A SOAP Web Services-Based Architecture for Floor Control in Multimedia Conferencing

Jagdeep Singh

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Applied Science at Concordia

University

Montréal, Québec, Canada

October 2012

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Jagdeep Singh

Entitled: "A SOAP Web Services-Based Architecture for Floor Control in Multimedia Conferencing" and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

Complies with the regulations of this University and meets the accepted

standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. F. Khendak

_____ Examiner, External
Dr. J. Rilling (Computer Science) To the Program

_____ Examiner
Dr. A. Agarwal

_____ Supervisor
Dr. R. Glitho

Approved by: _____
Dr. W. E. Lynch, Chair
Department of Electrical and Computer Engineering

_____20_____ _____
Dr. Robin A. L. Drew
Dean, Faculty of Engineering and
Computer Science

ABSTRACT

A SOAP Web Services-Based Architecture for Floor Control in Multimedia Conferencing

JAGDEEP SINGH

Multimedia conferencing applications are an important and widely-used category of Web applications. Floor control is a significant and advanced feature of multimedia conferencing applications. Floor control mechanisms, when introduced in audio/video conferencing, control the media streams such as identifying which participant is allowed to send and who can be seen or heard. This prevents conflict and ensures an optimized use of resources between the conference participants. Floor control is composed of three logical entities: a single floor control server (i.e. entity responsible for managing the floors and their status), one or more floor chairs (moderators), and any number of regular conference participants.

This thesis proposes a SOAP Web services based architecture for floor control in multimedia conferencing. Web services are designed to support interoperable machine-to-machine interaction over a network. They are attractive because of their flexibility. There are two types of web services: SOAP Web services and RESTful Web services. In SOAP Web services, interactions between the entities are based on XML and use SOAP, which is embedded in HTTP. RESTful web services are an architectural design style that rely on HTTP, but do not use SOAP.

XML is also optional. We propose a set of floor control requirements and use them to review the related work and pinpoint the weaknesses. The proposed architecture includes the main components of floor control. It also includes a comprehensive set of server-side and client-side SOAP web service APIs that expose the floor control capabilities to application developers. The proposed APIs are programming language-independent and provide a higher level of abstraction to the application developers, which enables the interoperability. Furthermore, in the proposed architecture the floor control clients do not interact directly with the floor control server (FCS) but through a gateway accessible using SOAP web services. This opens up the possibility to use different floor control protocols transparently to the floor control clients. Application portability is no longer a problem because floor clients access the floor capabilities independently of the protocol supported by the FCS.

We have built a conferencing application with floor control as a proof of concept to demonstrate the new interface for floor control and the feasibility of the proposed architecture. In addition, performance measurements have also been made to evaluate the viability of the architecture.

Acknowledgments

I wish to express my sincere gratitude and appreciation to all people who helped me made this master thesis possible. First and foremost, I dedicate special thanks to my supervisor Dr. Roch Glitho who kept an eye on my progress and supported me with sound advices and discussions. His words of wisdom always oriented me in the right direction or showed me different perspectives and I am very grateful for that.

I also offer my gratitude to Dr. Fatna Belqasmi for all her support, ideas and comments throughout my work. Her positive attitude and enthusiasm helped me a lot along the way.

I'm grateful to Dr. F. Khendek, Dr. Agarwal and Dr. Rilling for serving as members of my thesis committee.

I am grateful to Dr. Roch Glitho and Concordia University for their financial support.

Last but not least, my personal thanks go to my dearest friends and family who never ceased to give me moral support and were there when I needed them. You always understood and supported the choices I made during this long and personal journey

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

3GPP	3rd Generation Partnership Project
API	Application Programming Interface
AS	Application Server
BFCP	Binary Floor Control Protocol
XML	Extensible Markup Language
FCS	Floor Control Server
FCFS	First Come First Serve
FSCML	Floor Server Control Mark-up Language
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force

IMS	IP Multimedia Subsystem
IETF	Internet Engineering Task Force
IVR	Interactive Voice Response
MS	Media Server
MSCML	Media Server Control Mark-up Language
MRFP	Media Resource Function Processor
MRFC	Media Resource Function Controller
OSA	Open Service Access
OSA GW	Open Service Access Gateway
OSA SCS	Open Service Access Service Capability Server
OMA	Open Mobile Alliance
OEPE	Oracle Enterprise Pack for Eclipse
PoC	Push to talk over cellular
RTP	Real-time Transport Protocol
RTCP	RTP Control Protocol

REST	Representational State Transfer
RPC	Remote Procedure Call
ROA	Resource Oriented Architecture
SIP	Session Initiation Protocol
SDP	Service Delivery Platform
SOAP	Simple Object Access Protocol
TBCP	Talk Burst Control Protocol
TCP	Transport Control Protocol
URI	Uniform Resource Identifier
UE	User Equipment
UDP	User Datagram Protocol
UDDI	Universal Description, Discovery and Integration
WLAN	Wireless Local Area Network
WSDL	Web Services Description Language

Chapter 1

Introduction

This chapter first presents an overview of the research domain, followed by the motivations and problem statement. Next, it presents the thesis contributions. The last section presents the thesis organization.

1.1 Research Domain

Multimedia conferencing is an important category of Web applications. It is the basis of a wide range of applications including audio/video conferencing, gaming and distance learning. Conference control [2] is a core building block of multimedia conferencing. It includes conference management, membership control and floor control. Conference control has been an area of intensive research over the years. In this thesis we focus on the floor control feature of conference control and how to expose this feature to multimedia conferencing application developers.

1.1.1 Floor Control in Multimedia Conferencing

Floor control [1] is a significant and advanced feature of multimedia conferencing applications. Resources (e.g. audio/video channels, slide bar presentation) are usually shared in conferencing. Floor control is used to manage the joint or

exclusive access to these shared resources. It prevents conflict, degradation of quality of service (e.g. ten people talking at the same time) and ensures an optimized use of resources. The model for floor control is composed of three logical entities: a single floor control server, one or more floor chairs (moderators), and any number of regular conference participants. The floor control messages are conveyed between the floor chairs (moderators) of the conference, the floor control server (FCS), and the participants of the conference. A centralized architecture is assumed in which all messages go via one point: the FCS. Processing (granting or rejecting) floor control requests is done by the one or more floor chairs or by the server itself, depending on the policy (i.e. Chair-controlled or Algorithm-based). For example, floor control mechanisms, when introduced in audio/video conferencing, control the media streams such as identifying which participant is allowed to send and who can be seen or heard. The participant having a floor can make the related media available to the other participants (i.e. audio/video) of the conference.

1.1.2 Web Services

Web services [9] in their simplest definition are programmatic interfaces that allow application-to-application communication over a network. Web service interfaces are attractive because they provide a higher level of abstraction as well as loose coupling between the interacting software components. Web services have been adopted in many application domains (telecommunications, digital imaging, e-commerce etc.). The reason to do so has mainly been ease of integration with other applications or with other business processes.

There are two types of web services: SOAP Web services [11] and RESTful Web services [17]. In SOAP Web services, interactions between the entities are based on XML and use SOAP which is embedded in HTTP. RESTful web services are an architectural design style that rely on HTTP, but do not use SOAP. XML is also optional. The mentioned advantages of web services make them an attractive solution to expose the floor control capabilities in multimedia conferencing while hiding their domain-specific details, enable interoperability and ease application development.

In this thesis we use SOAP Web services to build an architecture for floor control in multimedia conferencing.

1.2 Motivation and Problem Statement

Floor control is used in most applications which are based on multimedia conferencing (e.g. audio/video conferencing, gaming and distance learning). Floor control avoids chaotic situations when everybody attempts to use the resource (i.e. audio/video) at the same time. Furthermore, floor control improves the efficiency when bandwidth restriction is a concern.

Various standard bodies (e.g. 3GPP, IETF, Parlay) have proposed architectures that integrate floor control with conferencing. However, current mechanisms used for exposing the floor control capabilities have shortcomings that can hinder application development. Some of these mechanisms are programming language-dependent, others require extensive knowledge of network domain and its low level details, and few of them do not provide the comprehensive functionality

required for the floor control. In brief, it is difficult to integrate the floor control capabilities in applications. Furthermore in the state of art, the floor client interacts directly with the floor control server (FCS). This constrains the possibility of using different floor control protocols transparently to the floor clients. Also, the client application remains no longer portable because if the FCS is replaced and the new FCS supports different protocols, the client application would have to be upgraded. .

Consequently, the motivation of the thesis is to provide a framework for floor control which enables interoperability and portability. This thesis established that a SOAP Web service based framework is the most promising, as web services provide a higher level of abstraction and integrate easily with other applications while being programming language and platform neutral.

1.3 Thesis Contributions

The contributions of the thesis are as follows:

- A set of requirements for floor control in multimedia conferencing.
- A review of the state of the art relevant to our work with an evaluation summary comparing to our requirements.
- A proposal for a novel SOAP web services based floor control architecture in multimedia conferencing that meets all our requirements
- SOAP web services based APIs for floor control that extend the existing Parlay-X (SOAP-based) multimedia conferencing web service

functionality with floor control capabilities, including a comprehensive set of server-side and client-side APIs that expose the floor control capabilities to application developers

- An implementation architecture and a proof of concept prototype
- A preliminary performance evaluation of the proposed architecture

1.4 Thesis Organization

The rest of thesis is organized as follows:

Chapter 2 discusses the concepts and definitions related to floor control in multimedia conferencing and Web services (with a main focus on SOAP-based web services) that will illustrate to the reader the basic ideas relevant to this thesis.

Chapter 3 introduces the requirements for floor control in multimedia conferencing, followed by the state of the art related to floor control in multimedia conferencing applications and SOAP-based web services. Furthermore, it presents the evaluation of related works comparing with our requirements.

Chapter 4 describes the proposed architecture for floor control in multimedia applications. It includes the main components for floor control, communication interfaces and the SOAP-based floor control interfaces.

Chapter 5 is dedicated to the implementation architecture of the system components. It presents the implemented proof of concept prototype and includes some performance measurements.

Chapter 6 concludes the thesis by briefly summarizing the overall contributions and suggesting some future work.

Chapter 2

Background on Floor Control in Multimedia Conferencing and Web Services

This chapter introduces the main topics which are relevant to this thesis research domain. The main topics introduced are floor control in multimedia conferencing and Web services.

2.1 Floor Control in Multimedia Conferencing

This section discusses floor control in multimedia conferencing. We start with a brief introduction of floor control in multimedia conferencing. Then we present the floor control model with some illustrative examples, followed by a sub-section that discusses integration of floor control with conferencing. Then the existing protocols involved in the floor control are introduced. Finally, protocols between the conferencing application server and floor control server (FCS) are explained.

2.1.1 Introduction

Multimedia conferencing applications are an important and widely-used category of Web applications. It is the basis of a wide range of applications including audio/video conferencing, gaming and distance learning. Conference control [2] is a core building block of multimedia conferencing, which includes conference

management, membership control and floor control. Resources (e.g. audio/video channels, slide bar presentation) are usually shared in conferencing.

Floor control [1] is used to manage the joint or exclusive access to these shared resources (e.g. audio/video) in the conference. It prevents conflict, degradation of quality of service (i.e. ten people talking at the same time) and ensures an optimized use of resources.

For example, floor control mechanisms, when introduced in audio/video conferencing, control the media streams, such as identifying which participant is allowed to send and who can be seen or heard. This prevents the access conflicts between the conference participants.

The usage of the resources can also be optimized by setting the number of participants who can hold the floor (i.e. share the resource) at the same time, depending on the available bandwidth. Furthermore, the participants make separate requests to access different resources. For example, if a participant wants to talk he will request audio floor and if he wants to write he requests for text floor. Also, if he wants to write and speak simultaneously he can ask for the floor, which has both the medias associated with it.

2.1.2 Floor Control Model

The floor is an individual temporary access or manipulation permission for a specific shared resource (or group of resources).

The model for floor control [1] is composed of three logical entities: a single floor control server, one or more floor chairs (moderators), and any number of regular conference participants, as illustrated in Figure 2.1.

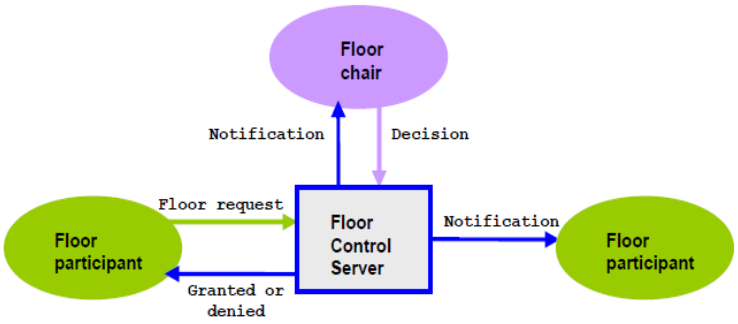

Figure 2.1: Floor control model

Floor Control Server (FCS): This logical entity maintains the floor status which includes information like who are the floor chairs, who holds the floor and which floor exists. It can inform the participants about the floor status. It can grant the floor depending on the floor policy adopted (e.g. if a non-chair policy is adopted).

Floor Participant: A conference participant entitled to request "right to speak" in form of a floor. Floor participant can request the floor from the FCS and will receive a grant or denied message back.

Floor Chair: A conference participant or an entity outside the conference who decides which participant can get the floor and when. It sends the decisions (e.g. floor accepted, revoked or granted) to the FCS.

Floor control mechanisms depend on the policy adopted for granting the floor. When the floor control policy is chair-moderated, then the decision to grant the floor is issued by the designated chairperson of the floor. However, if the floor policy is FCFS (first come first serve) or any other algorithm based, then the decision is made by FCS.

The following figure 2.2 illustrates a user requesting the floor to obtain the right to talk during a conference.

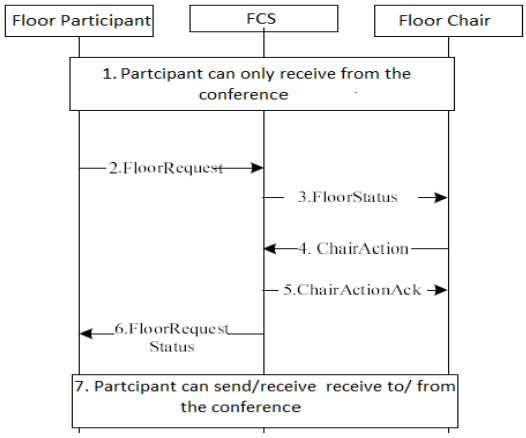

Figure 2.2: User requesting the floor to obtain the right to talk during a conference

2.1.3 Integrating Floor Control with Conferencing

Floor control itself does not support privileges such as creating/removing floors and appointing floor chairs [1]. Instead, some external mechanism such as conference management (e.g. internal Web-interface for policy manipulation) is used for that. The conference policy (and conference owner or creator) defines whether floor control is in use or not. In general, it is assumed that the conference policy defines who is allowed to create, change, and remove a floor in a conference. It is also the conference policy that defines which media streams may be used in a conference and which ones are floor-controlled. Typically, the conference owner creates the floor(s) using the conference policy control protocol (or some other mechanism) and appoints the floor chair.

The FCS is a separate logical entity. Therefore it can interact with the conferencing application server to stay updated with the latest floor information (e.g. a new floor is created, a new chairperson is designated, a floor is removed, and a participant is added to /removed from the floor) [16].

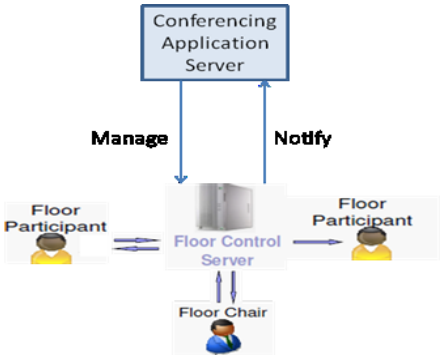

Figure 2.3: Integrating floor control with conferencing

In figure 2.3, the conferencing application server manages the FCS by providing the latest floor information. Additionally, the FCS can notify the conferencing server if any request related to the existing floor arrives from the floor client or chair.

2.1.4 Protocols Involved in Floor Control

The existing floor control protocols are Binary Floor Control Protocol (BFCP) and Talk Burst Control Protocol (TBCP).They are discussed in the following sub-sections.

2.1.4.1 Binary Floor Control Protocol (BFCP)

Binary Floor Control Protocol (BFCP) [3] is a protocol to coordinate access to shared resources in a conference. The Requirements for Floor Control Protocol [1] list a set of requirements that need to be met by floor control protocols. BFCP meets these requirements. It is used to convey the floor control messages among

the floor chairs (moderators) of the conference, the floor control server, and the participants of the conference. A centralized architecture is assumed in which all messages go via one point, the floor control server. Processing (granting or rejecting) floor control requests is done by one or more floor chairs or by the server itself, depending on the policy. BFCP mainly runs only over TCP, which makes it a reliable carrier. It uses binary encoding, resulting in smaller message sizes that helps to cope with incidents of low-bandwidth and transferring the delay-sensitive messages as opposed to textual protocols.

BFCP provides processes for:

- Floor participants to send floor requests to floor control servers.
- Floor control servers to grant or deny requests access to a given resource from floor participants.
- Floor chairs to send floor control servers decisions regarding floor requests.
- Floor control servers to keep floor participants and floor chairs informed about the status of a given floor or floor request.

Primitives provided by BFCP to support the floor control functionality are illustrated in figure 2.4

Primitive	*Description*	*Direction* *S – FCS* *P – Participant* *Ch – Floor Chair*
FloorRequest	Request a floor	P → S
FloorRelease	Release a floor or a pending floor request	P → S
FloorRequestQuery	Inquire information on floor request	P → S ; Ch → S
FloorRequestStatus	Inform of status of a floor request	P ← S ; Ch ← S
UserQuery	Inquire information on users and their floor requests	P → S ; Ch → S
UserStatus	Inform of participants and their floor requests	P ← S ; Ch ← S
FloorQuery	Inquire information on floors	P → S ; Ch → S
FloorStatus	Inform of floors	P ← S ; Ch ← S
ChairAction	The chair instructs the server	Ch → S
ChairActionAck	The server has accepted the ChairAction message	Ch ← S
Hello	Check the liveliness of the server	P → S ; Ch → S
HelloAck	The server is alive and accepted the Hello message	P ← S ; Ch ← S
Error	Errors in processing requests	P ← S ; Ch ← S

Figure 2.4: BFCP primitives [3]

However, the concrete floor creation, obtaining floor resource associations or information to contact a floor control server and floor control privileges are not in the scope of BFCP but are essential for the operation of the protocol.

BFCP connection parameters between the floor clients and FCS are negotiated using SDP offer/answer exchange [6].

Figure 2.5 shows how a floor participant requests a floor, obtains it, and, at a later time, releases it. This figure illustrates the use, among other things, of the Transaction ID the FLOOR-REQUEST-ID (i.e. represents a unique floor request) attribute. The other parameters like User ID identify the user requesting the floor, while FLOOR- ID identifies the unique floor which is requested.

