International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.5 (2015), pp.411-428
http://dx.doi.org/10.14257/ijmue.2015.10.5.38

Distributed Large-Scale Conferencing System Architecture with Dynamic Server Allocation in SIP Environment

Choonseo Jang

Department of Computer Engineering, Kumoh National Institute of Technology, Gumi, Korea
csjang@kumoh.ac.kr

Abstract

In this paper, in order to improve scalability of conferencing system in SIP(session initiation protocol) environment, a distributed large-scale conferencing system architecture with dynamic server allocation has been proposed. For this architecture, a new extended conference information data format has been presented to distribute system load dynamically to multiple conference servers. In order to control system load and conference object, the proposed extended conference information data format has been designed to include new added elements and attributes. The extended conference event package has been used to synchronize overall system conference information data between conference servers. This paper also shows detailed exchange procedures of SIP messages between conference servers and participants for distributing system load dynamically. An extensive experimental measurements have been done to verify the performance of the proposed distributed conferencing system architecture. The results show that the proposed architecture improves highly scalability and flexibility of the conferencing system

Keywords: *session initiation protocol, distributed conferencing system, conference information data format, conference event package*

1. Introduction

In SIP-based conferencing system, the conference focus maintains and manages SIP conference session, and the mixers processes audio/video frames generated from each participants [1-4]. Therefore load of the focus and mixers increases as the number of participants increases, and it becomes the major reason that limits the scalability of the large-scale conferencing system. However, many studies about conferencing system with multiple servers have some limitations. Some approaches have focused on allocation of media processing load to multiples mixers [5-6]. SIP session establishment control, conference information data [7] control and QoS (quality of service) control are not fully distributed in these approaches, and delay time to process SIP messages and audio/video streams may not be acceptable when many participants join the conference. Furthermore, these approaches are lack of conference information data synchronization capability between servers. In large-scale conferencing system, severs should frequently exchange their conference information data for effective load distribution during conferencing operation, however, these approaches do not present such functions.

Some studies have proposed P2P-signaling protocol scheme for distributed conference control [8-9], but their methods require resource location and discovery function for both conference servers and participants, and these facts limit their usages. So in this paper, a new distributed conferencing system architecture, which enable to controls each server's load and distributes dynamically overall system load to each server, has been suggested. In order to control conference system load, a new extended conference information data format which has some added elements for load distribution and control has been designed. And detailed exchange procedures of SIP messages between conference servers and participants have been presented.

In Section 2, the related work with regard to conferencing system architecture in SIP environment is described. Section 3 presents newly designed distributed large-scale conferencing architecture, and a new extended conference information data format with some added elements for the distributed conferencing system architecture. The extended conference event package and exchange procedures of SIP messages between conference servers and participants have been also described in this section. In Section 4, extensive experimental measurements have been done to verify the performance of the proposed distributed conferencing system architecture, and Section 5 is dedicated to conclusion remarks.

2. Related Work

The IETF researches on conferencing system have defined frameworks for allowing participants to exchange media in a centralized conference [7,10,11]. These documents also have defined logical entities required for conference systems, and have outlined conferencing protocols for building advanced conferencing applications. These researches use the conference object as a logical representation of a conference instance, representing the current state and capabilities of a conference. However, these frameworks are for centralized single conference server architecture, and they cannot be applied to distributed conferencing system with multiple servers. There have been a few researches about scalable conference architecture to increase extendability in the large-scale conferencing system. One of them is policy-based management architecture [6]. This approach focuses on allocation of media processing load to several mixers, and in this case conference information data control and SIP session establishment control are not fully distributed in the conference system. Furthermore, this policy-based management architecture does not use the concept of conference event package, so the approach has problem of real time conference information synchronization between servers.

Some studies propose P2P-signaling protocol scheme for distributed conference control using resource location and discovery method[8,9]. These approaches provide functions such as a mechanism for proximity-aware routing within a conference, and mechanisms for conference synchronization and call delegation for conferencing system. These researches have defined applications of resource location and discovery method, and suggest Kind code data structure for distributed conferencing system. This data structure provides mapping of a single conference URI to multiple conference controllers, so it provides scalable signaling for resource location and discovery in conferencing system. However, these methods require resource location and discovery function for both conference servers and participants, so this fact restricts their usage.

In conferencing system, authenticated and authorized participants should be allowed to create, manipulate, and delete conference objects. Some studies on centralized conferencing manipulation protocol (CCMP) have suggested conference operations such as adding participants, removing participants, changing their roles, adding and removing media streams and associated endpoints[12,13]. However, these papers only describe the client-server model within the centralized conferencing framework, so these approaches cannot be used to distributed conferencing system with multiple servers.

In order to overcome such restrictions on previous works, a new architecture for SIP-based distributed conferencing system with dynamic server allocation capability has been suggested in this paper. In this system, overall system load is effectively and dynamically distributed to each conference servers. In this study, a new extended conference information data format which has some added elements for load control and conference objects manipulation has been designed. Furthermore, a new extended conference event package for synchronizing conference information between servers has been also proposed. And, for distributing system load dynamically to each server, detailed exchange procedures of SIP messages between servers and participants has been presented.

3. System Design and Implementation

In this section, a newly designed distributed large-scale conferencing system architecture that allocates system load dynamically to multiple conference servers has been suggested. This section also describes an extended conference information data format with new added elements for controlling conferencing system load, and presents extended conference event package using the extended conference information data format with exchange procedures of SIP messages between conference servers and participants.

3.1. Design of Distributed Conferencing System Architecture

Figure 1 shows a designed architecture of distributed large-scale conferencing system proposed in this paper. In this system, the conference server is comprised of conference focus, media mixer, extended conference information and conference load control module. The conference focus maintains SIP sessions between serves and participants. The media mixer maintains RTP sessions with participants, and it mixes and distributes media streams from participants. The extended conference information represents overall conference information data of the conference system with new added elements required to allocate participants dynamically to each server in this distributed conferencing system. The conference load control module monitors and generates SIP signals to distribute system load to each server.

