vConnect

Distributed Collaboration with Audio/Video Conferencing

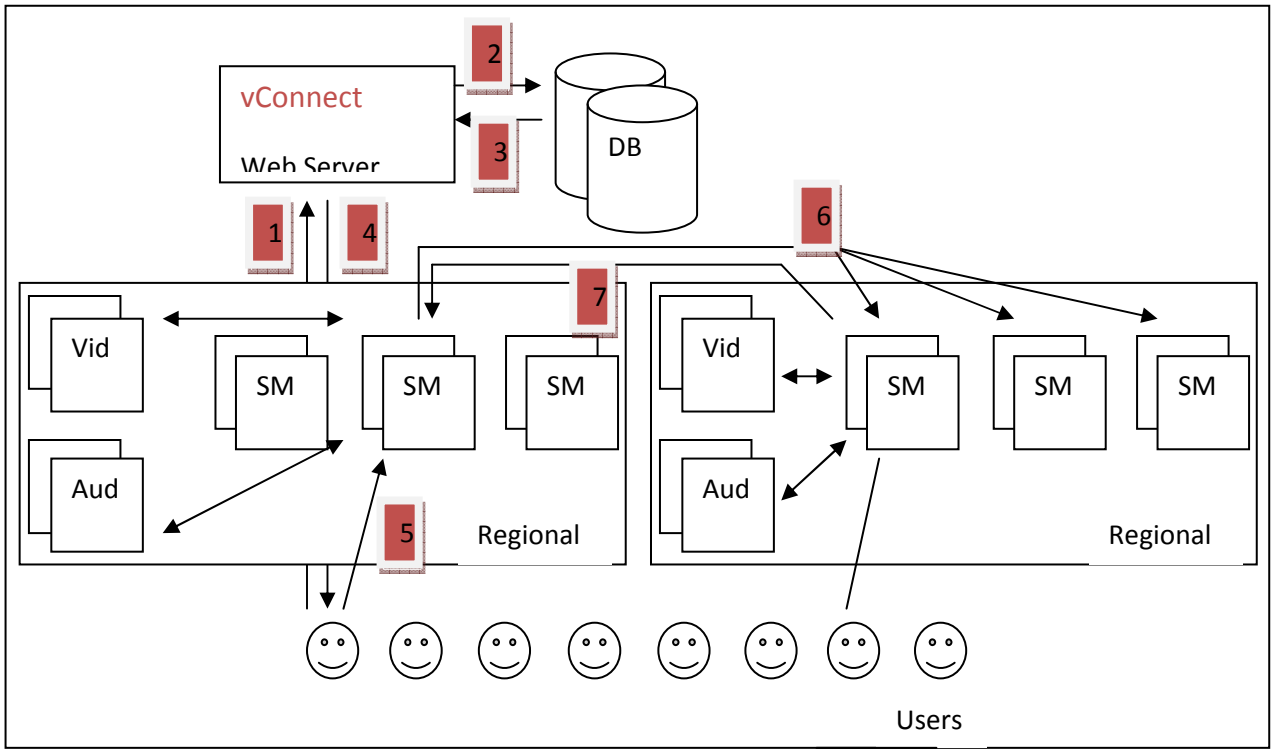
Anoop Jaishankar
Kalpana Chatnani
Nazmi Can Anik
Priyanka Warade

Contents

1. Abstract

We aim to build a distributed system which would enhance communication by means collaborative audio/video conferencing. This application is intended to deliver these services to the users over the internet. It is intended to be a freeware which would be distributed widely, especially amongst the CMU students. It can be used as a collaborative tool for the campus. This collaborative tool does not need a client application at the user end to interact with the network. The interface with which the client interacts is the web browser and hence, it can be accessible from anywhere. This feature makes our product highly portable. We aim to build a highly scalable, resilient and fault-tolerant system which would have almost zero down time. We also aim to provide quality of service to our customers which would enhance user experience. With all these features we plan to get our product in direct competition with the existing popular applications. If time permits we plan to in-corporate whiteboard into our application which can be used by users for programs like distance-learning, corporate conferencing etc. as well as file transfer. With this project we aim to increase the degree of interaction amongst people, thus getting them closer however far they are.