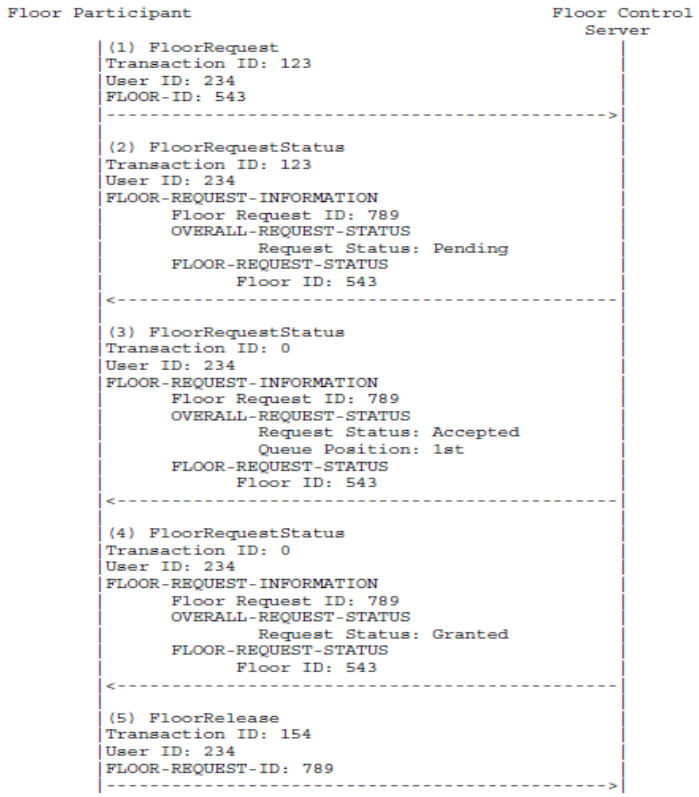

Figure 2.5: User requests a floor, obtains it, and, at a later time, releases it [3]

2.1.4.2 Talk Burst Control Protocol (TBCP)

TBCP (Talk Burst Control Protocol) [5] has been defined as the floor control mechanism for PoC (push to talk over cellular) services by the Open Mobile Alliance (OMA).TBCP uses the application extension features of RTCP (RTP Control Protocol) for the exchange of information.

The basic TBCP messages [4] are as follows:

- Talk Burst Request: This request is issued by the participant willing to access the floor. The request contains the priority level which the user can use to notify the server about the importance of the request.
- Talk Burst Granted/Denied: This is used by the server to notify the requesting client that its request has been either accepted or rejected. In

case the Talk Burst was granted, the server may notify a 'stop talking' timer in order to limit the length of the Talk Burst.

- Talk Burst Release: This is sent by a client to notify the server that it has finish sending the talk burst.
- Talk Burst Idle/Taken: This is used by the server to notify the participants whether the floor is free or not.
- Talk Burst Revoke: This is used by the server to pre-empt an ongoing Talk-Burst.

TBCP is a fast and secure protocol. The only limitation is that it provides basic floor control functionalities (e.g. no chair supported). Figure 2.6 illustrates the dome of TBCP operations. Here the PoC server acts as a floor control server and is responsible for granting the floor request to the clients.

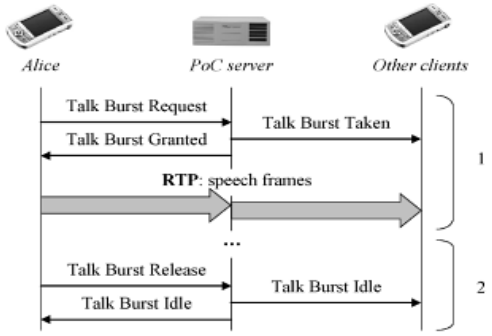

Figure 2.6: TBCP operations [4]

2.1.5 Protocols between the Conferencing Application Server and the Floor Control Server

The floor control server (FCS) capabilities are managed by the conferencing application server using the following protocols:

- Megaco/H.248: Used by the standard body 3GPP [7] between the application server and MRFP (i.e. FCS) to provide the floor-related capabilities such as creating/removing floors, associating the resources to the floors and appointing floor chairs.
- SIP Floor Sever Control Markup Language (SIP-FSCML) [8]: Proposed outside the standard bodies as a communication protocol between the application server and the FCS. It is less complex and easy to understand and use by SIP application developers. It follows SIP and XML paradigms. It enables a peer-to-peer communication model between the application server and FCS. This allows the FCS to be simultaneously used by multiple application servers. The characteristics of the protocol are as follows:
    - FSCML requests to the FCS are carried in SIP INFO messages where each INFO message includes a single FSCML body
    - An FSCML body can carry any number of FSCML requests.
    - SIP-FSCML responses are transported in a separate INFO message.
    - SIP-FSCML Is a request-response protocol with only final responses.

SIP-FSCML based operations:

- Open/close control connection
- Create floor
- Create floor connection

- Add/remove floor to/from a conference
- Set/update chair for a floor
- Add/remove floor participant(s)
- Set floor algorithm
- Add/remove media to/from a floor
- Set maximum floor holders
- Set maximum floor holding time

2.2 Web Services

This section discusses the Web services. We start with the definition of Web services followed by the Web services model. Then we discuss the SOAP Web services. The last section discusses Parlay-X Web services as one of the applications of SOAP Web services in the telecommunication domain.

2.2.1 Definition

Web services in their simplest definition are programmatic interfaces that allow application-to-application communication over a network [9]. Web services have become an attractive approach of application/service integration over internet mainly because of the following fundamental principles [10]:

- Coarse grained approach: The Web service technology provides a higher level of abstraction that allows developers to integrate required functionalities to their applications easily and rapidly.
- Loose coupling: Applications developed using Web services are loosely coupled, which makes them independent. For example, application A

which talks to application B should not necessarily be re-written if application B is modified.

- Synchronous and asynchronous mode of communication: Web service applications support both synchronous and asynchronous modes of communication.

2.2.2 Web Service Model

Web Services architecture is based on the interactions between three entities [9]:

- Service provider: From a business perspective, this is the owner of the service. From an architectural perspective, this is the platform that hosts access to the service.
- Service registry: This is a searchable registry of service descriptions where service providers publish their service descriptions. Service requestors find services and obtain binding information (in the service descriptions) for services during development for static binding or during execution for dynamic binding. For statically bound service requestors, the service registry is an optional role in the architecture because a service provider can send the description directly to service requestors. Likewise, service requestors can obtain a service description from other sources besides a service registry, such as a local file, FTP site, Web site, Advertisement and Discovery of Services (ADS) or Discovery of Web Services (DISCO).
- Service requestor: From a business perspective, this is the business that requires certain functions to be satisfied. From an architectural perspective, this is the application that looks for and invokes or initiates an

interaction with a service. The service requestor role can be played by a browser driven by a person or a program without a user interface (e.g. another Web service).

The interactions involved between the three entities are publish, find and bind operations. In a typical scenario, a service provider hosts a network-accessible software module (i.e. an implementation of a Web service). The service provider defines a service description for the Web service and publishes it to a service requestor or service registry. The service requestor uses a find operation to retrieve the service description locally or from the service registry, and uses the service description to bind with the service provider and invoke or interact with the Web service implementation. Service provider and service requestor roles are logical constructs and a service can exhibit characteristics of both.

Figure 2.7 illustrates the Web services business model. It includes the interaction between Web service entities using the defined operations (i.e. publish, find and bind).

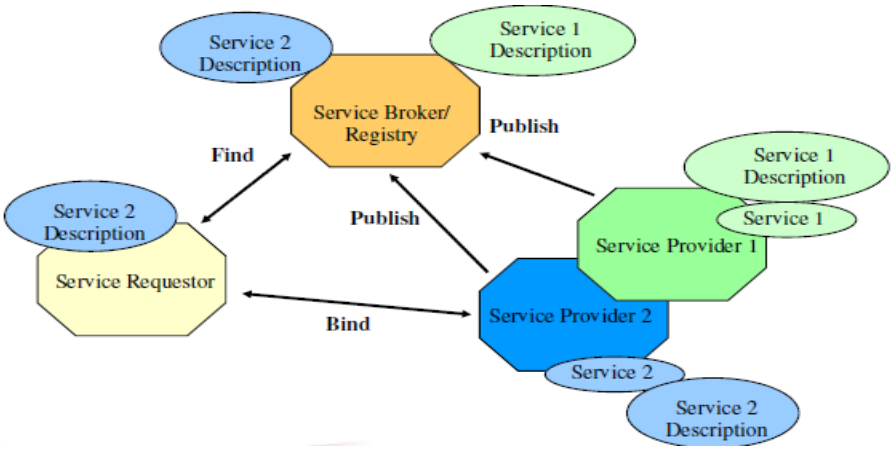

Figure 2.7: Web services business model

Types of Web Services

There are two types of Web services: SOAP Web Services and RESTful Web Services. Since our research interest is in SOAP Web services, it is discussed in detail in the following sub-section.

2.2.3 SOAP Web Services

This sub-section discusses the SOAP Web services, starting with its definition. Next, the technologies involved in the SOAP Web services are discussed. Then the applications of SOAP Web services in telecommunication are provided. Finally Parlay X Web services are discussed as one of the application of SOAP Web services.

2.2.3.1 Definition

SOAP Web services [11] are also called Big Web Services or WS-* Web services in the literature. SOAP Web services interactions between the entities are based on XML and use SOAP which is embedded in HTTP.

2.2.3.2 Technologies Involved

A SOAP Web services stack exisits that enables the feasibility of the three operations of publish, find and bind in an interoperable manner [9] [11]. The conceptual Web services stack is illustrated in figure 2.8.

The upper layers build upon the capabilities provided by the lower layers. The vertical towers represent requirements that must be addressed at every level of the stack. The text on the left represents standard technologies that apply to that layer of the stack.

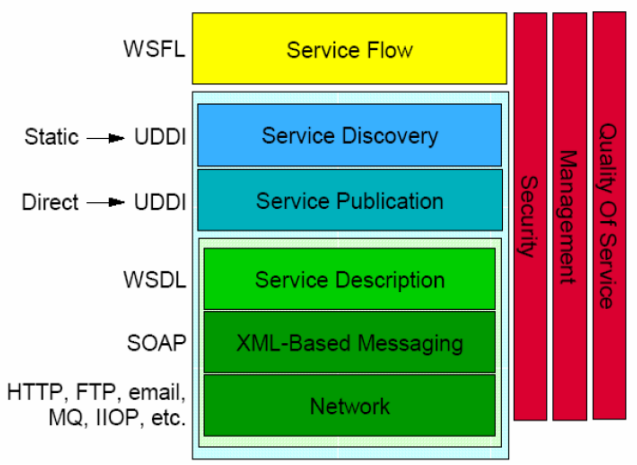

Figure 2.8: SOAP Web services conceptual stack and technologies involved [9]

The foundation of the SOAP Web services stack is the network. Web services must be network-accessible to be invoked by a service requestor. HTTP is the de facto standard network protocol for Internet-available Web services. Other Internet protocols can be supported, including SMTP and FTP.

The next layer, XML-based messaging [12], represents the use of XML as the basis for the messaging protocol. SOAP is the chosen XML messaging protocol for many reasons [12]:

- It is the standardized enveloping mechanism for communicating document-centric messages and remote procedure calls using XML.
- It is simple; it is basically an HTTP POST with an XML envelope as a payload.
- It is preferred over simple HTTP POST of XML because it defines a standard mechanism to incorporate orthogonal extensions to the message using SOAP headers, and uses standard encoding of operation or function.

- SOAP messages support the publish, find and bind operations in the Web services architecture

The next layer, service description, is actually a stack of description documents. First, WSDL [12] is the de facto standard for XML-based service description. This is the minimum standard service description necessary to support interoperable Web services. WSDL defines the interface and mechanics of service interaction. An additional description is necessary to specify the business context, qualities of service and service-to-service relationships. The WSDL document can be complemented by other service description documents to describe these higher-level aspects of the Web service. For example, business context is described using UDDI data structures in addition to the WSDL document.

Because Web services are defined as being network-accessible via SOAP and represented by a service description, the first three layers of this stack are required to provide or use any Web service. Considering this, the simplest possible stack would consist of HTTP for the network layer, the SOAP protocol for the XML messaging layer and WSDL for the service description layer (Figure 2.9).

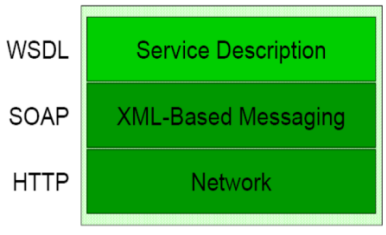

Figure 2.9: Simplest SOAP Web service stack [11]

This is the interoperable base stack that all inter-enterprise or public Web services should support. It provides interoperability and enables Web services to leverage the existing Internet infrastructure.

While the bottom three layers of the stack identify technologies for compliance and interoperability, the next two layers, service publication and service discovery, can be implemented with a range of solutions. WSDL can be made available in several ways, including:

- The service provider sends a WSDL document directly to a service requestor (i.e. direct publication).
- The service provider can publish the WSDL document describing the service to a host local WSDL registry, private UDDI registry or the UDDI operator node.

Similarly, there are varieties of discovery mechanisms to gain access to the service description and make it available to the application at runtime:

- The service requestor retrieves a WSDL document from a local file (usually the WSDL document obtained through a direct publish).
- The service can be discovered at design time or runtime using a local WSDL registry, private UDDI registry or the UDDI operator node.

2.2.3.3 Application of SOAP Web Services in Telecommunications

SOAP Web service is a key technology for service provisioning in next-generation networks (NGNs). SOAP Web services can be used to expose the network capabilities (e.g. multimedia conferencing, call control, presence,

messaging) as Web services to the application developers. Web services introduce loose coupling between applications and keep the communication at a higher level of abstraction which makes application development easier and faster.

Parlay-X [14] and Open Mobile Alliance (OMA) specifications [16] are based on SOAP Web services. Parlay-X Web services are the building blocks of telecommunication capabilities that application developers can quickly comprehend and use to generate new and innovative applications. Parlay-X specifications aim to cover all telecommunication capabilities [13]. However, OMA specifications focus more on mobile services. They aim at providing solutions to problems incurred when using Web services in OMA environments.

2.2.3.4 Parlay-X Web Services

This sub-section starts with the introduction of Parlay X Web services. Then it discusses the architecture for Parlay X Web Services.

2.2.3.4.1 Introduction

The Open Service Access (OSA) [15] defines an architecture that enables service application developers to make use of network functionality through open standardized interfaces (e.g. the OSA APIs (Parlay APIs) and Parlay X Web services). The Parlay APIs are designed to enable the creation of telephony applications as well as to 'telecom-enable' IT applications, but they are quite low-level APIs, requiring developers to have some understanding of telecommunications concepts. IT developers, who develop and deploy applications outside the traditional telecommunications network space and business model, are viewed as crucial for creating dramatic market growth in

next-generation applications, services and networks. On the other hand, Parlay X Web services intend to offer a higher-level abstraction of the network infrastructure by providing a set of interfaces where functions are grouped according to the type of services they enable instead of toward the original network capabilities to which each function is related.

The Parlay X Web Services [14] are intended to stimulate the development of next-generation network applications by IT developers who are not necessarily experts in telephony or telecommunications. The Parlay X specification describes a number of Web services that will provide a simple interface for telephony and other systems. They aim to cover all telecommunication capabilities [13] (e.g. third-party call, multimedia conferencing, calls notification, short messaging etc.)

2.2.3.4.2 Overall Parlay X Web Services Architecture

The Parlay-X Web service deployment model [15] [14] is shown below. The model illustrates publication of Parlay X Web services through a registry to make those Web services available for discovery, and application use of the Web service access methods to interact with the gateway, where the Web service interfaces are implemented.

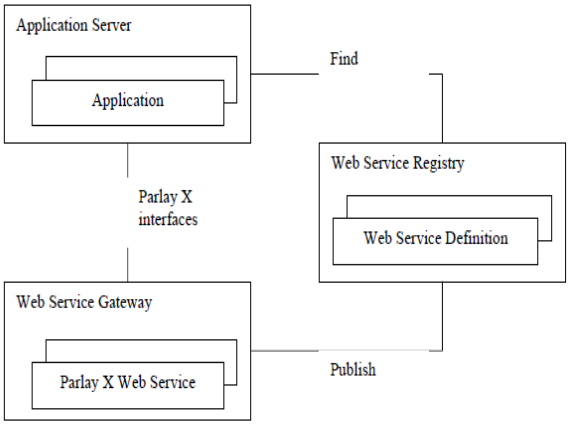

Figure 2.10: Parlay-X Web service model [15]

Combining this model with the existing OSA/Parlay deployment configurations gives the overall architecture for the Parlay-X Web services which is illustrated in the next figure. Parlay X is a subset of the Parlay technology that gives application developers access to the Parlay gateways using Web services. Parlay X Web services can be used independently of a Parlay gateway and can also be used to talk directly to a network (assuming the network implements the Parlay X specification), which is illustrated in Figure 2.10.

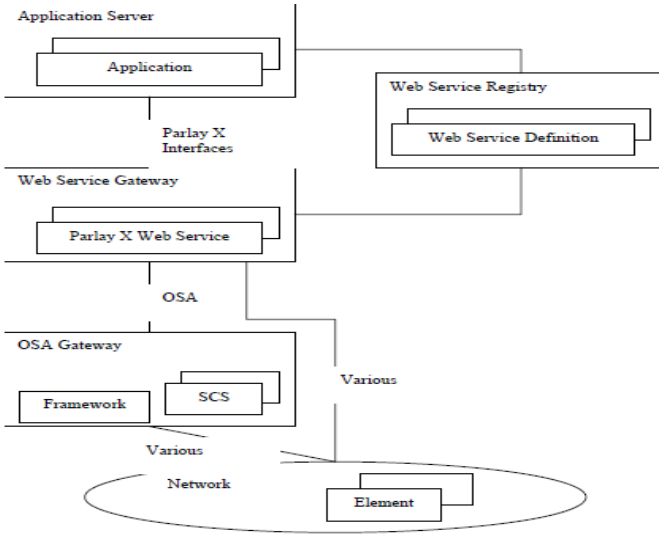

Figure 2.11: Overall Parlay-X Web service architecture [15]

2.3 Chapter Summary

This chapter has successfully introduced the background information related to the thesis. We first introduced the floor control in multimedia conferencing followed by the model for floor control. Next we discussed the floor control integration with conferencing. Then we discussed the existing protocols for floor control, including protocols between the conferencing application server and the floor control server (FCS) to manage the FCS capabilities. The next section introduced Web services where we provided the basic definition and business model of Web services. Then we discussed SOAP Web services in details (i.e. definition and technologies involved). Next we presented the Parlay X Web service as one of the applications of SOAP Web services .We also introduced Parlay Web service architecture.

In the next chapter we will propose a set of requirements for floor control architecture in multimedia conferencing. Afterward, we will discuss some existing works most related to our research and evaluate them based on our requirements.

Chapter 3

Requirements and State of the Art Evaluation

Various standard bodies (e.g., 3GPP, IETF, Parlay) and author have proposed architectures that integrate floor control with conferencing. However, the mechanisms used for exposing the floor control capabilities have shortcomings that can hinder application development. In order to develop an integrated structure that overcomes the existing shortcomings, a set of requirements should be derived and use to analyze the shortcomings in a systematic manner.

This chapter is composed of three sections. We first propose a set of requirements for floor control in multimedia conferencing. Afterwards, we review the state of art works related to the thesis research and evaluate them based on our requirements. Finally, we summarize the chapter in the end.

3.1 Requirements for Floor Control in Multimedia Conferencing

This section contributes two sets of requirements for floor control in multimedia conferencing: functional requirements and architectural requirements. Functional

requirements outline the floor control functionality a system should provide, and the architectural requirements specify criteria that can be used to judge the operation of a system.