In order to synchronize overall conference information data, each server subscribes extended conference event package, which uses extended conference information data format designed in this study, using SIP SUBSCRIBE message to the other servers. This extended conference event package is used also to notify changes of conference information data to the participants. When a participant sends SIP INVITE message containing conference URI to the primary conference server, the server checks its current value of <load-level> element of the extended conference information data format to determine whether it can handle the participant's request. When the current load value is within pre-determined limit, the server sends SIP 200 OK response, and establishes RTP session with the participant to make the participant join the conference. The detailed function of new added <load-level> element is described on Section 3.2. When the current load value exceeds pre-determined limits, the server checks current load values of the other servers, and selects a server with the lowest current load value. Then, the primary server sends 302 Redirection SIP response message representing this selected server to the participant, and the participant sends again SIP INVITE message to this server to make conference session with it.

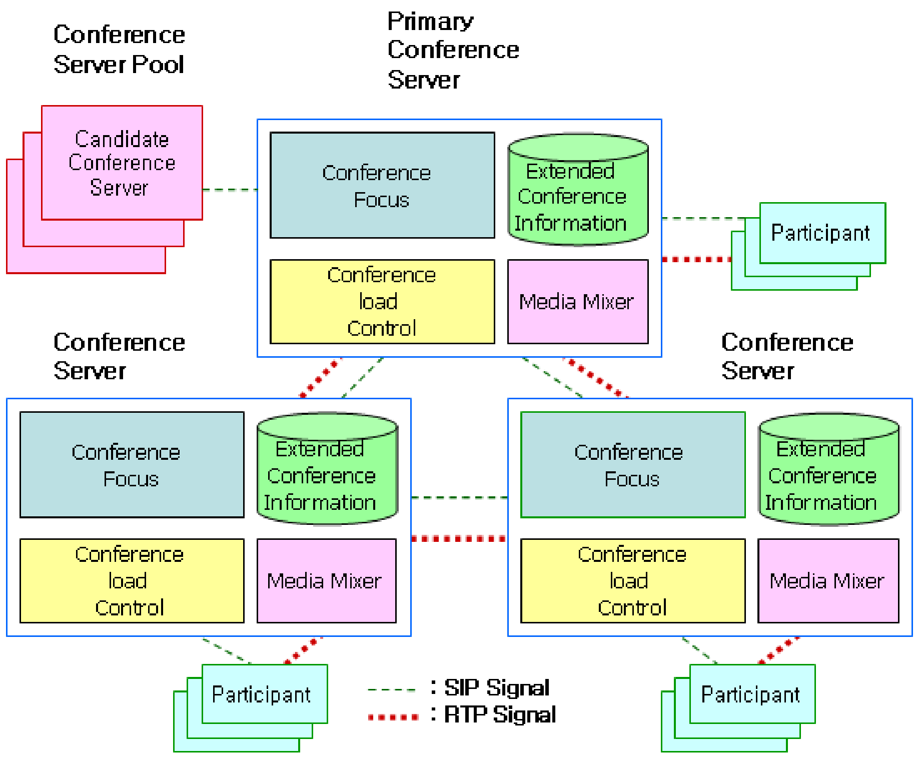

Figure 1. Designed Architecture of Distributed Conferencing System

When all the other server's current load values exceed pre-determined limits, the primary server fetches extended conference information data elements <reserved-servers-list>, to selects a new server which can be added to the conference from the reserved servers list. The detailed function of new added <reserved-servers-list> element is described on Section 3.2. Then, the primary server sends SIP INVITE message to this new server to make conference session with it. The new added server sends SIP SUBSCRIBE message to the primary server to subscribe extended conference event package, and the primary server notifies current extended conference information data using SIP NOTIFY message. After receiving current extended conference information data, the new added server make SIP session and RTP session with the other active servers in the conferencing system. The primary server sends SIP redirection response message representing this new added server to the participant, and the participant sends again SIP INVITE message to this new added server to make conference session with it. After new conference server is added successfully following the above procedures, the primary server starts to reassign the existing participants to this new added server to share overall load of the conference system. As primary server does more jobs than the other servers, its pre-determined load limit can be set to lower value than the other servers by conference policy. Figure 2 shows signaling procedure of the proposed conferencing system operation.

