fig. 6. Using with 150Kbyte wav file

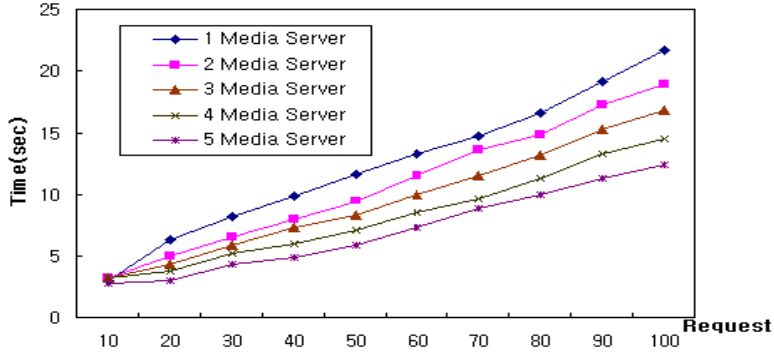

Fig. 7. Using with 50Kbyte wav file

As a result, we found that our system improves the performance dramatically than the legacy MMS system.

6 Conclusions

In this paper, we designed and implemented a workload scheduler by applying a least-connection algorithm in MMSC for MMS defined in our prototype system. The workload scheduler distributes job processing by monitoring the weights of the jobs in each media server to improve the system performance. We also implemented a fault tolerance that is a function of media server of retransmitting the job fast when a fault happens by monitoring the active or idle state of the media server and executing the processes. Test results present clearly that our scheduler performs a fast conversion and provides more stable and reliable system. Future work includes a test for video file conversion and streaming service.

References

1. 3rd Generation Partnership Project.: Technical Specification Group Services and System Aspects; Service aspects; Stage 1 Multimedia Messaging Service (Release 2000), 3G TS 22.140 v.4.0.1 (2000-07)
2. 3rd Generation Partnership Project.: Technical Specification Group Terminals; Multimedia Messaging Service (MMS); Functional description; Stage 2(Release 4), 3G TS 23.140 v.4.2.0 (2001-03)
3. Wireless Application Protocol Forum.: WAP MMS Architecture Overview, WAP-205-MmsArchOverview, Draft version 01-Jun-2000
4. Wireless Application Protocol Forum. : Wireless Application Protocol MMS Interworking with Internet Email Specification, WAP-207-MmsInetInterworking, Draft version 01-Jun-2000
5. http://www.mobilemms.com
6. http://www.linuxvirtualserver.org/docs/scheduling.html
7. B. Ozdenm, R. Rastogi, P.J. Shenoy, A. Silberschatz.: Fault-Tolerant Architecture for Continuous Media Server, ACM SIGMOD, Montreal, Canada(1996)
8. Iwata. A, Ching-Chuan Chiang, Guangyu Pei, Gerla. M, Tsu-Wei Chen.: Scalable routing strategies for ad hoc wireless networks, IEEE Journal, Vol. 17, Issue: 8(1999) 1369-1379
9. Golubchik. L, Muntz. R, Cheng-Fu Chou, Berson. Sm.: Design of fault-tolerant large-scale VOD servers With emphasis on high-performance and low-cost, IEEE Transactions on, Vol. 12,Issue: 4 (2001) 363-386
10. Sevanto. J.: Multimedia messaging service for GPRS and UMTS Wireless Communications and Networking, WCNC. IEEE, Vol. 3 (1999) 1422-1426