3.1.1 Functional Requirements

We define the main functional requirements for floor control as follows:

- A participant should be able to request the floor.
- The floor should be granted based on the floor policy. When the floor control policy is chair-moderated, the decision to grant the floor is issued by the designated chairperson of the floor. However, if the floor policy is FCFS (first come first serve) or based on any other algorithm, then the decision is made by the floor control server (i.e. an entity responsible for managing the floors and their status).
- A participant should be able to release the floor and make it available to others.
- The floor chair or moderator should be able to revoke the floor from the participant holding the floor.
- The participants should be notified about any changes in the floor status.

We believe that these requirements provide a complete set of functional requirements for floor control and can fulfill any conferencing scenario that needs a floor control mechanism.

3.1.2 Architectural Requirements

This sub-section aims at providing the architectural requirements to integrate floor control mechanisms in multimedia conferencing. The requirements are given below:

- The architecture should expose to application developers the floor capabilities (e.g. create/remove floor, add/remove participant to/from floor, set chair, request floor, release floor, revoke floor, floor query) along with basic conferencing capabilities (e.g. create multimedia conferences, add/remove participants from the conference, delete conference, add/remove media) via well-defined APIs.. The APIs should be programming language-independent and should also provide a higher level of abstraction to make the development of applications relatively easier.

- The architecture should enable any application residing in any application server to use the conferencing capabilities (including the floor capabilities) via these APIs.

- The entity responsible for providing the floor capabilities (i.e. FCS) should not be located inside the server implementing the conferencing capabilities. Otherwise, the framework is less scalable because the FCS cannot be used by other conferencing servers.

- Furthermore, the entity responsible for providing the media (i.e. media server) should not be collocated with the entity providing the floor capabilities (i.e.

FCS). Otherwise, the modularity of the framework is descreased because if one entity requires replacement, the other would have to be upgraded.

- The architecture should support use of different floor control protocols transparently to the floor clients.

- The architecture should support client portability, such that floor clients access the floor capabilities independently of the protocol supported by the framework.

3.2 Evaluation of Related Work

In this section we introduce several related works on floor control in multimedia conferencing and then evaluate them based on our requirements.

3.2.1 Related Work

In this subsection, we organize the related works into two approaches: work done by standard bodies and work done outside the standard bodies. Finally, we present the summary of evaluation.

3.2.1.1 Related Work Proposed by Standard Bodies

The floor control architecture proposed by the 3GPP [7] is depicted in Figure 3.1.

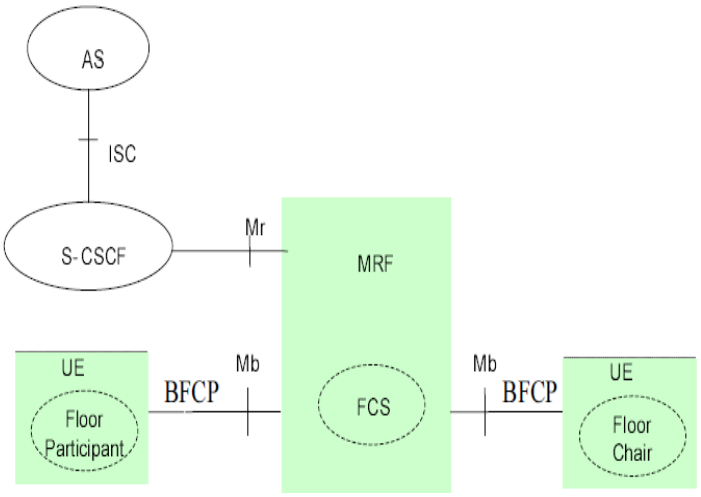

Figure 3.1: Functionality architecture of floor control [1]

According to the 3GPP specifications, the conference participants and Media Resource Function Processor (MRFP) can optionally support the floor control capabilities. Floor control offers control of shared MRFP conference resources In the proposed architecture, FCS is collocated with the MRFP.

BFCP protocol is used to convey the floor control messages between the floor chair of the conference, the FCS, and the floor participant. The other floor control requirements, such as associating s floor to the resources and BFCP connection negotiation between UE and FCS, are established using H.248/Megaco protocol as illustrated in Figure 3.2. MRFC is responsible for negotiating the required parameters for the BFCP connection between UE and MRFP (i.e. FCS).

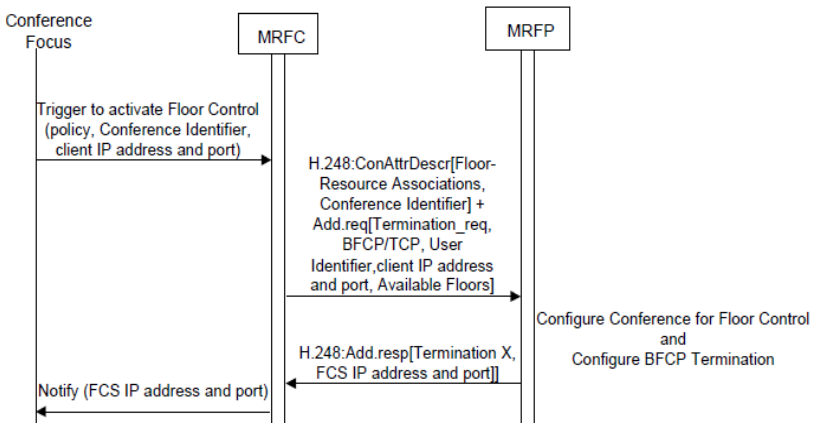

Figure 3.2: **Combined procedures to configure a conference and add a floor control termination [1]**

The architecture proposed has some drawbacks. Firstly, there is no API proposed for the application development. Secondly, MRFP has to host new functionalities to provide floor control capabilities. Because there is no interface between the FCS and MRFP, both have to be bought from the same supplier, which reduces the modularity of the framework. Thirdly, the floor client directly interacts with the FCS, so if the FCS is replaced, the client has to be upgraded. Client portability is a problem. Furthermore, this constrains the possibility of using different floor control protocols transparently to the floor clients.

The conferencing architecture proposed by IETF [18] exposes the floor control capabilities to the floor clients. The framework is built around the fundamental concept of a conference object. The conference object is a data representation of a conference during each of the various stages of a conference (e.g., creation, reservation, active, completed, etc.). It is accessed via logical functional elements with which a conferencing client interfaces, using the various protocols as illustrated in the Figure 3.3.

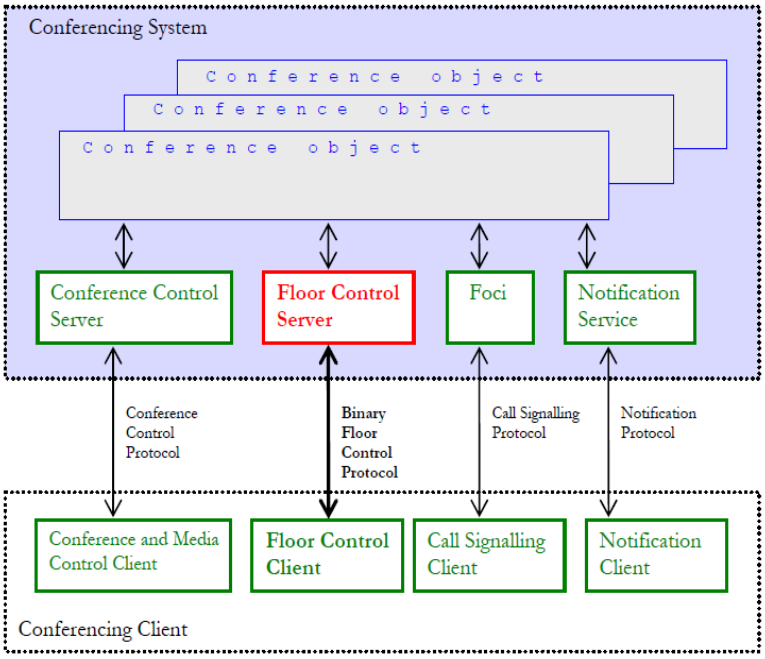

Figure 3.3: Conferencing system logical decomposition [2]

A floor client accesses the floor capabilities from the FCS using BFCP as the protocol. The parameters for the BFCP connection termination are negotiated using the SDP [RFC4566] offer/answer [RFC3264] exchange on the signaling interface with the focus. Once a connection has been established, a specific floor control message requires detailed information to uniquely identify a conference, a user and a floor. However, in the proposed architecture FCS is located inside the conference server, so it cannot be used by other conferencing application servers, which reduces the scalability of the framework. Client portability is a problem because the client interacts directly with the FCS. Furthermore, the architecture does not include any APIs for application development.

In [19], Parlay proposed the architecture that provides APIs to expose the basic floor control capabilities in a multi-party conference (e.g. chair selection, appoint speaker, floor request, inspect video, inspect video cancel). The scenario proposed includes a prearranged add-on multimedia conference where the end user, who initiates the call, communicates with the application via the Web interface. The end user can do things that normally the chair would be able to do (e.g. determine who has the floor, whose video is being broadcast to the other participants) or inspect the video of participants who do not have the floor (e.g. to see how they react to the current speaker) via the Web interface. The scenario is illustrated in Figure 3.4, where the end user executes the application to configure the conference with the selected participant via the Web interface.

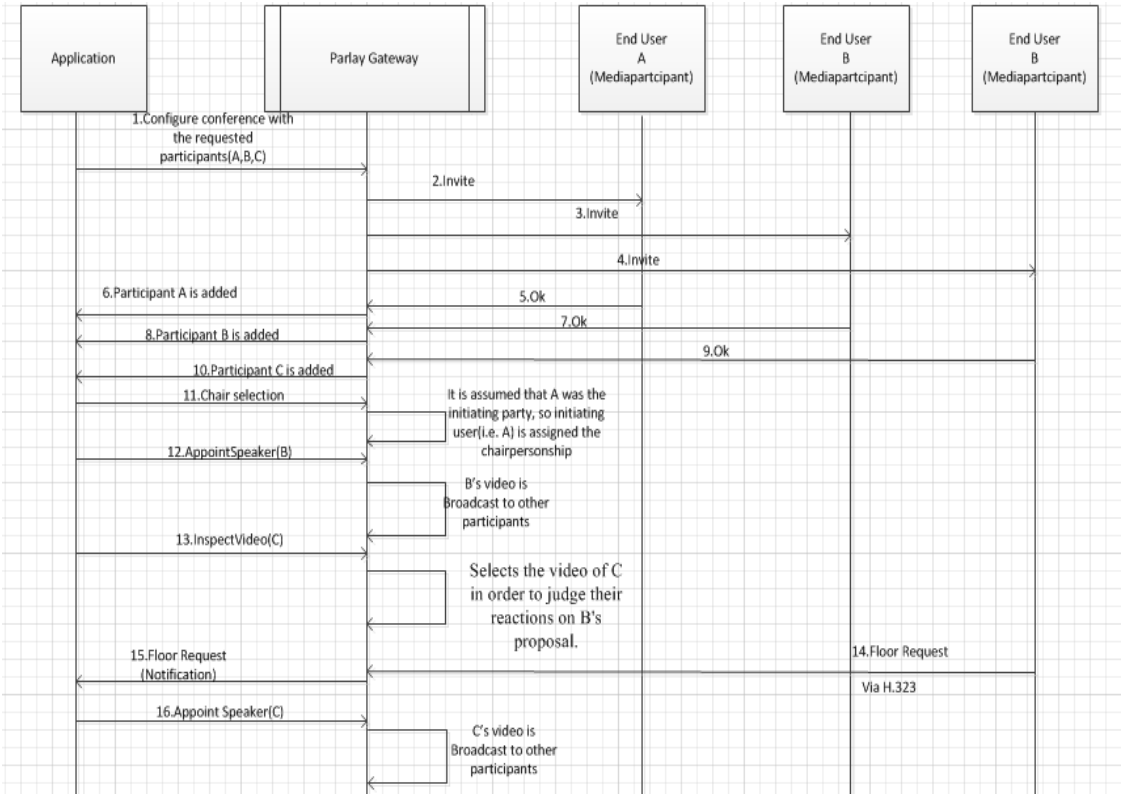

Figure 3.4: Scenario for floor control in conferencing

The application then renders the service from the gateway and is notified for each acceptance (Step 1 to 10) [Figure 3.4].Chairperson (A) decides via the Web interface that party B is the speaker. This means that the video of B is broadcasted to the rest of the participants (Step 12) [Figure 3.4]. A floor client requests the floor using H.323 protocol.

It can be concluded that floor APIs provided by the Parlay/OSA are very limited, because floor capabilities such as release floor, revoke floor and floor query are not included. Also, video is the only resource that can be shared between the participants in the assigned floor. The proposed APIs are at a low level, so development of application is relatively difficult. The client directly requests the floor from the FCS. The FCS is assumed to be located in the Parlay gateway. Therefore, client portability is a problem.

In [20], an architectural framework for media server control is described by IETF. This document presents the core architectural framework to allow application servers to control media servers. Figure 3.9 illustrates the basic signaling architecture between the entities involved in the media server control framework. SIP, being the primary signaling protocol for session signaling, is used for all media sessions directed toward a media server. SIP is also used for the creation, management and termination of the dedicated media server control channel. Application and media servers use the SDP attributes defined in [RFC4145] to allow SIP negotiation of the control channel. Application servers use the SIP Third Party Call Control [RFC3725] (3PCC) to establish, maintain and tear down media streams from those SIP user agents to a media server.

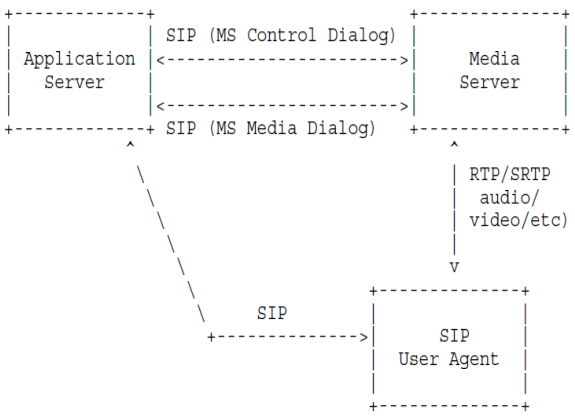

Figure 3.5: Basic signaling architecture [7]

The authors describe the media control for conferencing services such as creating a new conference, adding participant to the conference, media controls and floor control. The FCS is considered as a separate logical entity that can interact with the application server and media server as needed. According to the authors, the FCS can be collocated with either the application server or media server, as long as both elements are allowed to interact with the FCS by means of some kind of protocol. They presented both the approaches to better explain the interactions between the involved components in the Figures 3.10 and 3.11[7].

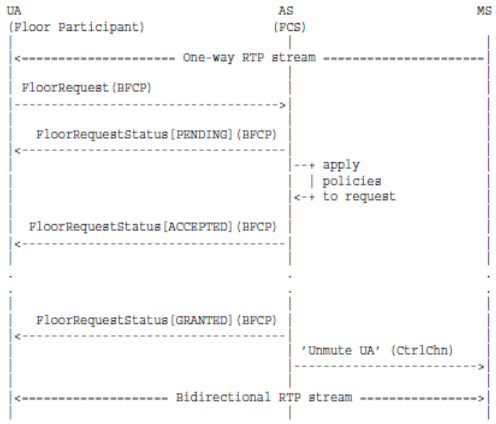

Figure 3.6: FCS collocated with application server

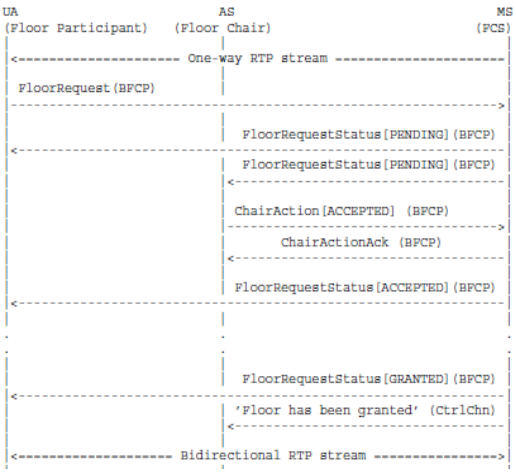

Figure 3.7: FCS collocated with media server

The framework does not consider the approach where the FCS can function as an independent entity (i.e. neither collocated with the application of media servers). There is no API proposed for application development. Furthermore, client portability is a problem since the floor client directly interacts with the FCS.

3.2.1.2 Related Work Proposed Outside the Standard Bodies

The architecture defined in [21] is outside the standard bodies. They proposed the floor control architecture for multimedia conferencing which includes a floor control server, conference server, one or more SIP servers (SIP for Session Initiation Server), conference owner, one or more floor chairs (moderators) and any number of regular conference participants as illustrated in Figure 3.5.

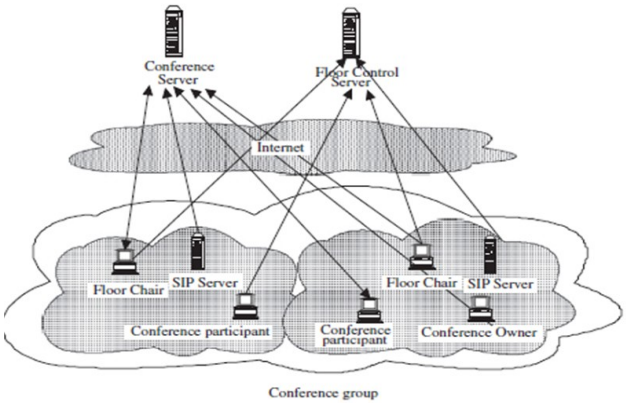

Figure 3.8: Floor control architecture [4]

Processing (granting or rejecting) of floor control requests are done by one or more floor chairs or by the server itself, depending on the policy. The conference server is used to dynamically maintain the conference information. It receives service requests from the owner, the chairs and the participants. The conference owner creates the conference and the floors, and assigns/changes floor chairs. Conference participants can request floors from the FCS, and when the floor is granted, that they can start sending media. The Simple Conference Control Protocol (SCCP) entity is used to provide the conference management services and floor control services.

The proposed architecture does not provide any API for application development. There is no interface defined between the FCS and conference server which means that the conference application cannot access the floor capabilities. Clients are burdened to implement most of floor control capabilities, which reduces the modularity and makes the client portability a problem.

Another work [8] outside the standard bodies proposed a novel floor control architecture which extends existing IMS multimedia conferencing architecture to introduce floor control capabilities. In the proposed architecture, the FCS is the key add-on to the existing IMS conferencing architecture as depicted in Figure 3.6. They provided both the client-side and server-side APIs that expose floor control capabilities to application developers.