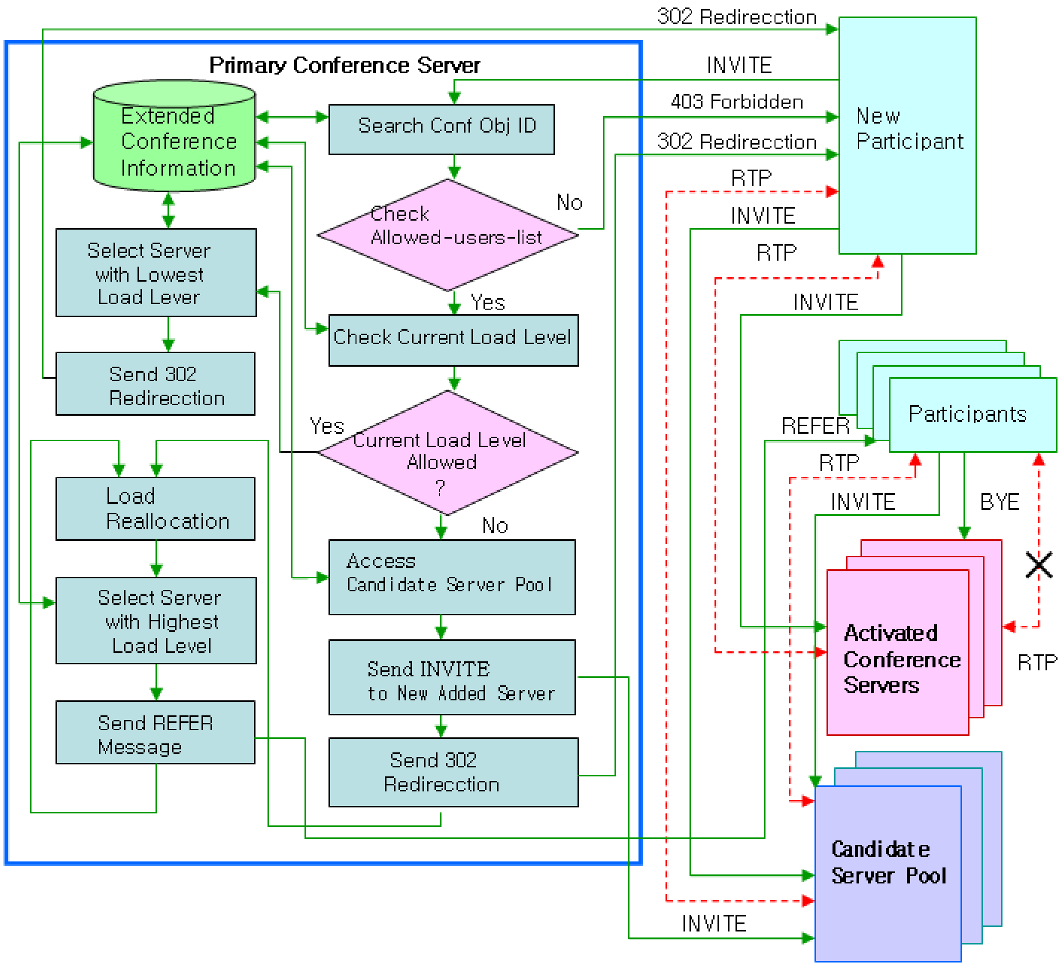

Figure 2. Signaling Procedure of the Proposed Distributed Conferencing System

In this procedure, when a participant requests to join the conference using SIP INVITE message, the primary conference server searches extended conference information data to find conference object identifier of the participant from its SIP URI, and determines whether the participant should be allowed using <allowed-users-list> element of the conference object. Then, the primary conference server checks current load values of all the activated servers, and select the server with the lowest current load value. Then, it sends SIP redirection message which has URI of this selected server to the participant. The participant makes SIP session with this server by sending SIP INVITE message, and it establishes RTP session to the server for exchanging media stream. When all the activated servers show high load value, a new server is required, and the primary server selects a new server from the candidate servers list. The primary conference server sends SIP INVITE message to the new server to establish conference session with it, and sends SIP redirection message with URI of this new added server to the participant. When the participant receives redirection message, it establishes SIP session and RTP session with the new added server.

Then, load reallocation process begins. The primary server sends SIP REFER message to the participants connected to the server which has highest load value. When the participants receives SIP REFER message, they send SIP INVITE requests to the new added server to make conference session with it. After establishing SIP and RTP session with the new added server, the participants send SIP BYE messages to their prior sever to release the conference sessions with it. This load reallocation process continues until load levels of all the active servers decrease to be nearly same value.

3.2. Design of Extended Conference Information Data Format

In this paper, a new extended conference information data format with some added elements has been designed to implement distributed conferencing system architecture. In order to control conference system load, <load-level> element is added as a child element of <conference-info> which is the root element of the original conference information data format. This <load-level> element contains information about load level in each conference server. The proposed load level L_i is calculated as :

$$L_i = \alpha N_i + \beta \sum_{j=1}^{N_i} (V_{ij} + A_{ij}) \qquad (1)$$

where index i represents each conference server, N_i is number of participants for each conference server, V_{ij} and A_{ij} represent video/audio frames generated per second according to RTP payload types that are negotiated via SDP messages. The parameters α and β are weighting factors. Each of these elements has three attributes, which are 'max-load-level' for representing maximum allowed load level, 'min-load-level' for representing minimum allowed load level, and 'server-id' for identifying each server. The value of 'max-load-level' for primary server can be lower than value of the other servers by conference policy. When a server's load level is below the minimum allowed load level, and this server's load can be distributed to the other servers, then the server becomes to be deactivated state. The element <load-level-components> has been designed to represent each component of load level value. This element has one attribute, 'server-id' for identifying each server, and two child elements such as <participants-numbers> and <media-frames>. The element <weighting-factors> has also been added to represent each load weighting factors described above. This element has two child elements, <alpha> and <beta>, to represent each load weighting factor.

The element <reserved-servers-list>, <server-num> and <current-participants-list> are also designed in this study. The element <reserved-servers-list> has list of candidate servers that can be added to the conference. It has a child element <uri> to represent each server. The <server-num> element represents number of active servers in conference system. The <current-participants-list> element represents active users list of each server, and it has a child element <uri> to represent each participant, and an attribute 'server-id' to identify each server.

The element <conf-control> has been added to control conference object by authorized participants. The authorized participant can create conference object, add participants to the created conference object, update participant's information, delete participant, and update conference object information by using this new <conf-control> element. This element has a child element <operation> to control the conference object. The <operation> element has 'conf_create', 'user_add', 'user_update', 'user_delete', 'conf_update' and 'conf_delete' as it's values. Figure 3 shows main part of the XML schema definition for extended conference information data format designed in this paper.

```
     <?xml version="1.0" encoding="utf-8"?>
  <xsd:schema
           …………..
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
          …………..
        <!-- load-level definition -->
        <xsd: element name=" load-level"  minOccurs="1"  maxOccurs=" unbounded " />
<xsd: complexType>
```

```
      <xsd: simpleContent>
         <xsd:extention base=" xsd: unsignedInt" >
<xsd:attribute   name=" max-load-level"   type=" xsd:unsignedInt"   use="required" />
<xsd:attribute   name=" min-load-level"   type=" xsd:unsignedInt"   use="required" />
<xsd:attribute  name=" server-id"  type=" xsd:anyURI"
   use="required" />
                    </xsd:extention >
      </xsd: simpleContent>
  </xsd:complexType>
</xsd: element>

<!—load-level-components definition -->
            <xsd: element name=" load-level-components "  minOccurs="1"
            maxOccurs=" unbounded " />
  <xsd:complexType>
     <xsd:sequence>
       <xsd:element   name="participants-numbers"  type=" xsd: unsignedInt" />
       <xsd:element   name="media-frames"   type=" xsd: unsignedInt" />
    </xs:sequence>
 <xsd:attribute  name=" server-id"   type=" xsd:anyURI "  use="required" />
  </xsd:complexType>
</xsd: element>

<!—weighting-factors definition -->
            <xsd: element name=" weighting-factors " />
  <xsd:complexType>
     <xsd:sequence>
        <xsd:element   name="alpha"  type=" xsd: double" />
       <xsd:element    name="beta"  type=" xsd: double" />
     </xs:sequence>
 </xsd:complexType>
</xsd: element>

<!—reserved-servers-list definition -->
            <xsd: element name=" reserved-servers-list" />
  <xsd:complexType>
     <xsd:sequence>
        <xsd:element    name="uri"   type=" xsd: anyURI"   minOccurs="1"
                    maxOccurs="unbounded" />
     </xs:sequence>
   </xsd:complexType>
</xsd: element>

<!—server-num definition -->
            <xsd: element name=" server-num " type=" xsd: unsignedInt " />

<!—current-participants-list definition -->
<xsd: element name=" current-participants-list "  minOccurs="1"          maxOccurs="1" />
  <xsd:complexType>
     <xsd:sequence>
        <xsd:element    name="uri"   type=" xsd: anyURI"   minOccurs="1"   maxOccurs="
                    unbounded " />
     </xs:sequence>
<xsd:attribute  name=" server-id"   type=" xsd:anyURI "   use="required" />
   </xds:complexType>
</xsd: element>

<!—conf-control definition -->
<xsd:element  name=" conf-control "  >
```

```
   <xsd:complexType>
     <xsd:sequence>
         <xsd:element   name=" operation"   type=" operationType" />
<xsd:element   name=" conf-subject"   type=" xsd: string"   minOccurs="0"   maxOccurs="1" />
      </xs:sequence>
   </xsd:complexType>
</xsd: element>

   <!-- operationType definition -->
      <xsd:simpleType name="operationType">
         <xsd:enumeration value="conf_create "/>
         <xsd:enumeration value="user_add"/>
         <xsd:enumeration value="conf_update"/>
         <xsd:enumeration value="user_update"/>
         <xsd:enumeration value="user_delete"/>
         <xsd:enumeration value="conf_delete"/>
      </xs:simpleType>
           …………..
```