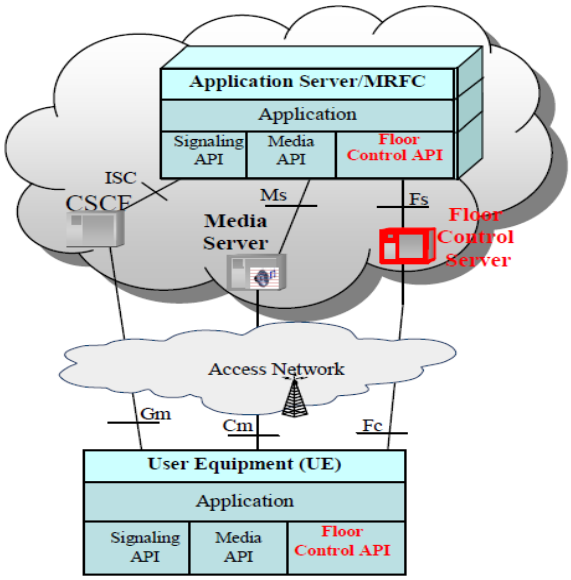

Figure 3.9: Overall floor control architecture [5]

They focused on the dial-out conferences and further assume that conference application, conference participants, floor participants and floor chairs are all in the same IMS domain. They provided an implementation architecture including the prototype built on it. BFCP protocol is used to provide the client floor capabilities (e.g. request floor, release floor, revoke floor, floor status). They proposed a SIP Floor Server Control Markup Language (SIP-FSCML) for controlling the FCS (e.g. add floors to a conference, set/modify a floor chair,

add/remove participants to/from an existing floor, subscribe to floor event notifications, and remove an existing floor from a conference). However, the APIs proposed are language-dependent. Furthermore, client portability is a problem because the client has to be upgraded if the FCS is replaced or supports different protocols.

Reference [2] outlines the requirements for conference control components: conference management and floor (resource) control. Furthermore, they proposed a conference control framework using SIP and SOAP protocols. It is shown that conference control can be implemented with two kinds of operations: commands and notifications. SOAP is used for commands since it fits well for exchanging RPC calls. The SIP event framework is used to deliver notifications. An example illustrating the framework is provided in Figure 3.8.

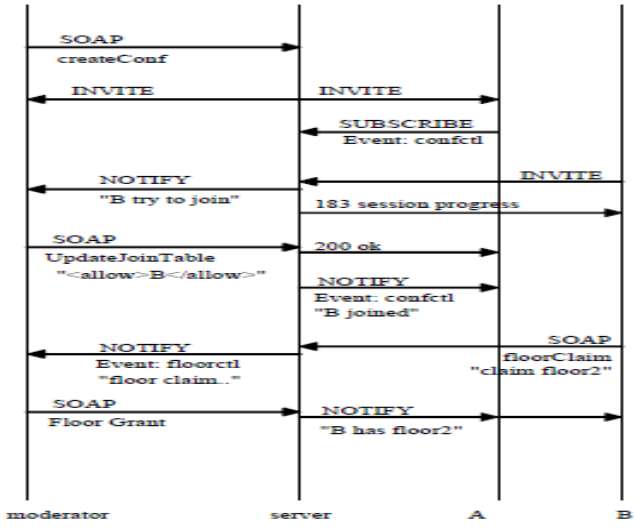

Figure 3.10: An example of conference control signaling [6]

However, the framework does not include any APIs for development. In the proposed architecture, the FCS is assumed to be collocated with conference server, which reduces the scalability of the framework because the FCS cannot be used by other application servers. Client portability is also a problem because the client interacts directly with the FCS.

3.2.1.3 Evaluation Summary

After presenting the most relevant works related to our research interest, we can observe that none of them fully satisfy our requirements.

In [7], [18], [19], [2] and [20], no floor control APIs are provided for application development. Reference [19] provides the APIs but they are not comprehensive and they also require low-level details for development. Reference [8] provides comprehensive floor capabilities but the APIs are programming language-dependent.

Standard bodies Outside Standard bodies

Related Works / Requirements	3GPP TS 23.333 version 10.3.0 Release 10, January 2012	"A Framework for Centralized Conferencing", IETF RFC 5239, June 2008.	Parlay 5.1 specifications: Conference Call Control SCF, ETSI ES 203915-4-5 V1.2.1	"Architectural Framework for Media Server Control", IETF RFC 5567, June 2009	"Description of teleconferencing floor control protocol and its implementation", ELSEVIER, 2008.	A Novel Architecture for Floor Control in the IP Multimedia Subsystem of 3G Networks, IEEE 69th, April 2009.	"A SIP-based Conference Control Framework", NOSSDAV '02.
1. Comprehensive floor control functionality	Supported	Supported	Not supported	Supported	Supported	Supported	Not supported
2. Support APIs, which are language independent and provide higher level of abstraction	No API is provided	No API is provided	Partially supported	No API is provided	No API is provided	Partially supported	No API is provided
3. FCS should not be collocated with the application server	Supported	Not supported	Not supported	Not supported	No interface between FCS and application server	Supported	Not supported
4. FCS should not be collocated with the media server.	Not supported	Not discussed	Not discussed	Not supported	Not discussed	Supported	Not discussed
5. Should support different floor control protocols transparently to client.	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
6. Client portability	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

Table 3.1: Evaluation of relevant state of art

None of the existing works support client portability. In [18], [19], [2] and [20], the FCS is collocated with the conferencing server, which makes the framework less scalable. The architectures proposed in [7] and [20] have the FCS collocated with the media server, which reduces the modularity of the framework.

3.3 Chapter Summary

In this chapter, we first derived a set of requirements which included both the functional and the architectural requirements for floor control in multimedia conferencing. In the next section, we presented most relevant state-of-the-art work related to our research and evaluated it based on our requirements. Finally, we concluded that none of them completely satisfies our requirements.

In the next chapter, we will present our proposed architecture based on the requirements presented in this chapter.

Chapter 4

Proposed Architecture

In this chapter, we propose a SOAP Web services-based architecture for floor control in multimedia conferencing, which is based on the requirements discussed in the previous chapter. This chapter is organized into three sections. Firstly, we present the overall architecture of the floor control in multimedia conferencing which includes the functional entities and the communication interfaces. We also summarize how the requirements are met by the architecture. The next section presents the proposed SOAP Web services-based floor control APIs. Finally, we present a few illustrative scenarios that show how entities in the system architecture interact with each other.

4.1 Overall Architecture

In this section, we will first present our proposed overall architecture that shows the functional entities in the system, followed by a sub-section that describes the communication interfaces between the system's entities. Finally we summarize how the requirements are met by the architecture.

4.1.1 Functional Entities

Figure 4.1 depicts the overall architecture for floor control in multimedia conferencing. It includes a conferencing application, conferencing gateway, floor control server (FCS), media server (MS) and client user equipments (UEs) as the main functional components.

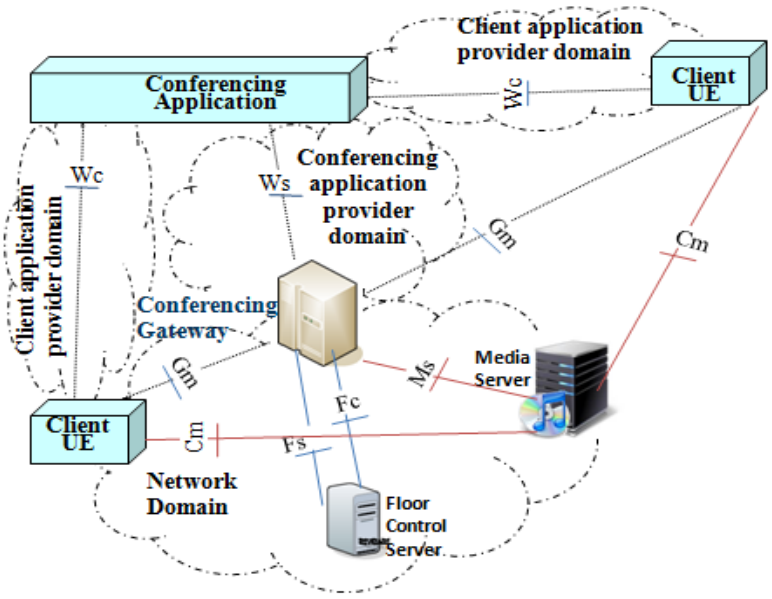

Figure 4.1: Overall architecture

In the overall architecture [Figure 4.2], the conferencing gateway offers and implements conferencing and floor capabilities for the conferencing application via well defined application programming interfaces (APIs). The offered functionality includes both, conferencing capabilities, which are provided by the standard Parlay X multimedia conferencing Web service [22] and floor control

capabilities, which include both the server–side (e.g., adding and removing floors to/from the conference, adding and removing participants to/from the floor, revoking the floor from the participants and setting the chair for the floor) and client-side (e.g., requesting a floor, releasing a floor, granting a floor, denying a floor and getting the floor information) floor control capabilities.

The client application accesses client-side floor control capabilities via well defined APIs from the conferencing application. The chair and the participant of the floor can access floor control capabilities depending on their role. For instance, a chair can grant/deny or revoke a floor from a participant, while other floor participants are not privileged to these capabilities. The conferencing application receives the function calls from the client application and maps them onto the eventual functional calls that are sent to the conferencing gateway, where they are actually implemented.

The conferencing gateway is responsible for the network implementation of the services provided by the conferencing application. In the network domain, we have FCS and MS as the main components. The FCS is the entity that maintains the floor(s) status (e.g., which floors exists, who the floor chairs are, who holds the floors). It is controlled using conferencing gateway, to expose the floor control capabilities to the both conferencing and client applications. The media communications between the conference participants are managed by a media server (MS), which is controlled via the conferencing gateway.

The conferencing gateway manages call session management (e.g. set up, modification, and teardown) of the conference participants in the network. It

interacts with the participants located in different domains by inter-domain signaling.

4.1.2 Communication Interfaces

This sub-section discusses the existing interfaces between the system components of the architecture.

SOAP Interfaces (Wc and Ws)

Wc is a SOAP Web services-based interface between the client application and the conferencing application. It offers client-side floor control functionalities (e.g., requesting a floor, releasing a floor, granting the floor, denying the floor and getting the floor information).Similarly, Ws is a SOAP Web services-based interface; it is used for the communication between conferencing application and conferencing gateway. It offers functionality that includes both the conferencing capabilities (provided by the standard Parlay X multimedia conferencing Web service) and the floor control capabilities, which are provided by the proposed SOAP Web services-based floor control APIs discussed in the next section. We choose SOAP [11] interface because it is a standardized enveloping mechanism for communicating document-centric messages and remote procedure calls using XML. SOAP messages supports the PUBLISH, FIND and BIND operations in the Web service architecture. Furthermore, it provides language, platform and transport neutrality.

Floor Control Interfaces (Fc and Fs)

The conferencing gateway interacts with the FCS to expose the floor capabilities using two different interfaces: Fc and Fs.

The Fs interface is used to control the FCS .It can be implemented using SIP Floor server control mark-up language (SIP-FSCML) [8] or H.248/Megaco [7] protocols. However, we propose SIP-FSCML over H.248/Megaco because it is less complex than H.248/Megaco and is easy to understand and use by SIP application developers.

The Fc interface coordinates access to shared resources by providing all the client-side floor control functionalities. Binary Floor Control Protocol (BFCP) [3] and Talk Burst Control Protocol (TBCP) [4] are two standard protocols that can be used for the Fc interface. BFCP is fast (due to binary encoding), secure, reliable (uses TCP) and provides all the floor control functionalities. TBCP is also fast and secure, but it only provides basic floor control functionalities (e.g. no chair supported). Therefore, we propose to use BFCP for the Fc interface.

Media Interfaces (Ms and Cm)

The Ms Interface is used by the conferencing gateway to control the media server. It can be implemented using standard protocols such as H.248/Megaco [23] and SIP Media Server Control Markup Language (MSCML) [24]. H.248 is the standard for media server control protocol. However, it is complex and there are few commercial deployments. SIP MSCML on the other hand is an emerging alternative. It provides SIP-based enhanced conferencing and interactive voice response (IVR) functions. It is less complex and there are more commercial deployments.

Also, XML-RPC [25] calls can be used to control the media server.

Cm is a media-handling interface based on Real time Transport Protocol (RTP), between the client UE and media server.

Signaling interface (Cm)

Cm is the signaling interface used between the conferencing gateway and client UE. It provides the capabilities such as session establishment, modification and termination. Cm can be implemented using standard signaling protocols like SIP [26] and H.323 [27]. However, we propose SIP over H.323 because it is the most widely deployed signaling protocol for multimedia conferencing. SIP supports inter-domain signaling, so the conferencing gateway can interact with end users in different domains.

4.1.3 Requirements met by the architecture

The refined architecture satisfies all the requirements derived in the previous chapter. Firstly, the SOAP Web services–based APIs are used to expose both the conferencing and floor capabilities, which makes the framework more interoperable. Secondly, the client does not interact directly with the FCS to access floor capabilities, which allows the framework to use any floor control protocol transparently to the clients. Additionally, it provides client portability. Furthermore, the conferencing gateway responsible for implementation of the conferencing and floor capabilities is not collocated with the FCS. This makes the framework more scalable since the FCS can be simultaneously used by other conferencing gateways. Lastly, both the FCS and the MS are separated in the architecture, which adds modularity to the framework.

4.2 Proposed SOAP Web Services-based Floor control APIs

The proposed SOAP Web services-based floor control APIs includes comprehensive set of the server-side and the client-side APIs that exposes the floor control capabilities to application developers. In the following subsections we will present the proposed floor control APIs for the conferencing application (server-side) and the client application (client-side).

4.2.1 Proposed server side Floor Control APIs

SOAP Web services-based floor control APIs are proposed to extend the existing Parlay-X (SOAP-based) multimedia conferencing Web service functionality with floor control capabilities. The conferencing application is able to access conferencing and floor capabilities (e.g., creating conferences with and without floor control, adding and removing participants from conference and floor, and setting the floor chair) via these proposed APIs. The following sub-sections outlines the proposed server-side APIs.

4.2.1.1 Adding Floors

There are two ways to add a floor to a multimedia conference:

- The floor is added when the conference is initialized.
- The floor is added after the conference has been initialized.

This is achieved by the following proposed APIs.

Create_Conference_with_FloorControl()

The standard Create_Conference () API of the Parlay X multimedia conferencing Web service is extended to provide floor capability. It creates an empty Dial-Out

multimedia conference with floor control. The reference to the newly-created multimedia conference and floor is returned in the output parameter. Tables 4.1 and 4.2 detail the selected parameters for the request and response of the method.

Parameter Name	Parameter Type	Optional	Description
conferenceType	String	No	Type of conference (e.g. audio, video, messaging).
conferenceDescription	String	Yes	Description of the conference.
charging	String	Yes	If present, defines the charge per unit of time consumed on the conference call.
maximumDuration	Integer	Yes	If present, defines the maximum duration of the multimedia conference in seconds.
maximumNumberOfParticipants	Integer	No	Defines the maximum number of participants allowed in a conference.
conferenceOwner	String	Yes	Address of the multimedia conference owner. If present, and the maximumDuration is not present, the conference is terminated when this user disconnects, otherwise this information can be used for billing or other purposes.
FloorControlAlg-orithm	String	Yes	If present, defines the floor policy to be used (e.g. chair-controlled, algorithm-based). If not present, FCFS is used by default.
MaxNoOfFloorHldrs	Integer	Yes	Defines how many users can hold the floor simultaneously. If not present, the default value is 1.
MaxHldTime	Integer	Yes	Defines the maximum time a participant can hold a floor.

Table 4.1: Input message: createConferenceWithFloorControlRequest

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the initiated conference.
FloorIdentifier	String	No	Identifies the floor in the conference.

Table 4.2: Output message: createConferenceWithFloorControlResponse

Create_Floor()

This API is used to create and add a floor to an already-initiated multimedia conference. The floor is associated with a set of resources that are used in the conference. The reference to the new floor created in the existing multimedia conference is returned in the output parameter. Tables 4.3 and 4.4 detail the selected parameters for the request and response of the method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is to be added.
FloorControlAlg-orithm	String	Yes	If present, defines the floor policy to be used (e.g. chair-controlled, algorithm-based. If not present, FCFS is used by default.
MaxNoOfFloorHldrs	Integer	Yes	Defines how many users can hold the floor simultaneously. If not present, the default value is 1.
MaxHldTime	Integer	Yes	Defines the maximum time a participant can hold a floor.

Table 4.3: Input message: createfloorRequest

Parameter name	Parameter type	Optional	Description
FloorIdentifier	String	No	It is used to identify the floor in the existing conference.

Table 4.4: Output message: createfloorResponse

4.2.1.2 Adding Participants to a Floor

Similarly, participants can be added to floor in the following two ways:

- A participant is added to the floor and the conference simultaneously.

- A participant is added to the floor after he/she has joined the conference.

This is achieved by the following proposed APIs:-

Add_Participant_To_Conference_and_Floor()

The existing Invite_Partcipant() API of Parlay X multimedia conferencing is extended to add the participant to the existing conference and floor simultaneously. Only conference participants added to the floor can request the floor. Tables 4.5 and 4.6 detail the selected parameters for the request and response of the method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the participant is to be added.
FloorIdentifier	String	No	Identifies the floor to which the participant is to be added.
UserIdentifier	String	No	Identifies the participant.

Table 4.5: Input message: addPartcipantToConferenceAndFloorRequest

Parameter name	Parameter type	Optional	Description
None	n/a	n/a	n/a

Table 4.6: Output message: addPartcipantToConferenceAndFloorResponse

4.2.1.3 Floor Management APIs

The following APIs are used to remove the floor, revoke the floor, set the floor chair and remove participant from the floor.

Remove_Floor()

This request is used to remove the existing floor from the conference, so that a media resource associated to the floor is not floor-controlled anymore.

Tables 4.11 and 4.12 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference from which floor is to be removed.
FloorIdentifier	String	No	Identifies the floor to be removed, as multiple floors can exist in same conference.

Table 4.7: Input message: removeFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.8: Output message: removeFloorResponse

Revoke_Floor()

This request is used by the application to revoke the floor from the floor participant. However, the floor gets auto-revoked if the participant has exceeded the holding time limit (specified in the create floor request). Tables 4.13 and 4.14 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated.
FloorIdentifier	String	No	Identifies the floor to be revoked.
UserIdentifier	String	No	Identifies the participant.

Table 4.9: Input message: revokeFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.10: Output message: revokeFloorResponse

Set_FloorChair()

This request is to set the chairperson who will manage the floor (i.e. grant, deny or revoke a floor). Tables 4.15 and 4.16 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated.
FloorIdentifier	String	No	Identifies the floor whose chair is being set.
ChairIdentifier	String	No	Identifies the chair of the floor.

Table 4.11: Input message: setFloorChairRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.12: Output message: setFloorChairResponse

Remove_Parcticipant_From_Floor()

This request is used to remove the conference participants from the existing floor in the conference. Tables 4.17 and 4.18 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference from which the participant is to be removed from the floor.
FloorIdentifier	String	No	Identifies the floor from which the participant is to be removed.
UserIdentifier	String	No	Identifies the participant.

Table 4.13: Input message: removePartcipantToFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.14: Output message: removePartcipantToFloorRequest

4.2.2 Proposed Client-side Floor Control APIs

The client applications access the floor capabilities (e.g., requesting a floor, releasing a floor, granting a floor, denying a floor and getting floor information) via these APIs depending on their roles (i.e. chair or a regular floor participant).

Request_Floor()

This request is used by the floor participant to request the floor. Participants with the floor can share their data in the conference. Tables 4.19 and 4.20 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated
FloorIdentifier	String	No	Identifies the floor being requested.
UserIdentifier	String	No	Identifies the participant requesting the floor.

Table 4.15: Input message: requestFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.16: Output message: requestFloorResponse

Release_Floor()

It is used by the participant to release the floor and make it available to other users. Tables 4.21 and 4.22 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated.
FloorIdentifier	String	No	Identifies the floor to be released.
UserIdentifier	String	No	Identifies the participant requesting the floor.

Table 4.17: Input message: releaseFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.18: Output message: releaseFloorResponse

Subcribe_Floor_events() This request is used by floor participants to subscribe to floor control events in order to be notified of the changes in the floor status. Tables 4.23 and 4.24 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated.
FloorIdentifier	String	No	Identifies the floor whose events are subscribed.
UserIdentifier	String	No	Identifies the participant.

Table 4.19: Input message: subscribeFloorEventsRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.20: Output message:subscribeFloorEventsRequestResponse

Revoke_Floor

This request is used by the floor chair to revoke the floor from the floor participant. However, the floor gets auto-revoked if the participant has exceeded the holding time limit. Tables 4.25 and 4.26 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated.
FloorIdentifier	String	No	Identifies the floor to be revoked.
UserIdentifier	String	No	Identifies the participant.
ChairIdentifier	String	No	Identifies the floor chair.

Table 4.21: Input message: revokeFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.22: Output message: revokeFloorResponse

Grant_Floor()

This request is used by the floor chair to grant the floor to the floor participant who has requested the floor. Tables 4.25 and 4.26 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated.
FloorIdentifier	String	No	Identifies the floor to be granted.
UserIdentifier	String	No	Identifies the participant.

ChairIdentifier	String	No	Identifies the floor chair.

Table 4.23: Input message: grantFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.24: Output message: grantFloorResponse

Deny_Floor()

This request is used by the floor chair if he decides to reject the floor request.

Tables 4.25 and 4.26 detail the selected parameters for this method.

Parameter Name	Parameter Type	Optional	Description
ConferenceIdentifier	String	No	Identifies the conference to which the floor is associated.
FloorIdentifier	String	No	Identifies the floor.
UserIdentifier	String	No	Identifies the participant.
ChairIdentifier	String	No	Identifies the floor chair.

Table 4.25: Input message: denyFloorRequest

Parameter Name	Parameter Type	Optional	Description
None	n/a	n/a	n/a

Table 4.26: Output message: denyFloorResponse

4.3 Illustrative Scenarios

This section studies a few scenarios to show how the floor control service in multimedia conferencing can be realized using the proposed architecture.

4.3.1 Scenario: Creating a Multimedia Conference with Floor Control

Figure 4.3 shows the sequence diagram to create an empty dial-out multimedia conference with floor control using the SOAP interface. The conferencing application sends a SOAP request to the conferencing gateway, along with the conference and floor information (e.g. maximum number of participants ,conference duration, floor policy, and maximum number of floor holders) required to create a new conference configured with floor control (step 1).The request is received and validated by the conferencing gateway. The request is then processed by the gateway which creates a new conference object with the provided conference information, that is stored locally (step 2). Then it sends the SIP INVITE in conjunction with MSCML message to the media server to reserve resources for the new conference (steps 3, 4 and 5). Next, the gateway creates and stores the floor object, and associates the resources to the floor (step 6). It then opens a floor control connection with the FCS using SIP Invite (steps 7, 8 and 9). Next, the conferencing gateway forwards the request to the FCS in order to create a floor with the provided floor information using SIP-INFO in conjunction with FSCML messages (steps 10, 11, 12 and 13). Finally, the gateway provides the both the conference and floor identifiers to the conferencing application in the response (step 14).

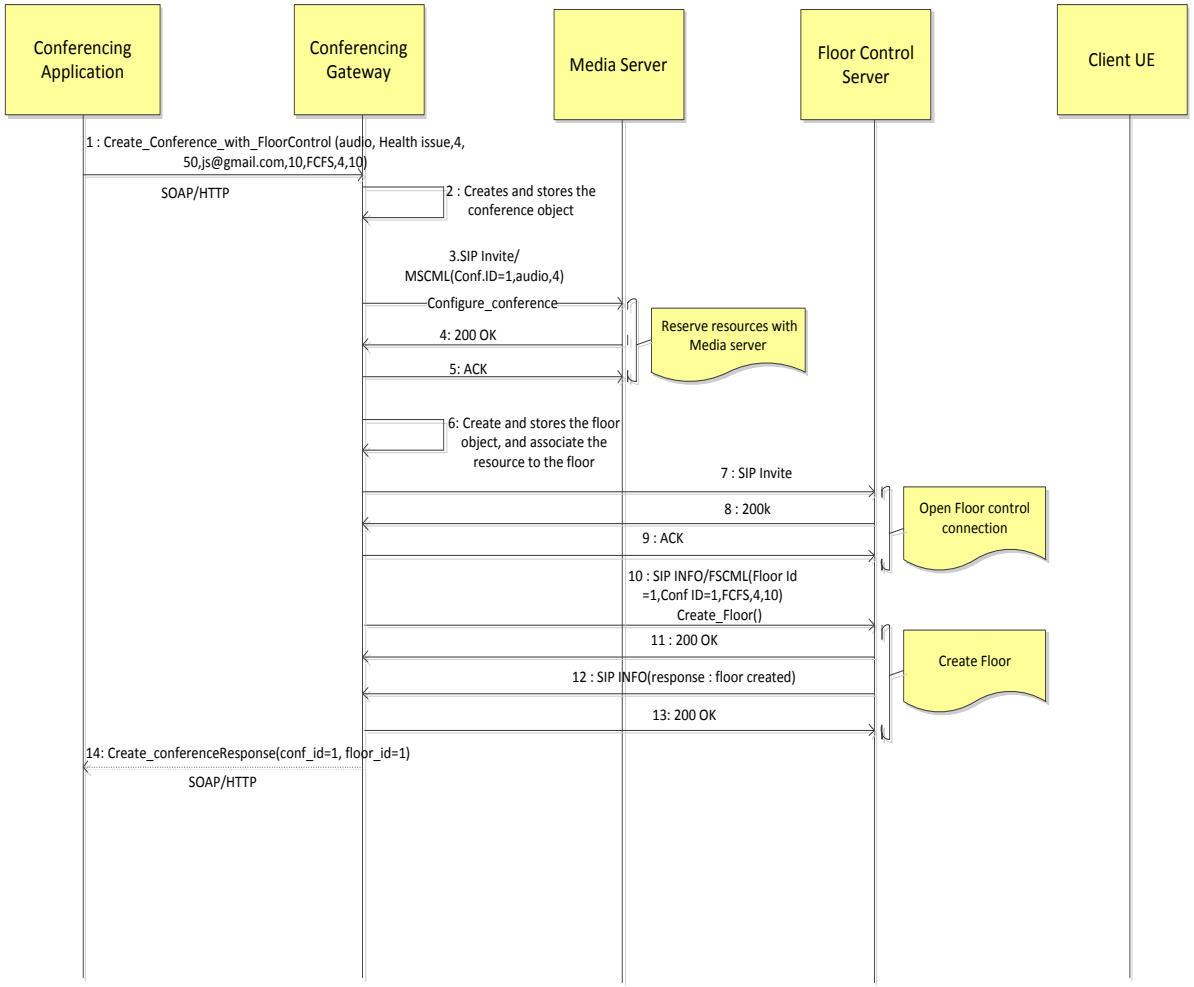

Figure 4.2 : Scenario: Creation of multimedia conference with floor control

4.3.2 Scenario: Adding participant to conference and floor

Figure 4.4 illustrates a sequence diagram to add a new participant to an existing multimedia conference and floor. Using the SOAP interface, the conferencing

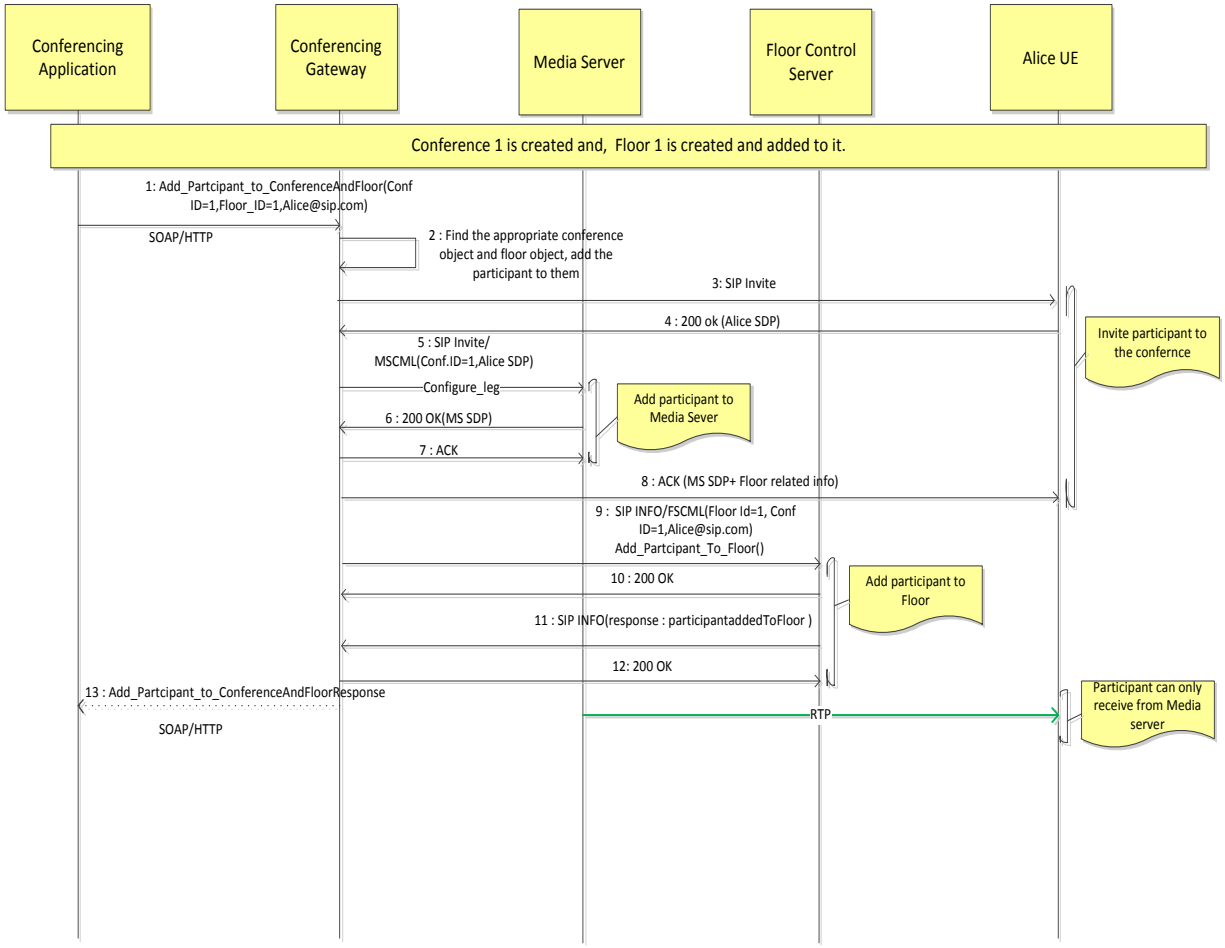

Figure 4.3: **Scenario: Adding participant to conference and floor**

application sends the request to the conferencing gateway with the URI of the participant and the identifiers of the conference and floor to which the participant should be added (step 1). The request is received by the conferencing gateway which verifies that the target conference and floor exists. The request is rejected if the target conference does not exist .It is also rejected if the conference has reached the maximum number of participants specified in the conference creation request (step 2). The participant is only added to the floor, if the specified floor already exists in the conference. The conferencing gateway invites the participant to the conference by sending an SIP INVITE message (step 3). It moderates the

negotiation of the session description information (e.g. IP address, media codec and port number) between the participant and the media server (steps 4, 5, 6 and 8). It also provides the floor information (e.g., floor identifier, floor resource associations) to the participant along with media server SDP (step 8). The participant is added to the conference with the receive_only RTP mode (i.e., participant can only receive from media server). Next, the conferencing gateway sends the request to FCS, to add participant to the floor using the SIP FSCML messages (steps 9, 10, 11 and 12). Finally, the conference application is notified that the participant has been added to the conference and the floor (step 13).

4.3.3 Request Floor and Release Floor (scenario)

Figure 4.5 shows how a floor participant requests a floor, obtains it, and, at a later time, releases it. The conference is assumed to be configured with FCFS (First come first serve) floor policy, such that floor is granted by the FCS itself following FCFS algorithm.

The client application uses a SOAP interface to send the floor request to the conferencing application with floor information (floor identifier, conference identifier and user identifier) (step 1).The conferencing application maps the request on the appropriate API and forwards it to the conferencing gateway (step 2). The conferencing gateway verifies the request and finds the appropriate floor object. The request is rejected if the target floor does not exist in the conference. The conferencing gateway requests the floor from the FCS using a BFCP message with the provided floor information. The FCS is responsible for granting the floor following the first come first serve (FCFS) floor policy (steps 4, 5 and

6). The conferencing gateway then sends an SIP re-invite to the participant and communicates with the media server using SIP Info messages to update the media properties for the participant based on the floor request decision.

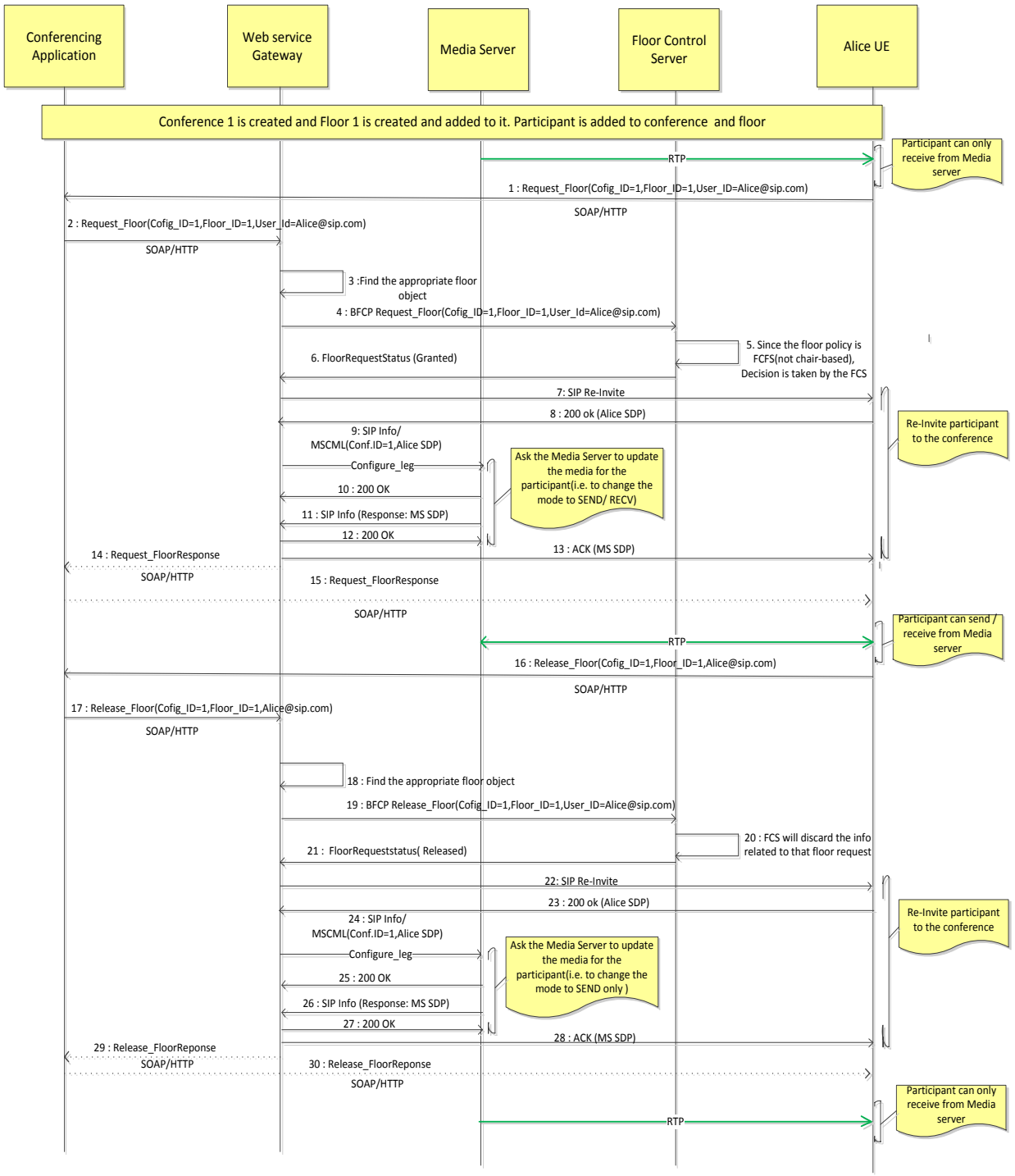

Figure 4.4: Scenario: Request Floor and Release Floor

(steps 7-to-13). A new RTP connection is established between the participant and the media server, such that the participant can now send/receive to/from the media sever. The client application is notified that the participant has been granted the floor (steps 14 and 15).

Later on, the participant sends the request to the conferencing application to release the floor using the SOAP interface (step 16). The request is forwarded to the conferencing gateway which verifies that the target floor exists (steps 17 and 18). The conferencing gateway communicates with FCS to release the floor using BFCP messages (steps 19, 20 and 21). The conferencing gateway then sends an SIP re-invite to the participant and communicates with the media server using SIP Info messages to update the media properties for the participant based on the floor request decision (steps 22-to-28). A new RTP connection is established between the participant and media server, where the participant can only receive from the media server. The client application is notified that floor is released (steps 29 and 30).

4.3.4 Scenario: Revoke Floor by application

The conferencing application can revoke the floor from a participant in order to make it available for other users. Figure 4.6 illustrates a scenario where the application revokes the floor from the current floor holder.