Figure 3. XML Schema Definition for Extended Conference Information Data Format

3.3. Exchange Procedure of SIP Messages between Conference Servers and Participants

In centralized conferencing system, the original conference event package uses SIP SUBSCRIBE/NOTIFY messages to exchange conference information between SIP user-agent[10]. In this paper, in order to synchronize conference information between servers in distributed conferencing environment, the conference event package has been extended by using some added information data format elements as described on Section 3.2. Followings are the exchange procedures of SIP messages between conference servers and participants using the extended conference event package.

Figure 4 shows exchange procedure of SIP messages between conference servers and participants. In this exchange procedure, each conference server subscribes to primary conference server using the extended conference event package, and primary conference server also subscribes itself to the other servers. When a participant(participant A) sends SIP INVITE message to the primary conference server, it checks current load level of each active server to find a server with the lowest load level. If load level of the server(conference server C) is within the maximum allowed load level, the primary server sends SIP redirection message with URI of this server to the participant.

Then, the participant sends again SIP INVITE message to this server to establish SIP session, and makes RTP session with the server to exchange media stream. The server(conference server C) notifies this change to the primary server, and primary server notifies to other servers the change of conference information to synchronize the overall conference information of the system. The added participant sends SIP SUBSCRIBE message to the server(conference server C) to subscribe conference event package, then it receives current conference information from the server. Then, other participant(participant B) sends request message to join the conference, and in this case, the primary server select conference server B, and sends SIP redirection message with URI of this server to participant B. Then, the participant establishes conference session by sending SIP request message to this server, and conference server B processes media streams from participant B through RTP session. The conference server B notifies this conference information change to the primary server, and the primary server sends NOTIFY messages to the other servers.

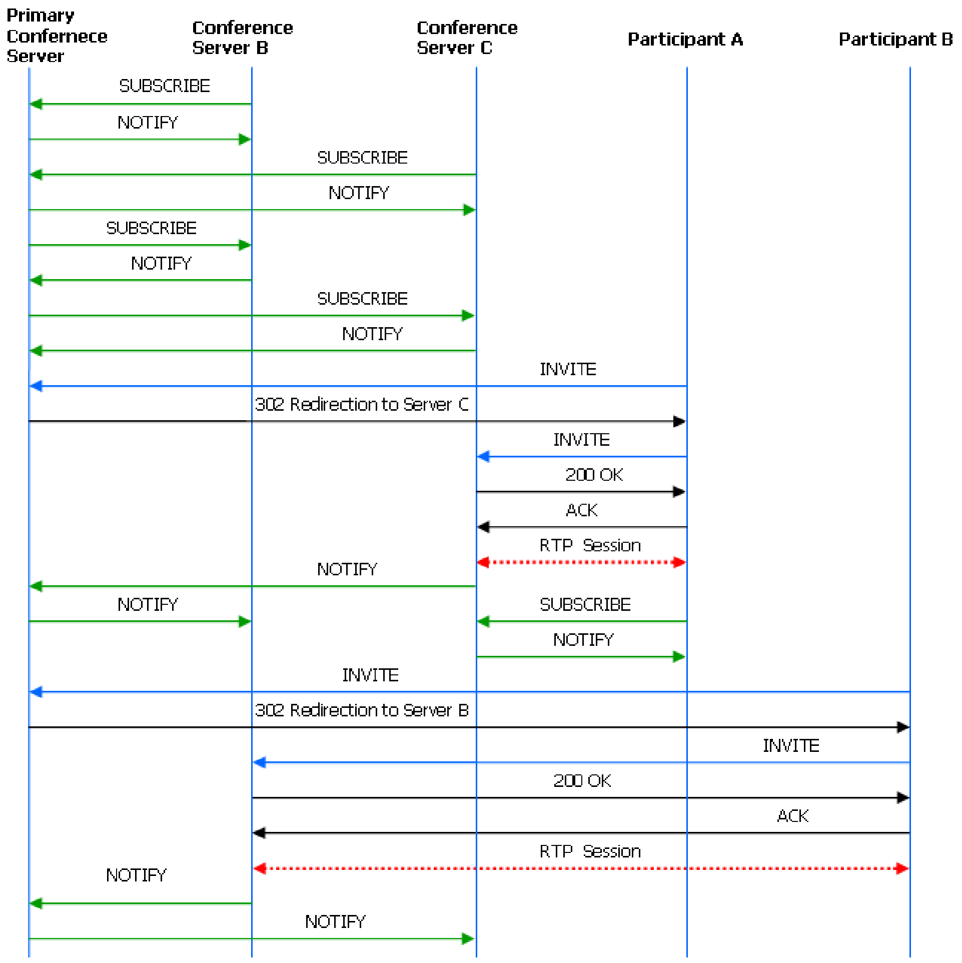

Figure 4. Exchange Procedures of SIP Messages between Conference Servers and Participants

Figure 5 shows exchange procedure of SIP messages when a new conference server is added to process participant's request. When the primary server determines to add a new server after checking current load levels of all active servers, it selects candidate sever from reserved servers list, and sends SIP INVITE message to this selected new server to establish conference session with it.

Then, the new server sends SIP SUBSCRIBE message to the primary conference server to subscribe extended conference event package. The primary conference server sends SIP NOTIFY message which contains current conference information of the system in its body to the new server, and by this way the new server can share current conference information with the other active servers. The primary conference server sends SIP SUBSCRIBE message to the new added server to subscribe itself to this new server to get conference information changes from this server. Then, the new added server makes RTP sessions with the other servers, and the primary conference server sends SIP redirection message with URI of this new added server to the participant (participant C). The participant makes SIP session and RTP session with the new added server.

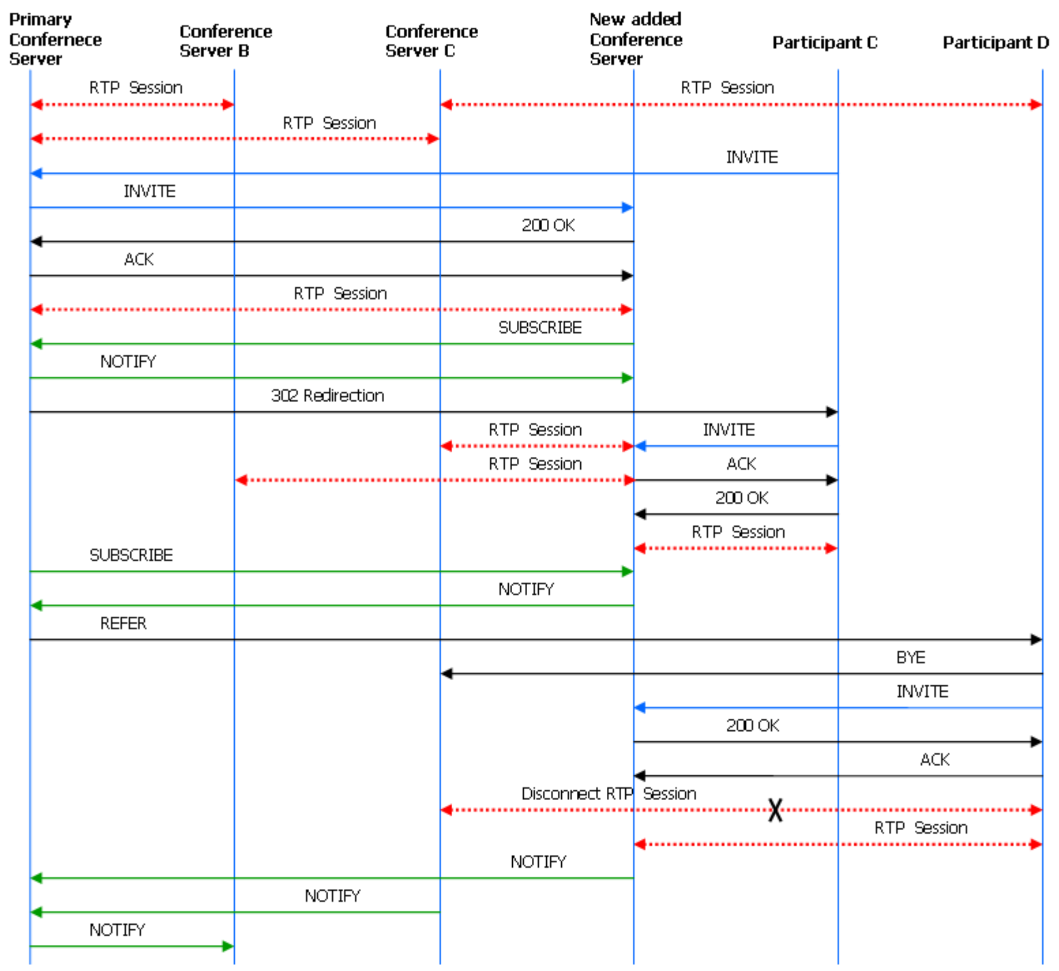

Figure 5. Exchange Procedures of SIP Messages when a New Server is Added

Then, load reallocation process begins. In this figure, the primary server sends SIP REFER message to the participant D who is connected to the server which has higher load level value than the other servers. When participant D receives SIP REFER message, it sends INVITE request to the new added server to make SIP conference session with it. After establishing SIP session with the new added server, participant D sends SIP BYE message to its prior server(conference server C) to release the conference session with it. Then, participant D disconnects RTP session from its prior sever, and establishes new RTP session to the new added server. The conference server C and new added server notify their conference information changes to the primary server, and finally, the primary server sends NOTIFY message to other servers to synchronize the overall conference information of the system.

Figure 6 shows exchange procedures of SIP messages when an authorized participant (participant A) requests to the primary conference server to create a conference object. Some SIP messages are omitted to simplify the figure. The authorized participant sends INVITE message with Allow-Events header which has value ‘x-conf’ representing the proposed extended conference event package. The body part of INVITE message has extended conference information element <conf-control>, as described on Section 3.2, with child element <operation> which has value ‘conf_create’ representing creation of conference object. The body part has also <allowed-users-list> element with child element <target>. The element <target> has attribute ‘uri’ representing participant’s SIP URI, and attribute ‘method’ representing conference joining method.

The primary conference server creates conference object, and selects a server(conference server B) as a focus of the created conference object. And the primary conference server sends SIP NOTIFY message which has conference information of the created conference object in its body to conference server B. The primary conference server also sends a created conference object ID to participant A. Then, the conference server B sends INVITE message to each participants in <allowed-users-list> element, and establishes SIP session and RTP session. Each participant subscribes to the conference server B and receives conference information.

The authorized participant (participant A) sends INVITE message which has conference object ID, <operation> element with value 'user_add', and new participant's SIP URI in the body part. In this case, the primary conference server decides to assign a new conference server to serve the new participant due to high load level of conference server B. It selects candidate sever from the reserved servers list, and sends SIP INVITE message to this selected new server to establish conference session with it. Then, the new added server subscribes to the primary conference server, and it responds with notification which has current conference information of the system. Then, the new added server establishes SIP conference session with the new participant(participant D). The new added server notifies changes of conference information to the primary conference server, and the primary conference server sends NOTIFY messages to the other servers. Then, the conference server B notifies changes of conference information about new participant to the other participants.