The conferencing application uses the SOAP interface to send request to the conferencing gateway with floor information (floor identifier, conference identifier and user identifier) (step 1). The conferencing gateway verifies the request and finds the appropriate floor object (step 2). The conferencing gateway

communicates with the FCS to revoke the floor using BFCP messages (step 3, 4 and 5). Then based on the decision, the conferencing gateway then sends an SIP re-invite to the participant and communicates with the media server using SIP Info messages to update the media properties for the participant based on the floor

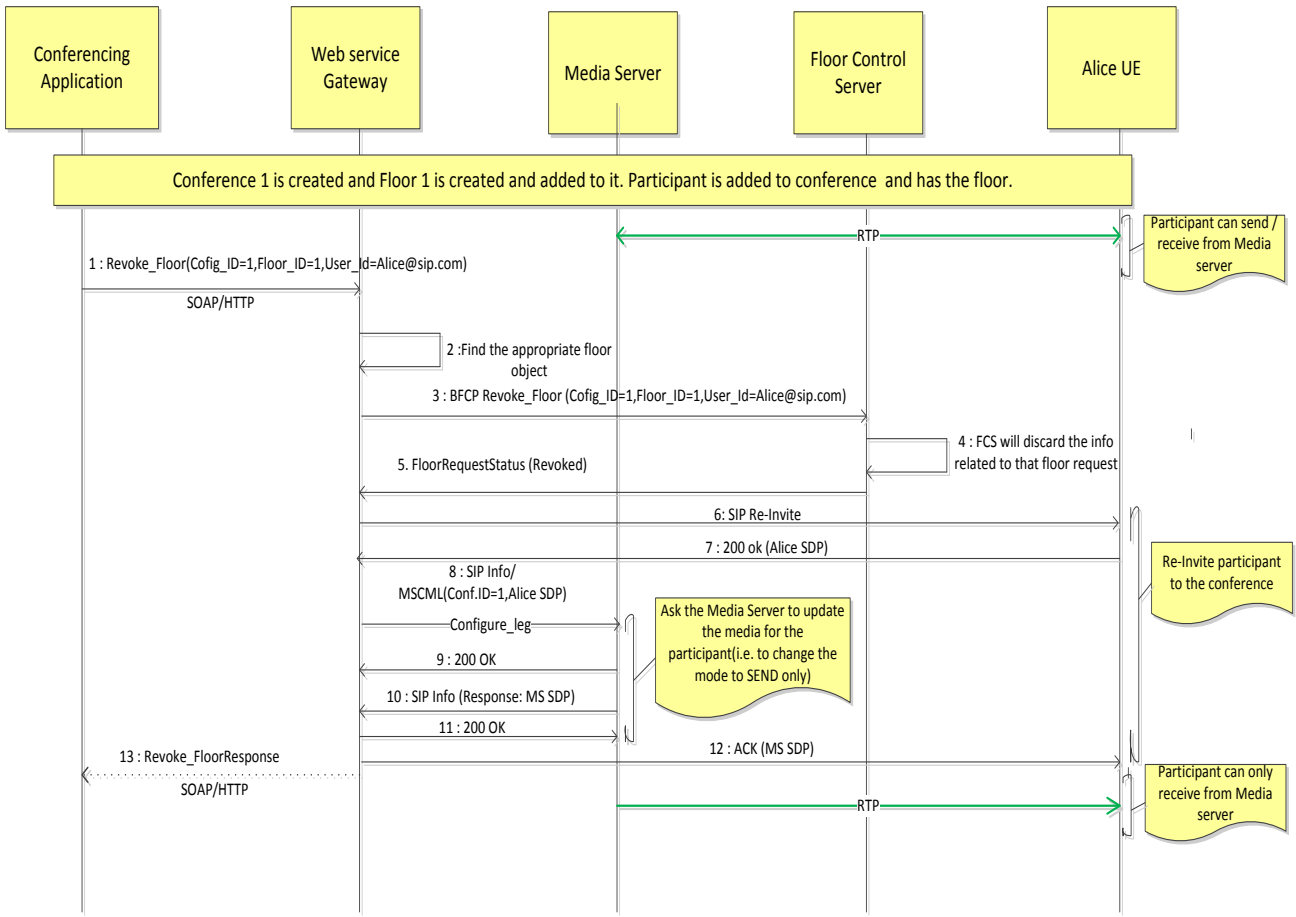

Figure 4.5: Scenario: Revoke floor by application

request decision (steps 6-to-12). A new RTP connection is established between the participant and media server where the participant can only receive from the media server. The conferencing application is notified that the floor is revoked (step 13).

4.3.5 Scenario: Subscribe to Floor Events and Set Up Notifications

Figure 4.6 illustrates a scenario where a participant subscribes to floor events, is assigned as the chair of the floor and is then notified by the application.

The client application uses the SOAP interface with the conferencing application to subscribe for floor events (step 1). The request includes floor information (floor identifier, conference identifier and user identifier). The conferencing application forwards the request to the conferencing gateway, which is responsible for processing the request (steps 2 and 3). The response is sent back to the client (steps 4 and 5). Next, using the SOAP interface, the conferencing application sends the request with information (e.g., floor identifier, conference identifier, identifier for participant to be selected as chair) to the conferencing gateway to set the chair for the floor (step 6 and 7). The conferencing gateway verifies the request and updates the FCS with the provided floor information using SIP-FSCML messages (step 9, 10, 11 and 12).The conferencing gateway then notifies the selected floor chair using an SIP Notify message (step 14 and 15).

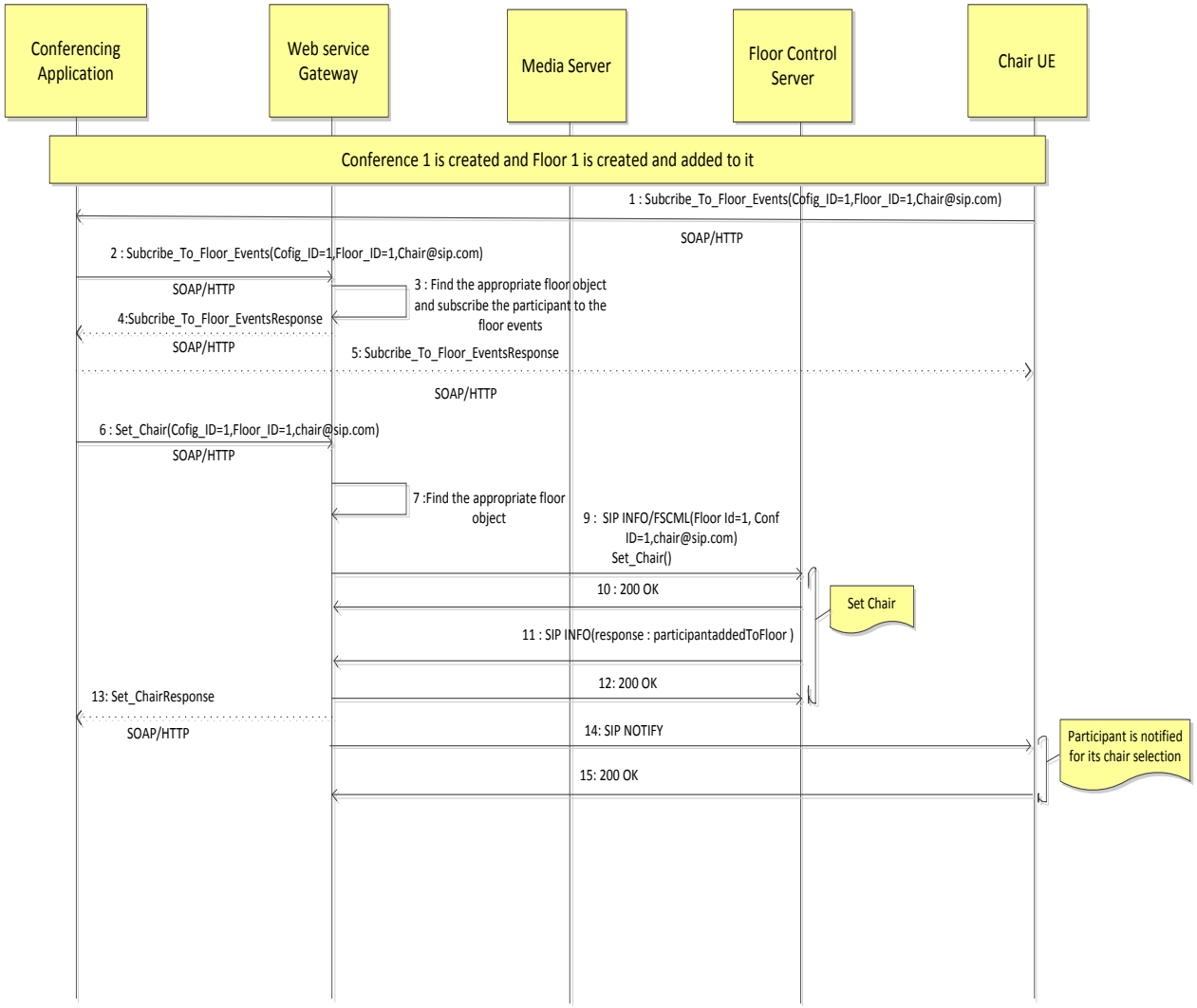

Figure 4.6: Scenario: Subscribe to Floor Events and Set Up Notifications

4.3.6 Scenario: Request Floor When Floor Control Policy is Chair-controlled

For the scenario [Figure 4.7], it is assumed that the conference is configured with chair moderated floor control policy .Therefore, the floor is granted by the designated chair of the floor. Furthermore, it is also assumed that the chair of the floor is set and the selected chair has also subscribed for the floor event notifications.

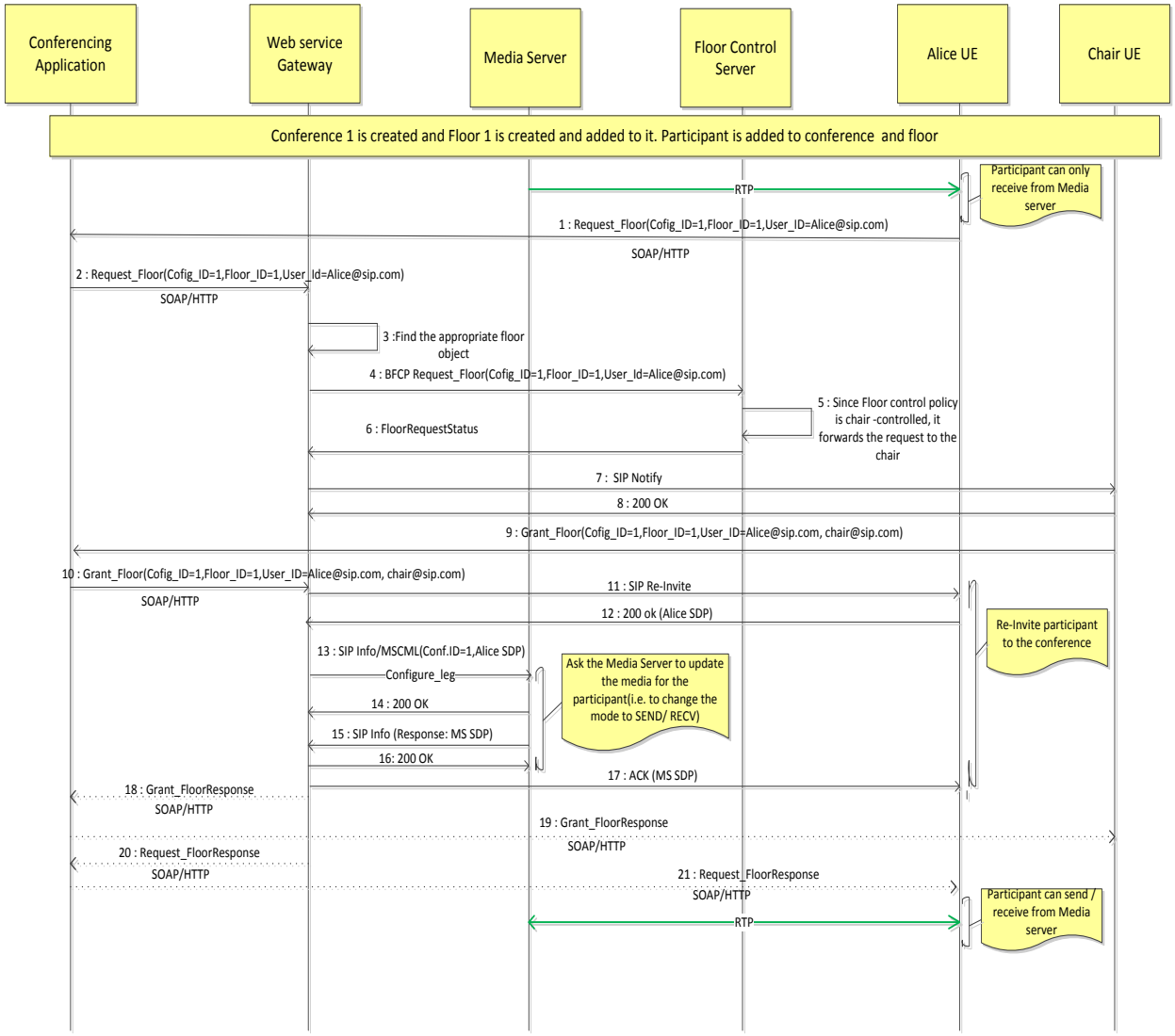

Figure 4.7: Scenario: Request floor, when floor policy is Chair-controlled

The client application uses the SOAP interface to send floor request to the conferencing application with floor information (floor identifier, conference identifier and user identifier) (step 1). The conferencing application forwards the request to the conferencing gateway (step 2). The conferencing gateway verifies the request and finds the appropriate floor object (step 3). The request is rejected if the target floor does not exist in the conference. The conferencing gateway requests the floor from the FCS using a BFCP message with the provided floor

information (steps 4, 5 and 6). The FCS forwards the request to the designated chair of the floor because the floor policy adopted for the floor is chair-controlled. The conferencing gateway then notifies the chairperson using SIP Notify (steps 7 and 8). The chair uses the SOAP interface to grant the floor by forwarding the request to the conferencing application (step 9). The request is forwarded to conferencing gateway (step 10). Next, the conferencing gateway sends an SIP re-invite to the participant and communicates with the media server using SIP Info messages to update the media properties for the participant based on the floor request decision (steps 11-to-17). A new RTP connection is established between the participant and the media server, where participant can send/receive to/from the media server. The floor client and the chair are notified that the floor is granted (steps 18-to-21).

4.4 Chapter Summary

In this chapter, we presented the overall architecture for floor control, which included the main functional components such as conferencing application, conferencing gateway, floor control server (FCS), media server (MS) and client user equipments (UEs). The communication interfaces between the system entities are categorized as: SOAP interfaces, floor control interfaces, media interfaces and signaling interface. The SOAP interface is the main interface between the system entities. It is used to establish the communication between the client application and the conferencing application, and the conferencing application and the conferencing gateway. We also concluded that the architecture

satisfies all the requirements derived in the previous chapter. We then discussed the proposed SOAP Web service-based floor APIs for the client side and the server side. Finally, we demonstrated the interaction between the entities by presenting few illustrative scenarios.

In the next chapter, we will present the implementation architecture of the system components. It will be followed by the implemented proof of concept prototype and includes some performance measurements

Chapter 5

Validation: Prototype and Evaluation

This chapter is broken down into three sections. In the first section we present the implementation architecture followed by the illustrative scenarios that show how entities interact. The next section presents the proof of concept prototype that we have implemented. Lastly, we discuss some performance measurements in order to validate our architecture.

5.1 Implementation Architecture

Figure 5.1 depicts the implementation architecture for SOAP Web services-based floor control in multimedia conferencing. The key components implemented are conferencing gateway, conferencing application, floor control server, media server and client UE. They are discussed below:

Conferencing Gateway

The conferencing gateway architecture is composed of three layers: API layer, processing layer and communication layer. The API layer exposes the network conferencing and floor capabilities toward the application server. It includes a SOAP request handler module that receives SOAP conferencing and floor request

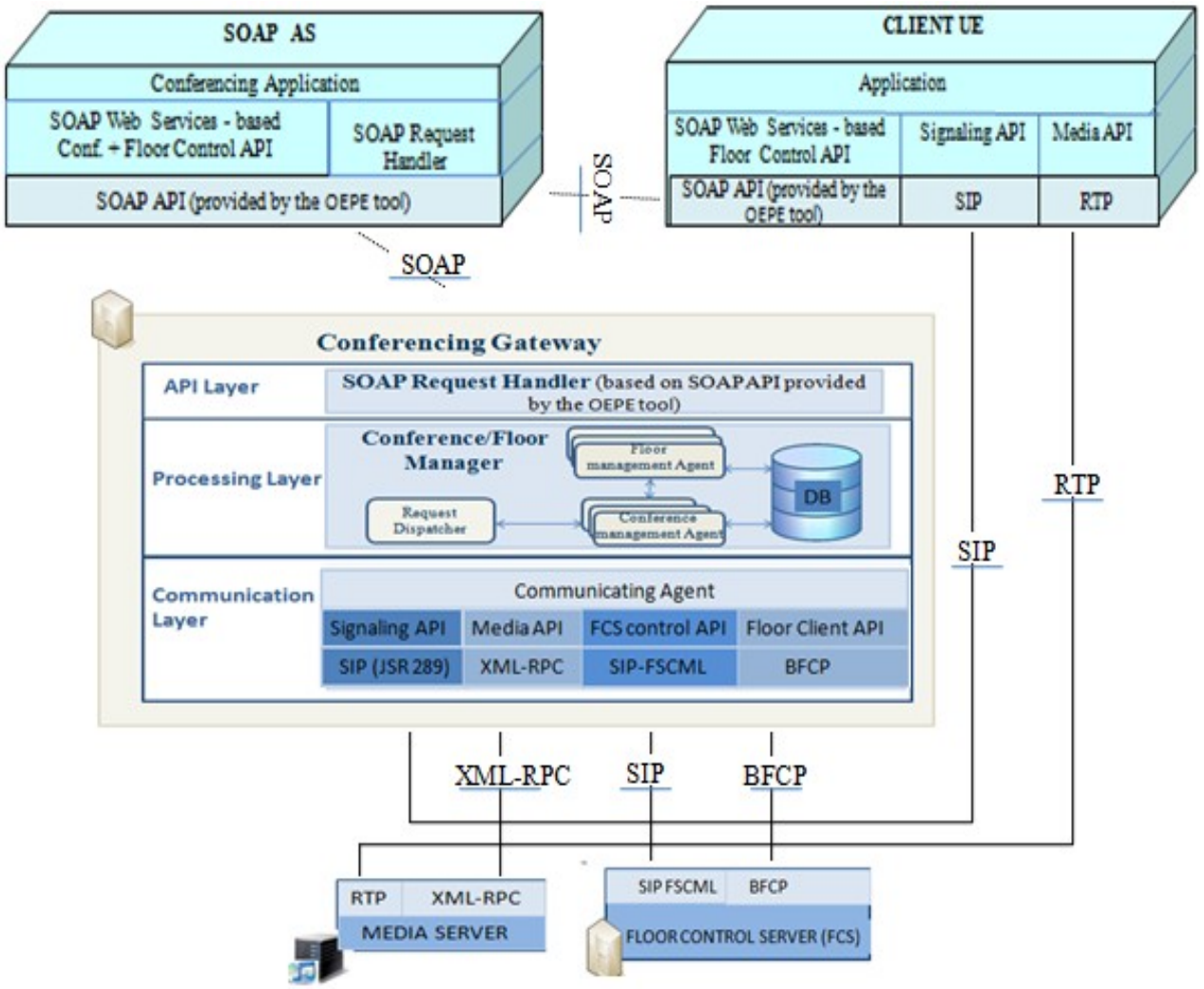

Figure 5.1: Implementation architecture

from the applications, analyses them, and then passes their content (e.g. the method to be executed and its parameters) to the conference/floor manager module in the processing layer. The SOAP request handler is responsible for creating and sending SOAP responses to the conferencing application.

The processing layer contains the conference/floor manager, which is responsible for the creation and management of the different multimedia conferences and floors associated with them. The conference/floor manager is composed of four

modules: request dispatcher, conference management agent, floor management agent and a database. The request dispatcher gets conference creation requests from the SOAP request handler and creates a new conference management agent, which creates new conferences. Each conference is managed by a separate conference management agent. The request dispatcher dispatches subsequent requests related to a given conference to the agent that created the conference. The relationships between conferences and their agents are preserved when the conferences are created.

Similarly, floor creation requests associated with a particular conference are forwarded to the appropriate conference management agent, which then creates a new floor management agent. A separate floor management agent creates the floor and manages the requests related to that floor. Both conference and floor management agents stores the conference and floor information respectively in a local database. Such information includes, for instance, the unique conference identifier, the conference type (i.e., audio, video, chat etc.), the conference status (i.e., initiated, active and terminated), as well as the participants' information (e.g. number of participants, participants' URIs, and type of media for each participant), the unique floor identifiers, the floor policy (i.e., FCFS or chair moderated), the floor status, and the floor holders.

The communication layer includes a communicating agent module that handles the message exchanges between the conferencing gateway and the other entities in the network (e.g. media server, floor control server and the end-users). The

communicating agent is supported by four types of APIs: signaling API, media API, FCS control API and floor client API.