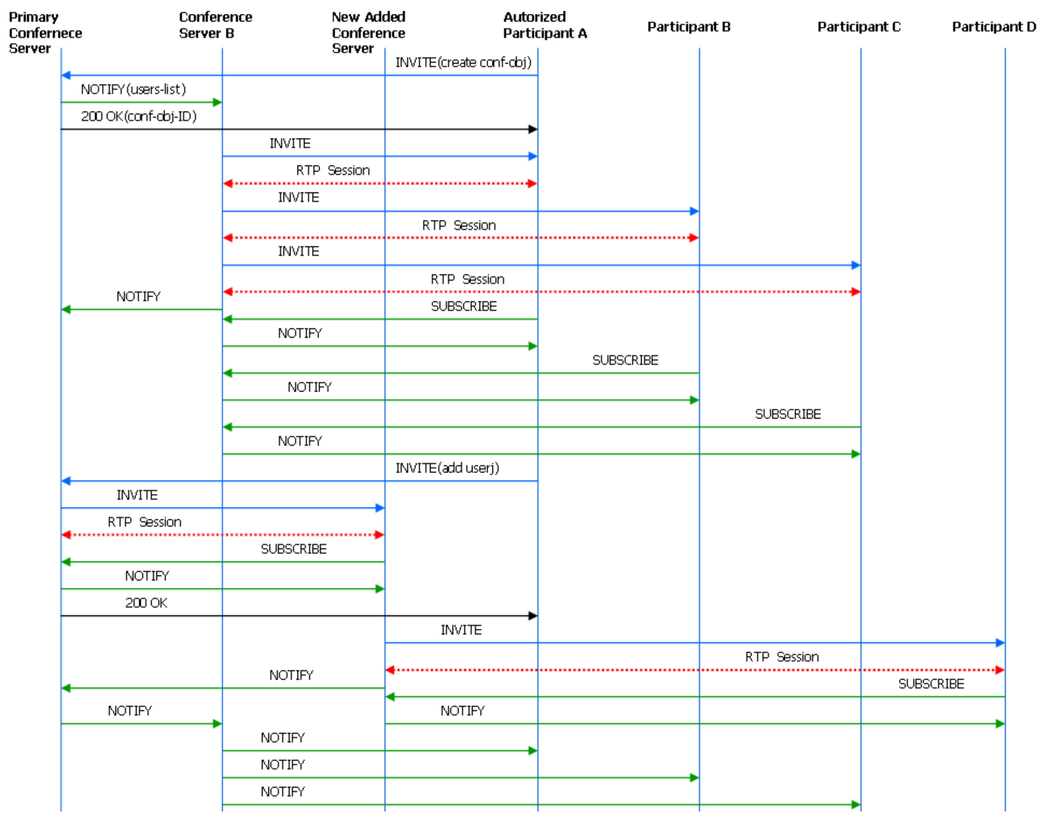

Figure 6. Exchange Procedures of SIP Messages when an Authorized Participant Requests to Create Conference Object

4. Performance Evaluation

In the experiments, 10 personal computers are used to emulate conference participants. Each personal computer has been programmed to generate workload up to 30 conference participants. For acting as conference servers, 7 personal computers are used. SIP protocol stack and packet generator are written by JAVA in MS-Windows and LINUX environment. SIP protocol stack sources are written according to SIP standard, and packet generator is configurable and extensible via XML. All personal computers are interconnected using a gigabit Ethernet switch. Conference object ID has been assigned for every 10 participants, so total number of conference objects used for tests is 30.

The average media packet transmission delay and the average SIP signaling delay are measured. The performance of the proposed distributed architecture has been compared with the centralized conferencing architecture of IETF [11]. Table 1 shows detailed parameters used for experimental measurements. In order to measure media packet transmission delay time, audio signal encoding method CS-ACELP G.729 Annex A which has a payload type 18 defined in RTP profile has been used. The sampling frequency for encoding signal is 8 KHz, and 15 packets are generated every second. The frame length is 10 msec, packet generation period is 66.67 msec, and packet duration is 20 msec.

Table 1. Parameters Used for Experimental Measurements

Parameters	Value
sampling frequency	8000 Hz
frame length	10 msec
number of packets per second	15
packet generation period	66.67 msec
encoding rule	CS-ACELP G.729A
number of bits per frame	80
packet duration	20 msec
bit generation speed	8000 bps
number of conference objects	30
network bandwidth	1000 Mbps

Figure 7 shows average media packet transmission delay as the number of participant increases from 30 to 300. The average media packet transmission delay includes processing time for mixing media packets in the mixer, delay time for media packet transmission, and delay time for generating media packet. The value of the maximum load level, ‘max-load-level’, which is an attribute of <load-level> element defined in Section 3.2, is set to be 600. The value of weighting factor parameter α is set to be 10, and β is set to be 0.001. In this figure, the average media packet transmission delay of the proposed architecture is measured much smaller than that of the centralized architecture. This figure shows that as the number of participants increases, the average media packet transmission delay of the proposed architecture remains almost at constant, while delay of the centralized architecture grows rapidly. This is because the whole media streams are concentrated to a single conference server in the centralized architecture, while in the proposed architecture, the media streams are distributed to multiple servers dynamically according to load level of each server. In this figure, transmission delay of the proposed architecture grows slightly when the number of participants exceeds 210. This is because new conference server is added for every 30 participants according to the value of the maximum load level and weighting factor parameters. So there are no further reserved servers when the number of participants exceeds 210, and each server's load slightly increases.

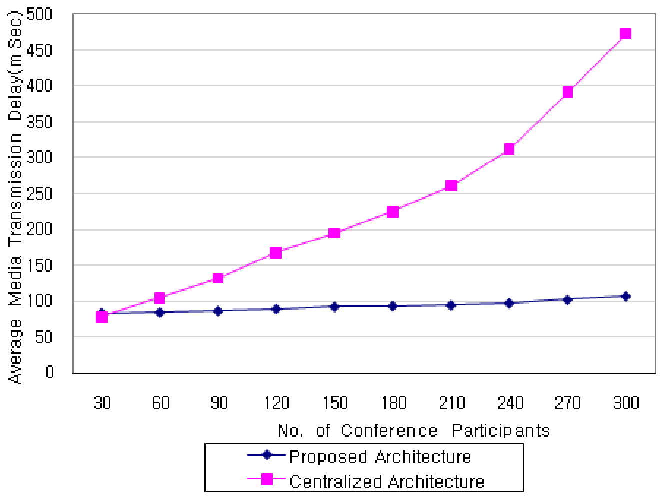

Figure 7. Average Media Packet Transmission Delay

Figure 8 presents average SIP signaling delay when participants join a conference. The delay is measured by completion time of SIP INVITE transaction to the conference server. The testing parameters for the proposed architecture are same as in Figure 7, and only SIP sessions are used in the centralized architecture. When the number of participants is 30, one server is used in both cases, and the proposed architecture shows somewhat higher delay due to media processing load. As the number of participants increases beyond 30, the proposed architecture shows higher delay time. This is because additional SIP messages are required to complete the transaction during the redirection procedures of adding conference servers as described in Section 3.3.

When the number of participants exceeds 120, the average SIP signaling delay of the centralized architecture increases continuously. This is because server load required to keep conference sessions with all participants also increases even without media processing. However, the average SIP signaling delay of the proposed architecture remains almost constant with increasing number of participants due to distribution of server load to multiple servers by dynamic server allocation, and this fact improves scalability of the large-scale conference system.

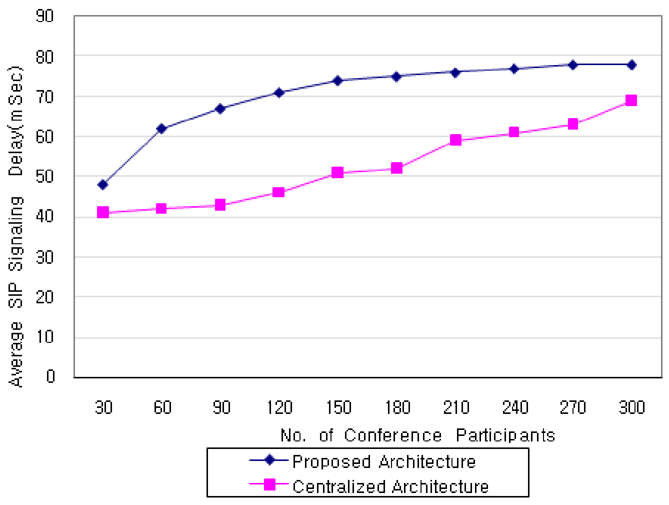

Figure 8. Comparison of Average SIP Signaling Delay

Figure 9 shows comparison of average media packet transmission delay of the proposed architecture with different values of the maximum load level, 300 and 1200. The weighting factor parameters are same as in test for Figure 7. In this test, when maximum load level is 300, new conference server is added for every 15 participants from the equation (1). When maximum load level is 1200, new conference server is added for every 60 participants.

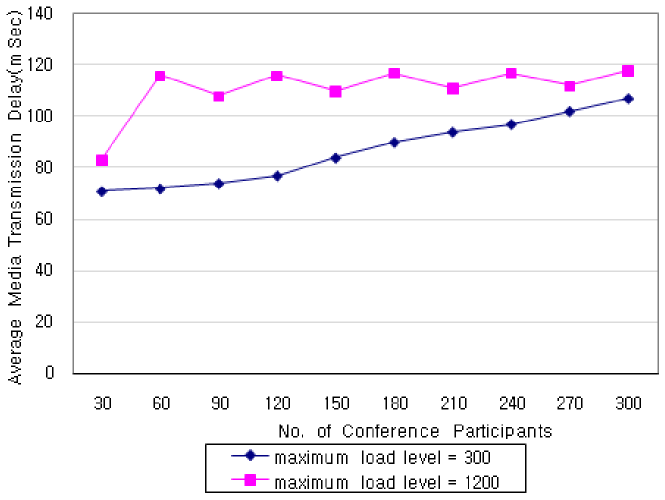

Figure 9. Average Media Packet Transmission Delay with Different Maximum Load Level

In this figure, when maximum load level is 300, the average media packet transmission delay is measured nearly constant until the number of participants reaches 90. When the number of participants exceeds 120, transmission delay gradually increases. This is because new conference server is added for every 15 participants, so there is no further reserved servers when the number of participants exceeds 120. In this case, the load of the system has been equally distributed to all conference servers, so the load per server gradually increases. When maximum load level is 1200, the average media packet transmission delay shows some fluctuation around 115 msec when the number of participants exceeds 60. This is because new conference server is added for every 60 participants, and the number of participants allocated to each server fluctuates around 55 after 60 participants. This result presents that load balancing of the system can be easily controlled by changing load level value.

Figure 10 shows effect of media frames generated per second according to RTP payload types. Equation (1) in Section 3.2 describes about that. In the experiment, PCMμ encoding method has been used for 30% of participants, and G.729A has been used for 70% of participants in one case. In the other case, all participants use G.729A. The maximum load level and weighting factor parameters are same as in test for Figure 7. In this figure, the average media packet transmission delay of using 30% PCMμ shows nearly constant, and somewhat lower than the transmission delay of using 100% G.729A until the number of participants reaches 90. However, when the number of participants exceeds 120, transmission delay of using PCMμ increases somewhat rapidly. This is because the sum of PCMμ media frames generated per second is 8 times higher than that of G.729A, therefore, the load level becomes higher when PCMμ is used. So in this case, all servers are allocated when the number of participants reaches 120, and the load per server increases after that.

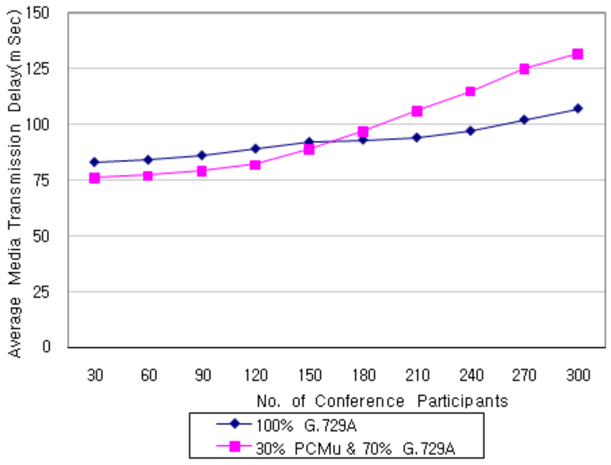

Figure 10. Effect of Media Frames Generated per Second According to RTP Payload Types