We have used the JSR 289[28] standard for the SIP-based signaling APIs. XML-RPC [25] APIs are used for the media APIs to control the media server.

The floor control APIs, which include floor control server (FCS) control API and floor client API, are used to communicate with the FCS to expose its capabilities. The FCS control API, based on SIP-FSCML [8], is used to control the FCS. The floor client API is used to communicate with the FCS to expose the client-side floor capabilities (e.g. request floor, release floor, revoke floor, grant floor, deny floor and floor query) via BFCP [3] protocol.

Client UE

The client UE relies on the SOAP API to access the floor capabilities from the conferencing application. SIP is used as the signaling protocol and Real-Time Transport Protocol (RTP) is used for media handling of the client.

Conferencing Application

The conferencing application uses SOAP APIs to access the conferencing and floor capabilities from the conferencing gateway. It includes a SOAP request handler module that receives SOAP floor requests from the client applications, analyzes and maps them to the appropriate SOAP API, and then sends them to the conferencing gateway.

Floor Control Server

We have reused the FCS implementation architecture from one of the previous work done by our research team [8]. It supports both BFCP and SIP-FSCML.

Media Server

Any commercially-available media server on the market (e.g. Medooze, SIP Express Media Sever) that provides multimedia conferencing capabilities can be used to support the media mixing of the clients.

5.1.1 Illustrative Scenarios

This sub-section studies a few scenarios to show how the floor control service in multimedia conferencing can be realized using our implementation architecture.

5.1.1.1 Scenario: Creating a Multimedia Conference with Floor Control

Figure 4.3 shows the sequence diagram to create an empty dial-out multimedia conference with floor control via a SOAP interface. The conferencing application sends a SOAP request to the conferencing gateway, along with the conference and floor information (e.g. maximum number of participants, conference duration, floor policy, and maximum number of floor holders) required to create a new conference configured with floor control. The request is first received and validated by the SOAP request handler (step 1). Next, the request handler passes the request content to the request dispatcher (step 2). The request dispatcher creates a new conference management agent, and assigns it the task to create a

new conference (steps 3, 4 and 5). The conference management agent stores the conference object (with conference information) in a local database and then uses the communication agent to send an XML-RPC message to the media server to reserve resources for the new conference (steps 6, 7, 8 and 9). Next, the conference management agent creates a new floor management agent and assigns it the task to create a new floor with the provided floor information (steps 11, 12 and 13). The floor management agent creates the floor object and stores it in the database (step 14). Then, it passes the request content to the communicating agent by calling the appropriate API (step 15). The communicating agent opens the floor control signaling session with the FCS through an SIP INVITE message (steps 16, 17 and 18). Next, it forwards the request to the FCS in order to create the floor (with the provided floor information) using SIP-FSCML (i.e. SIP INFO messages) (steps 19, 20, 21 and 22). The responses are then sent back to the conferencing application with respective identifiers of the created conference and floor (steps 23, 24, 25, 26 and 27).

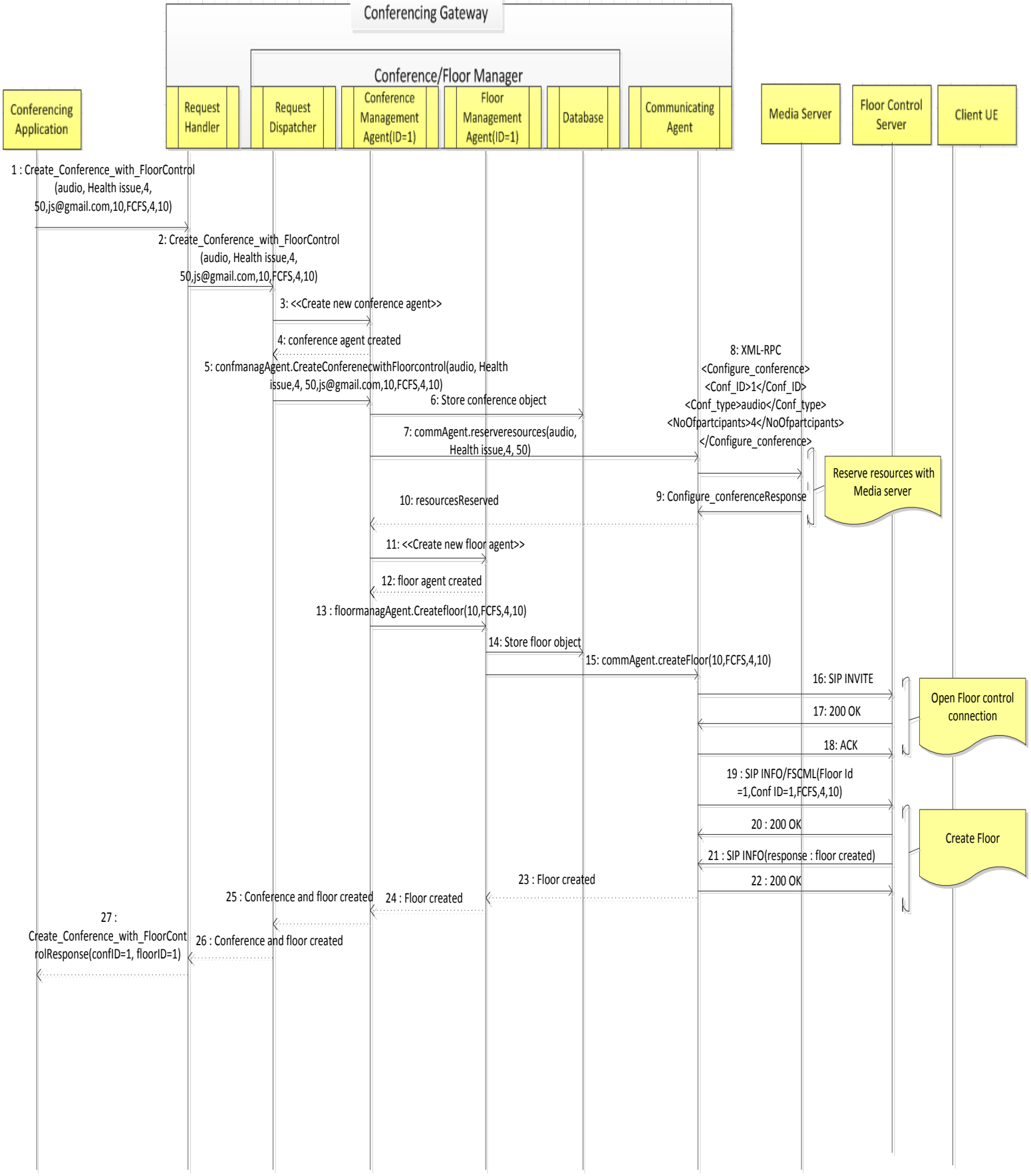

Figure 5.2: Creating a Multimedia Conference with Floor Control

5.1.1.2 Scenario: Adding a Participant to a Conference and Floor

Figure 5.4 illustrates a sequence diagram to add a new participant to an existing multimedia conference and floor. Using the SOAP interface, the conferencing application sends the request to the conferencing gateway with the URI of the participant and the identifiers of the conference and floor to which the participant should be added (step 1). The request is received by the SOAP request handler, which validates the request and then passes the request content to the request dispatcher (step 2).The request is rejected if the conference does not exist or if it has already reached the maximum number of participants (specified in the conference creation request). The request dispatcher searches for the appropriate conference management agent and forwards the request to it (steps 3, 4 and 5). The conference management agent then uses the communicating agent to send an SIP INVITE to the participant (steps 6 and 7). The communicating agent moderates the negotiation of the session description information (e.g. IP address, media codec and port number) between the participant and the media server. The RTP connection established between the participant and the media server is unidirectional (i.e. the participant can only receive from the media server) (steps 8, 9, 10 and 11). The conference management agent updates the database with the information (step 13). Next, it searches for the appropriate floor management agent and forwards it the request to add a participant to the floor (steps 14, 15 and 16). The floor management agent calls the appropriate API on the communication

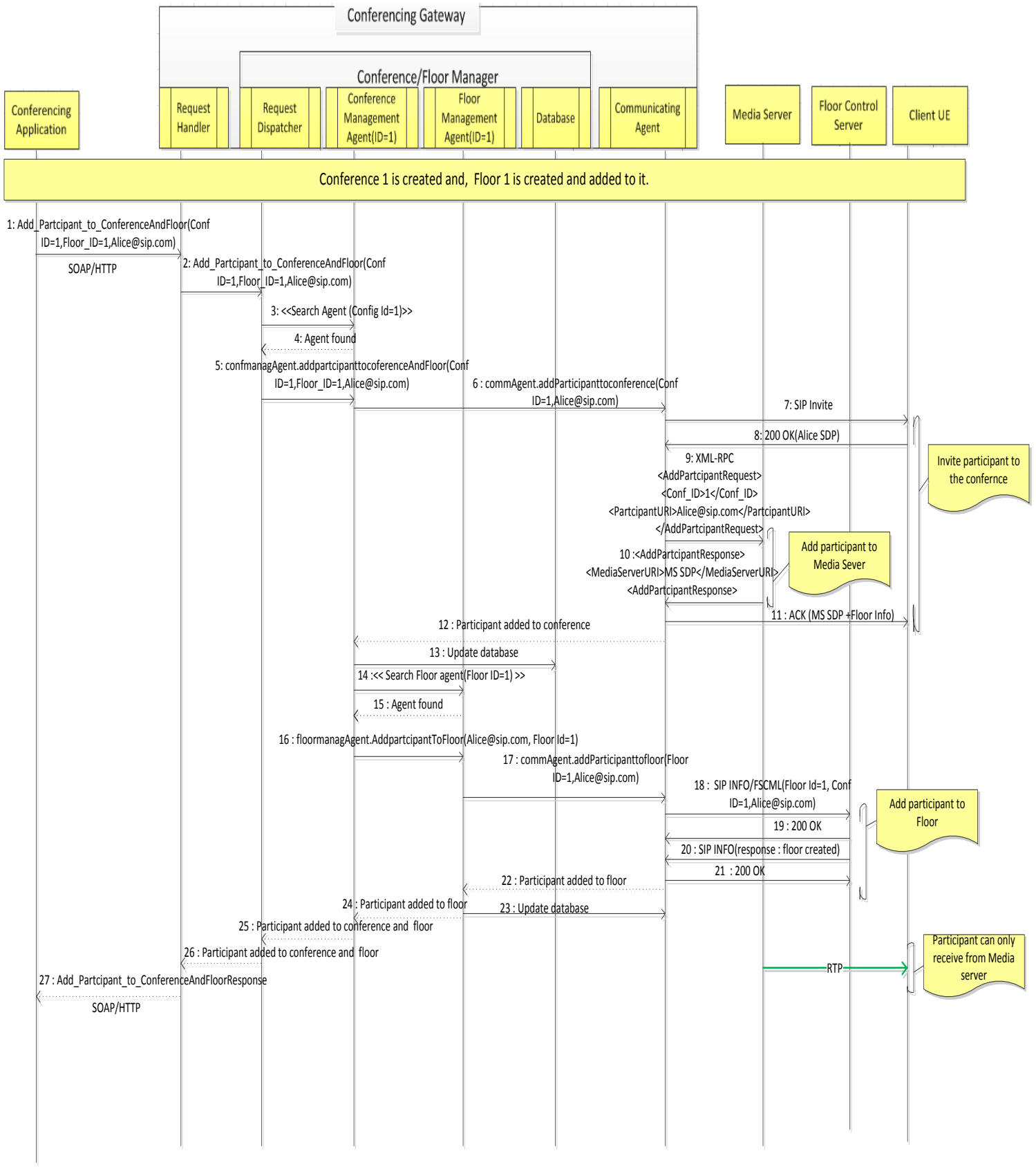

Figure 5.3: Adding a participant to a conference and floor

agent, which sends the request to the FCS to add the participant to the floor using SIP-FSCML messages (steps 17-21). The floor management agent updates the database when the participant is added to the floor (steps 22 and 23). Finally, the conference application is notified that the participant has been added to the conference and the floor (steps 24-27).

5.1.1.3 Scenario: Requesting the Floor

Initially, all the participants are added to the conference with a "receive_only" RTP mode, so they can only receive from the media server. Participants must request the floor in order to share their data in the conference.

Figure 4.5 shows how a floor participant requests a floor and obtains it. The client application uses a SOAP API to send a floor request with floor information (e.g. floor identifier, conference identifier and user identifier) to the conferencing application (step 1). The request is first received and validated by the SOAP request handler module in the conferencing application. It is then mapped to the appropriate SOAP Web service based API and sent to the conferencing gateway. The SOAP request handler validates the request once it is received at the conferencing gateway, and then passes the request content to the request dispatcher. The request dispatcher searches the appropriate conference management agent and then forwards the request to it. The conference management agent then searches for the appropriate floor management agent and forwards the request to it. The floor management agent uses the communicating agent to send the floor request to the FCS via BFCP messages. Once the floor is granted by the FCS (assuming that the floor control policy is FCFS), the

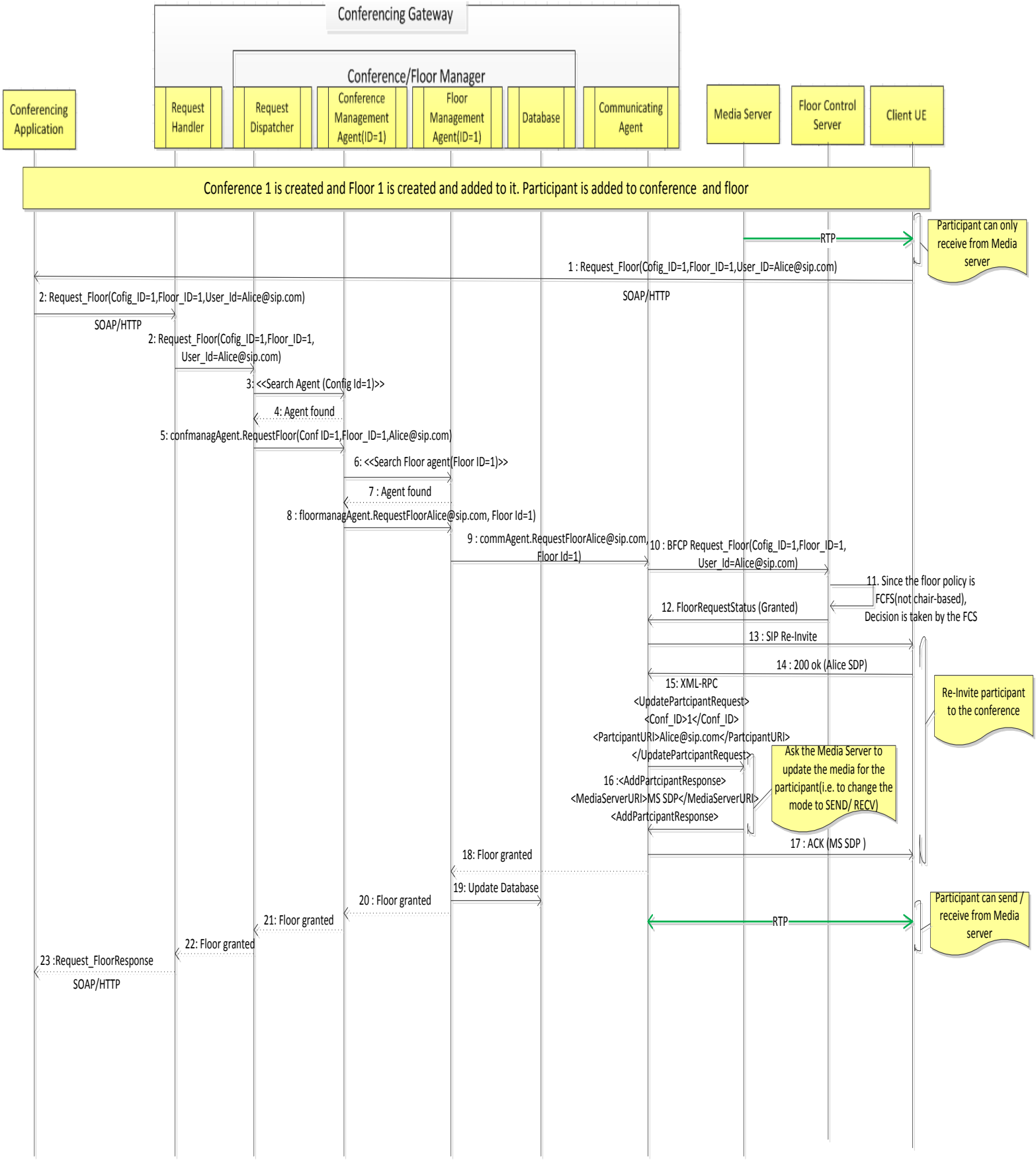

Figure 5.4: Requesting the floor

communicating agent then updates the media properties of the client requesting the floor. The communicating agent then sends an SIP RE-INVITE to the participant and communicates with the media server using XML-RPC messages to update the media connection between the media server and the participant, so that the participant can send and receive from the media server. The floor management agent then updates the database on receiving the response from the communicating agent, and the client application is notified that the requested floor has been granted.

5.2 Prototype

This sub-section discusses the implemented components, the prototype capabilities and the graphical user interfaces (GUIs) involved in the prototype.

5.2.1 Implemented Components

A proof-of-concept prototype is based on the implementation architecture [Figure 5.1] and tested using various scenarios. The prototype includes the SOAP-based conferencing application, a set of client UEs, a conferencing gateway, a media server and a floor control server.

The conferencing application is developed as a Web application using an Eclipse environment and Oracle Enterprise Pack for Eclipse (OEPE), a set of Eclipse plug-ins designed to support application development for Oracle WebLogic application server [29]. The SOAP request handler module and conferencing and floor APIs are based on a SOAP API provided by OEPE, The application is

deployed on Oracle WebLogic Server 11gR1 [29], which natively supports SOAP applications.

Similarly, the client application is developed and deployed the in same environment as the conferencing application. Floor control APIs use the SOAP API provided by OEPE. X-Lite soft phones [30] are used as media participants in the client UE, which provide the SIP-based signaling and RTP-based media APIs.

Medooze [31] is the media server used for the media mixing of the participants in the conference. It is controlled by the conferencing gateway using XML-RPC APIs.

The conferencing gateway is fully implemented and is deployed on Oracle WebLogic Server 11gR1. The SOAP request handler is based on the SOAP API provided by OEPE. We have used JSR 289 APIs [32] to provide SIP signaling between the conferencing gateway and end users, as Oracle WebLogic Server 11gR1 supports the full implementation of JSR 289.

The floor control server is fully implemented as an independent box that supports both BFCP [3] and SIP FCSML [8] protocols.

5.2.2 Prototype Capabilities

The prototype starts with an empty dial-out multimedia conferences with/without floor control and then participants are added one by one to the conferences and floors, following the Parlay-X conferencing service specifications. Initially, all participants are added to the conference in “receive-only” RTP mode (i.e. they

can only receive from the media server). Participants must request the floor in order share their data in the conference. According to the implemented prototype, the floor is always granted by the FCS itself using the FCFS algorithm.

A rich set of applications can be easily developed using our prototype. The prototype can create both simple conferences (i.e. a conference without policy and floor control) and floor-enabled conferences. Different conferencing and floor operations are tested including, add participant to floor and conference, remove participant from floor and conference, get conference participants, get participant information, get conference information, create floor, remove floor, revoke floor, request floor, release floor and floor query.