Figure 11 shows effect of weighting parameter β in the load level equation (1). In this experiment, 30% of participants assumed to use PCMμ encoding method as in Figure 10. The value of weighting parameter β is changed from 0.001 to 0.00035 to decrease the effect of media frames generated per second on the load level. As shown in Figure 11, the average media packet transmission delay of using weighting parameter β = 0.00035 and 70% G.729A remains almost constant until the number of participants becomes to be 210. This is due to the fact that new conference server is allocated for every 30 participants by reducing the value of weighting parameter β in the case of using 70% G.729A. In this figure, the average media packet transmission delay of using 100% G.729A shows somewhat lower value than that of using 70% G.729A, this is because the amount of media frames generated per second is lower than that of using 70% G.729A, so the load level is lower at the same number of participants. When the value of weighting parameter β is 0.001 and 30% PCMμ encoding method is used as in Figure 10, conference server is allocated for every 15 participants due to highly generated load level. Therefore, there is no further reserved servers when the number of participants exceeds 120, and the transmission delay increases somewhat rapidly after that. However, by changing weighting parameter value, transmission delay can be controlled to remain almost constant until the number of participants becomes to be 210 in this experiment.

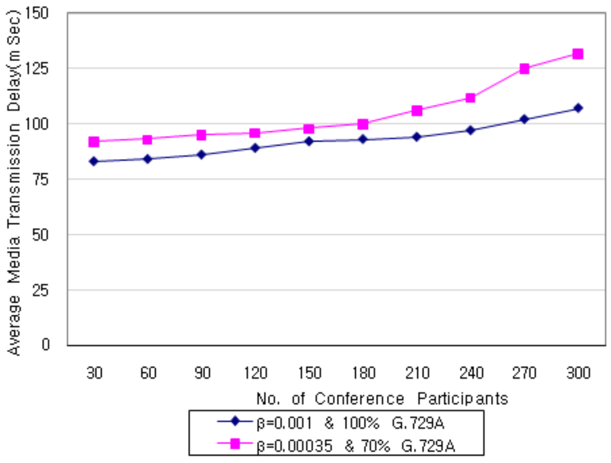

Figure 11. Effect of Weighting Parameter β

5. Conclusion

In this paper, distributed large-scale conferencing system architecture which has effective dynamic server allocation capability has been proposed. A new extended conference information data format has been designed to distribute system load dynamically to multiple conference servers. The extended conference information data format has new added elements for controlling conference system load and conference object. To synchronize conference information data between conference servers, the extended conference event package has also been used. The amount of video frames and audio frames generated per second according to RTP payload types and the number of participants for each conference server is used for load level control. The presented exchanging procedures of SIP messages show detailed SIP request and response signals transferred between conference servers and participants to dynamically distribute system load.

Experimental measurements have been done to verify performance of the proposed distributed conferencing system architecture. The experiments measure average media packet transmission delay and the average SIP signaling delay by changing the number of participants, load level values, and weighting factor parameters. The results show that the proposed architecture can provide high scalability and flexibility for distributed large-scale conferencing system. Future work may include security issues which would adapt advanced security mechanism for implementing distributed large-scale conferencing system architecture.

Acknowledgements

This paper was supported by Research Fund, Kumoh National Institute of Technology.

References

[1] W. Su, C. Bo and J. Chen, "The design and implementation of sip control for Multimedia Conference system", in Proc. of International Conf. on Circuits, Communications and System, **(2010)** August 1-2; Vouliagmeni, Italy.
[2] R. Shekh-Yusef and M. Barnes, "Indication of Conference Focus Support for the Centralized Conferencing Manipulation Protocol", RFC 7082, **(2013)**.
[3] J. Zhang Yanyan and Yao Yuan, "SIP-based multimedia conference system design and implementation", in Proc. of International Conf. on Computer Design and Applications, **(2010)** June 25-27; Amsterdam, Netherlands.
[4] J. Li, W. Lei and X. Zhang, "Design and Implementation of a SIP-Based Centralized Multimedia Conferencing System," in Proc. of International Conf. Communication Software and Networks, February 27-28, **(2009)**; Macua, China.
[5] L. Wang and S. Jun, "SIP-Based Multimedia Conference Management System and Its Application", in Proc. of International Conf. on Information Engineering, **(2009)** July 10-11; Taiyuan, China.
[6] Y.-H. Cho, M.-S. Jeong, J.-W. Nah, W.-H. Lee and J.-T. Park, "Policy-based distributed management architecture for large-scale enterprise conferencing service using SIP," IEEE Journal on Selected Areas in Communications, vol. 23, no. 10, **(2005)**, pp. 1934-1949.
[7] J. Rosenberg and H. Schulzrinne, "Conference Information Data Model for Centralized Conferencing (XCON)," RFC 6501, **(2012)**.
[8] A. Knauf, G. Hege, T. C. Schmidt and M. Wahlisch, "A virtual and distributed control layer with proximity awareness for group conferencing in P2PSIP," in Proc. of IPTComm, **(2010)** August 2-3; Munich, Germany.
[9] C. Jennings, B. Lowekamp, E. Rescorla and H. Schulzrinne, "REsource LOcation and Discovery Base Protocol," RFC 6940, **(2014)**.
[10] G. Camarillo, S. Srinivasan, R. Even and J. Urpalainen, "Conference Event Package Data Format Extension for Centralized Conferencing (XCON)," RFC 6502, **(2012)**.
[11] M. Barnes, C. Boulto and O. Levin, "A Framework for Centralized Conferencing," RFC 5239, **(2008)**.
[12] [12] M. Barnes, C. Boulton, S. Romano and H. Schulzrinne, "Centralized Conferencing Manipulation Protocol", RFC 6503, **(2012)**.
[13] M. Barnes, L. Miniero, R. Presta, S. P. Romano and H. Schulzrinne, "CCMP: a novel standard protocol for Conference Management in the XCON Framework," in Proc. of IPTComm, **(2010)** August 2-3; Munich, Germany.

Author

Choonseo Jang, He received the B.S. degree from the Seoul National University, Seoul, Korea, in 1978, and received the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejon, Korea, in 1981 and 1993, respectively. From 1981 He is a professor at department of computer engineering, Kumoh National Institute of Technology, Gumi, Korea. His research interests include Session Initiation Protocol and Voice over IP.