5.2.3 Graphical User Interfaces

A graphical user interface is developed to support the conferencing application operations as depicted in Figure 5.5. The required operation is executed when the appropriate button is clicked. For example, when users click CreateConferenceWithFloorControl operation, a form as shown in Figure 5.5 appears in a new window; users provide the required values and click Submit, and then receive a response with the conference and floor identifiers. Other operations can be invoked from the application in the same way.

Similarly, a GUI is created for the client-side operations [Figure 5.6]. Users click the required operation, fill in the required information in the pop-up form, and then submit the request.

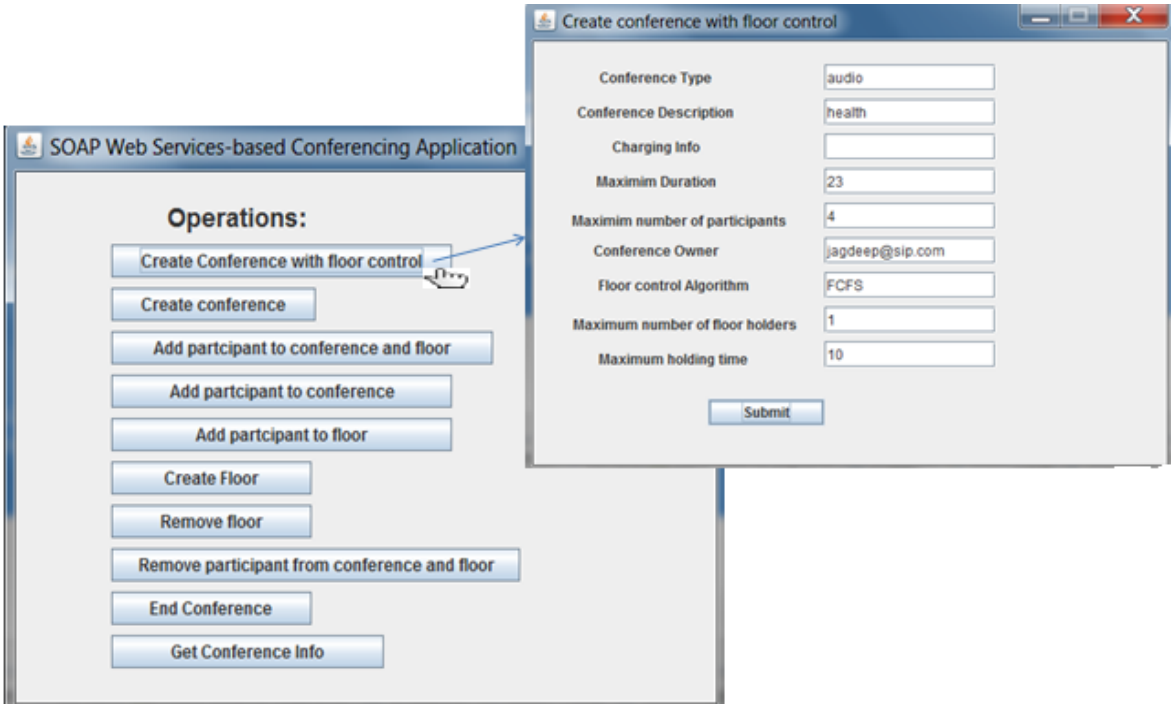

Figure 5.5: A screen shot of the Conferencing Application

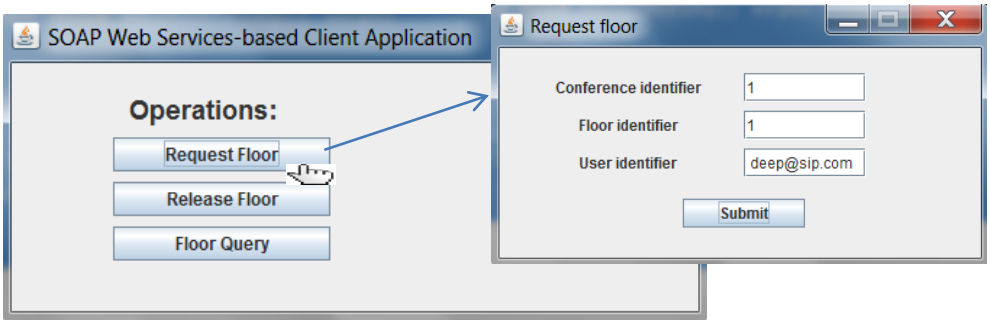

Figure 5.6: A screen shot of the Floor Client Application

5.3 Performance Measurements

In this sub-section, we first describe the experiment setup, and then present the performance metrics used. The performance results are presented and discussed at the end.

5.3.1 Experimental Setup

The experiment is set up with one SOAP conferencing application, one conferencing gateway, one media server, one floor control server and some client UEs.

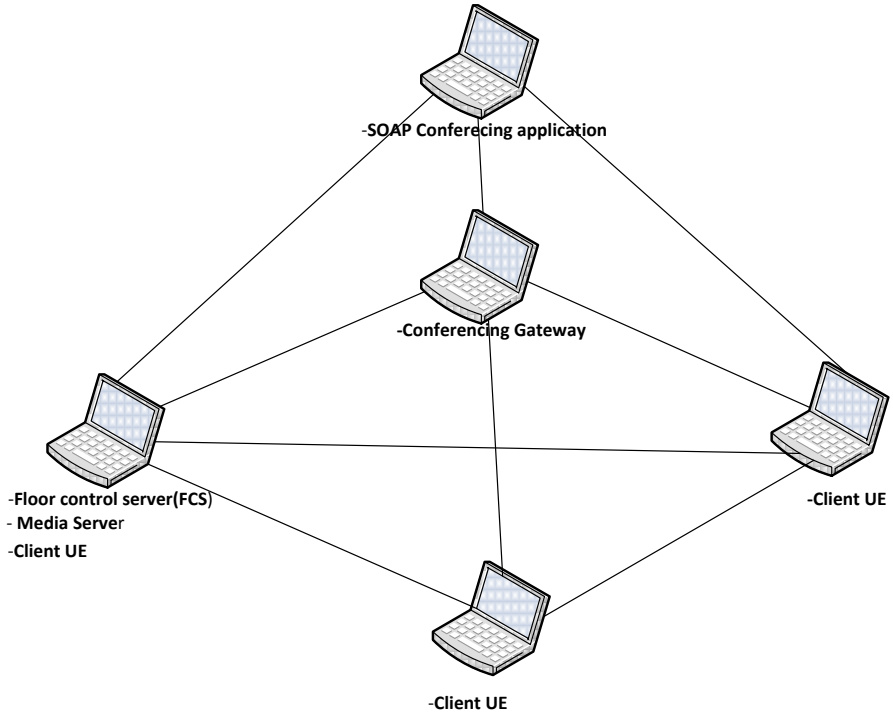

Figure 5.7: Experimental setup

The SOAP conferencing application runs on a laptop with an i3 processor and 3 GB RAM. The conferencing gateway is deployed on a second laptop, equipped with a dual-core processor and 4GB RAM. Both the FCS and media server run on a third laptop with a dual-core processor, 2 GB RAM and an Ubuntu virtual machine installed. Ubuntu is installed as a virtual machine because the media server (Medooze) is Ubuntu-deployable. This same laptop also runs the client UE.

There are two additional laptops used to support two other client UEs; one is configured with a dual-core processor and, 2 GB RAM and the other is configured with a dual-core processor and 1 GB RAM. All the laptops run on Windows 7 and are equipped with 802.11 (WLAN) and Ethernet cards.

5.3.2 Performance Metrics

The performance of the prototype is evaluated in terms of the end-to-end time delay when executing different conferencing and client application operations. The delays are measured as the difference between the time when the conferencing application sends a request and the time it receives a response from the conferencing gateway. Similarly, the delays for the client application are measured as the difference between the time the request was sent and then received from the conferencing application.

The time for creating an empty dial-out conference with floor control includes:

- Time to send a SOAP request to the gateway
- Time to send message to media server to reserve resources and to get its acceptance
- Time to send request to the FCS to create floor and to get its response
- Time to send the corresponding SOAP response back to the conferencing application

Similarly for the client application, the execution time for the request floor operation is the difference between the time when the client sends a floor request

to the conferencing application and the time when the client receives the response from the conferencing application. This includes:

- Time to send SOAP request to the conferencing application
- Time to forward the request to the conferencing gateway
- Time to process the request by the conferencing gateway (i.e. send a floor request to the FCS, and then establish the media properties between the participant and media sever depending on the FCS response)
- Time to send the SOAP response back to the client application via the conferencing application.

We have eliminated the human delays, which get introduced while responding to the invitation requests from the conferencing application.

5.3.3 Measurements Analysis

Tables 5.1 and 5.2 show the respective conferencing and client application evaluation results for the indicated operations. The delays are measured in milliseconds, and each result is calculated as the average of 10 experiments.

For the server-side operations, the delay incurred for the Create_conference_with_floor_control operation is 1638.25 ms on average, where more than 60% of the delay is due to the SOAP messages exchanged between the conferencing application and the conferencing gateway. There are basically two reasons that cause the additional delay induced by SOAP messages: Firstly, the SOAP message processing includes opening the envelope and extracting the name of the target service as well as the name and parameters of the method to be executed. Secondly, the mandatory SOAP body adds extra information (e.g.

SOAP envelope) to the message, resulting in larger network delays. The findings are well supported by our recent case study paper [33], which included performance measurements for SOAP-based and REST-based application operations.

Similarly, on the client-side, the Request_floor operation is executed in 2351 ms on average. The reason for the significant delay is the use of SOAP messages between the client application and conferencing application, and then between the

SOAP-based Conferencing Application APIs	Delay in the distributed environment
Create conference with floor control	1638.2
Create conference	1221.3
Create Floor	1370.1
Add participant to conference and floor	1711.2
Add participant to conference	1308.2
Add participant to floor	1271.1
Remove Floor	1290.3
Remove participant from conference and floor	1680.12
End Conference	1677.12
Get Conference Info	992.1

Client Application floor control APIs	Delay in the distributed environment
Request floor	2351.1
Release floor	2410.2
Floor Query	1810.6

Table 5.1 and 5.2: Performance results

conferencing application and conferencing gateway. The SOAP messages contribute 72% of the total delay. However, the delay for the execution of the floor request is quite acceptable (i.e. from 2.1 to 2.4 seconds).

Furthermore, it can be observed that creating a conference with floor control (1638.2 ms) is more efficient than creating a conference first and then adding a floor to it (1221 + 1370 ms), comparing the delays and operations required. Similarly, for adding a participant to a conference and floor simultaneously is more efficient than doing it separately. However, it sometimes depends on the requirements. For example, a conference without a floor is needed or a participant only needs to be added to a conference.

5.4 Chapter Summary

In this chapter, we have first presented the implementation architecture that follows the overall system architecture described in the previous chapter. Next, we provided illustrative scenarios that show how the entities interact for specific operations based on the implementation architecture. We have presented the prototype implemented as a proof of concept, using an OEPE environment for the development of conferencing and client applications, and Oracle WebLogic server 11gR1 to deploy the applications. Medooze media sever was used for media mixing of the SIP clients (X-Lite softphones).

To validate our proposed architecture, various conferencing and floor scenarios were tested and performance measurements were collected. The measurements show that the proposed architecture is feasible as delays incurred were quite acceptable. We analyzed that the reason behind the additional delays was due the use of SOAP messages. In the next chapter, we will summarize the contribution of the thesis and propose some additional future works.

Chapter 6

Conclusion and Future Work

In this chapter, we will first summarize the contributions of the thesis and then we will give some ideas for future work.

6.1 Summary of Contributions

Multimedia conferencing applications are an important and widely-used category of Web applications. Floor control is a significant conference control feature; it prevents conflict and ensures an optimized use of resources between the conference participants. However, current mechanisms used for exposing the floor control capabilities have shortcomings that can hinder application development.

As one of the contributions of this thesis, we have first identified a set of requirements that included both the functional and the architectural requirements for floor control in multimedia conferencing. The functional requirements outline the floor control functionality a system should provide, and the architectural requirements specify criteria that can be used to judge the operation of a system. Next, we have reviewed the most relevant related works and evaluated them based on our requirements; we have observed that none of them meet all our requirements.

Then, we proposed a novel SOAP Web service based floor control architecture in multimedia conferencing that meets all our requirements. The proposed architecture includes the main components of floor control and the interfaces between them. It also includes a comprehensive set of server-side and client-side SOAP Web service APIs that expose the floor control capabilities to application developers. We have provided illustrative scenarios that show how various components in the architecture interact. Next, we presented the implementation architecture for the components involved in the overall system architecture and discussed the operational procedures.

A proof-of-concept prototype is implemented based on the implementation architecture and tested using various scenarios. The prototype includes the SOAP-based conferencing application, a set of client UEs, a conferencing gateway, a media server and a floor control server. A rich set of applications can easily be developed using our prototype. The prototype can create both simple conferences (i.e. without floor control) and floor-enabled conferences. Different conferencing and floor operations are tested, including: add a participant to a floor and conference, remove a participant from a floor and conference, get conference participants, get participant information, get conference information, create a floor, remove a floor, revoke a floor, request a floor, release a floor and floor query. Finally, to validate our prototype, a preliminary performance evaluation of the proposed architecture has been made. Based on the results, we conclude that our architecture is a valid and promising approach for floor-controlled multimedia

applications. However, due to the nature of SOAP messages, we also observed that they are responsible for introducing additional delays that cannot be avoided.

6.2 Future Work

One of the biggest drawbacks of SOAP Web services is performance in terms of response time and in some cases, network load. Knowing that the bottleneck resides in the SOAP serialization and deserialization, one possible future work is to investigate the different mechanisms to accelerate or to avoid the serialization/deserialization of SOAP. Using RESTful Web services for exposing the floor control functionality in multimedia conferencing can be considered as an alternative. Because the use of RESTful Web services results in improved performance results as compared to SOAP Web services, due to the nature of REST. This was achieved in our recent case study [33] where a conferencing application was developed using both SOAP Web services and RESTful Web services, and their performance was evaluated.

Furthermore, to the best of our knowledge, the existing works follow the centralized conferencing model for floor control mechanism. We also have considered the centralized approach. Therefore, an interesting work item could be carried out in future works addressing the floor control issue in a non-centralized conferencing model.

Bibliography

[1] Koskelainen, P., Ott, J., Schulzrinne, H., and X. Wu, “Requirements for Floor Control Protocols”, IETF RFC 4376, February 2006.

[2] Petri Koskelainen, Henning Schulzrinne and Xiaotao Wu ,Dept. of Computer Science Columbia University New York, NY 10027, USA“ A SIP-based Conference Control Framework” , *NOSSDAV* '02.

[3] G.Camarillo, G., Ott, J., and K. Drage, "The Binary Floor Control Protocol (BFCP)", IETF RFC 4582, November 2006.

[4] Andrew Rebeiro-Hargrave and David Viamonte Solé, “Multimedia Group Communication: Push-To-Talk Over Cellular, Presence and list management concept and application” , *WILEY publication*, April 2008.

[5] LIU Hai-peng, LIAO Jian-xin and ZHU Xiao-min, “Decentralized improvement on floor control mechanism for PoC”, *Journal on Communications*, 2012.

[6] G. Camarillo, “Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams”, IETF RFC 4583 ,November 2006.

[7] “Multimedia Resource Function Controller (MRFC)- Multimedia Resource Function Processor (MRFP)Mp interface: Procedures descriptions”, 3GPP TS 23.333 version 10.3.0 Release 10, January 2012.

[8] Mohammed Al Rubaye, Fatna Belqasmi, Chunyan Fu, Roch Glitho-" A Novel Architecture for Floor Control in the IP Multimedia Subsystem of 3G Networks", *Vehicular Technology Conference, IEEE* 69th, April 2009.

[9] W3C, "Web Services Architecture", W3c Working Group Note 11 February 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/

[10] Adam Bobsworth in ACM Queue, Voll, No1.

[11] F. Curbera et al., Unraveling the Web services Web: An Introduction to SOAP, WSDL and UDDI, *IEEE Internet Computing*, Vol. 6, No2, March-April 2002, pp. 86-93.

[12] E. Newcomer, Understanding Web Services: XML, WSDL, and UDDI, Addison Wesley, 2002

[13] "Parlay X Web services". [Online]. Available: http://www.parlayx.com

[14] H Lofthouse, M J Yates and R Stretch, "Parlay X Web Services",BT Technology Journal, Volume 22 Issue 1, 2004 , Pages 81-86.

[15] "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Open Service Access (OSA); Stage 2 (Release 7)" , 3GPP TS 23.198 V7.2.0 (2007-03).

[16] Belqasmi, F. , Alrubaye, M. ,Glitho, R. and Chunyan Fu, "Design and implementation of advanced multimedia conferencing applications in the 3GPP IP multimedia subsystem", *Communications Magazine*, *IEEE*, November 2009.

[17] Belqasmi, F. ,Glitho, R. ; Chunyan Fu, "RESTful web services for service provisioning in next-generation networks: a survey", *Communications Magazine*, *IEEE* , December 2011.

[18] M. Barnes Nortel, C. Boulton Avaya, O. Levin Microsoft Corporation , "A Framework for Centralized Conferencing", IETF RFC 5239, June 2008.

[19] Parlay 5.1 specifications: Conference Call Control SCF , ETSI ES 203 915-4-5 V1.2.1, January 2007.

[20] T. Melanchuk, Ed, "An Architectural Framework for Media Server Control", IETF RFC 5567,June 2009.

[21] Mohammed Ouzzifa, Mohammed Erradia, Hassan Mountassir, "Description of a teleconferencing floor control protocol and its implementation", *ELSEVIER*, 2008.

[22] ETSI ES 202 391-12 V1.3.1 (2008-05) Open Service Access (OSA); Parlay X Web Services; Part 12: Multimedia Conference, 3GPP TS 29.199-12 v. 9.0.0, Rel. 9, Jan. 2010.

[23] H.248.1, "Gateway Control Protocol: Version 3", ITU-T, Sep., 2005.

[24] J. Van Dyke et al., "Media Server Control Markup Language (MSCML) and Protocol," RFC 5022, Nov. 2007.

[25] "XML-RPC APIs." [Online] . Available: http://en.wikipedia.org/wiki/XML-RPC,http://www.medooze.com/products/media-mixer-server.aspx, http://ftp.iptel.org/pub/sems/doc/current/Readme.html.

[26] Rosenberg et al, "SIP: Session Initiation Protocol", RFC3261, June 2002.

[27] “H.323.” [Online] . Available: http://www.itu.int/rec/T-REC-H.323.

[28] M. Kulkarni et al; SIP Servlet Specification, version 1.1; JSR 289 Expert Group ; August 2008.

[29] “OEPE”.[Online]. Available: http://www.oracle.com/technetwork/developer-tools/ eclipse/ downloads/ oepe-1116-161753.html.

[30] “X-lite softphones”. [Online]. Available: http://www.counterpath.com/x-lite.html.

[31] “Medooze media server”, [Online]. Available: http://www.medooze.com/.

[32] M. Kulkarni et al; SIP Servlet Specification, version 1.1; JSR 289 Expert Group ; August 2008.

[33] Belqasmi, F. ,Singh, J. , Bani Melhem, S.Y. , Glitho, R.H. ,” SOAP-Based Web Services vs. RESTful Web Services for Multimedia Conferencing Applications: A Case Study”, *IEEE Internet Computing* , Jul-Aug.2012.