2. High Level Architecture

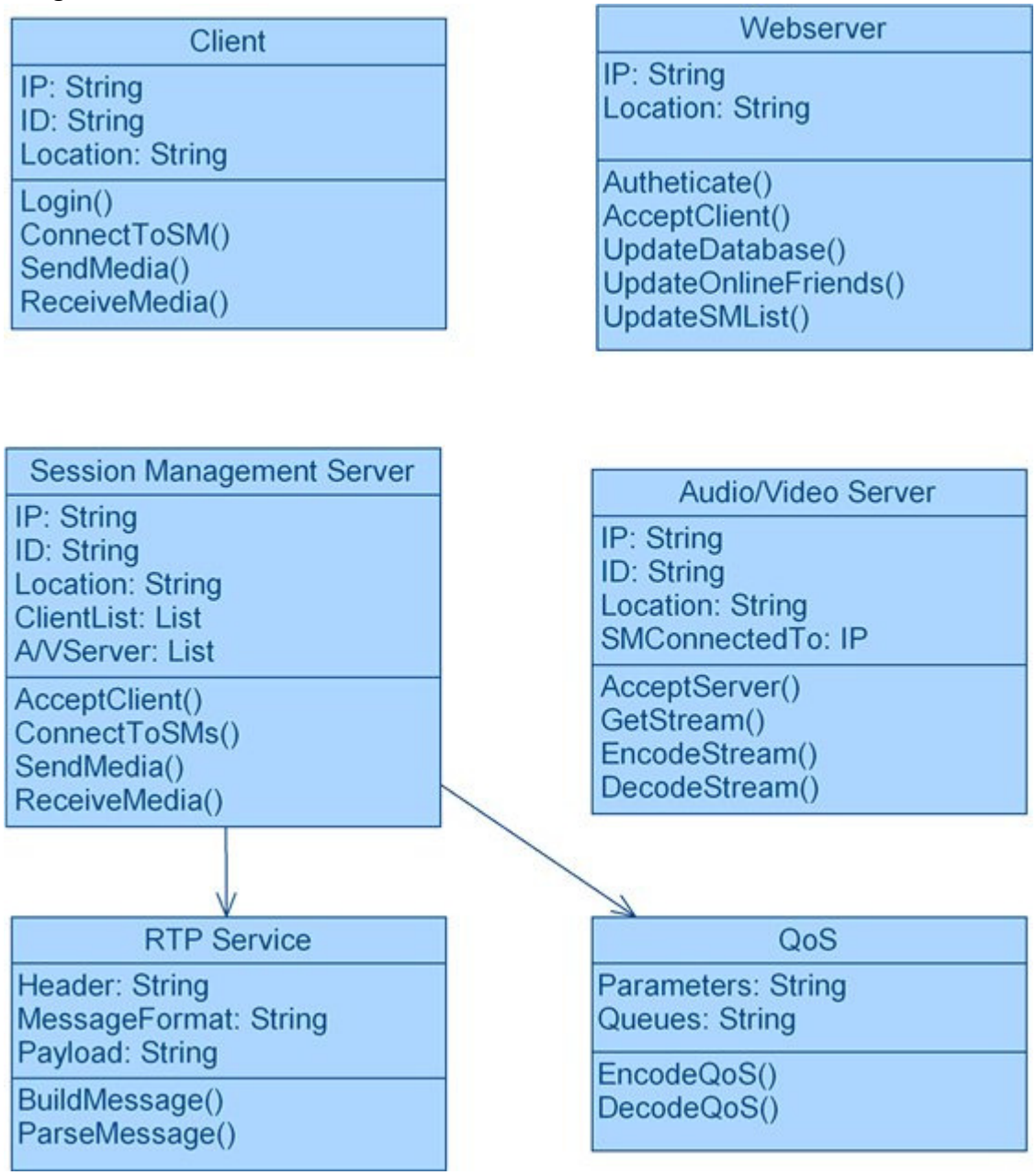


The diagram gives us an overview of the high level architecture design of our system. The system is composed of the following components.

1) Web Server and the Database
2) The Session Management server
3) The Audio server
4) The Video Server
5) The Client

As we can see, each region is made up of the audio/video server and session management servers. In each region, there can be more than one Session Management servers which are directly connected to the client. The number of these servers depends upon the number of clients in each region. At any point of time, the regions can be replicated to service more clients. This makes the system highly scalable. The Web Server and the Database are used for management of the clients connected to the network. Within itself, the web server/database keeps records for each client which is later used for making connections and managing the list of friends.

The class diagram is as shown below:



3. Infrastructural Features

3.1 Scalability

In our system, each region consists of the session management servers, audio servers and video servers. All these interconnect with each other along-with the client and the web server to provide service. At any point of time any number of these regions can be replicated to improve scalability. Such a design helps us to incorporate more number of users to improve the future scope of our project.

3.2 Fault-Tolerance

Our system is designed such that the web server, Session Manager and Audio/Video servers have been replicated to provide fault-tolerance and uninterrupted service. For the web server, this replication is done at the back-up server which has been implemented as a mirror image of the active server. In case the primary server crashes the back-up server takes over via the primary-backup approach. The back-up server takes over using the bootstrapping technique. In this technique, the back-up server information is already present with the client. So, when the web-server crashes, the client connects to it. Failure is detected since the heart beats between the client and the primary web server stop. Hence the client then connects to the backup server. Similarly, the Session Manager also has a back up Session Manager. The bootstrapping technique is used, such that the back up server's information is present with all the modules that interact with the SM (the client, web server, audio/video server). The same way, for the audio/video servers, their back up server's information stays with the SM and the same process is followed.

When the dead primary of web server or SM or audio/video servers, comes back up, its information is updated to the new state and it is made the new back up. The old primary is made the new backup because this keeps away the overhead of informing other components of the new primary. This is based on the assumption that the primary and backup have the same capabilities.

3.3 Quality of Service (QoS)

The session management server handles the quality of service. The session management server generates the RSVP request on behalf of the client based on the quality desired by the client. The session management server sends this request to the destination session management server. If the RSSVP negotiation succeeds the QoS is provided. Otherwise, best effort service is provided by the session manager. We are also planning to provide is DiffServ or Differentiated Services to obtain the desired service that provides different queues for various degrees of service and also different kinds of customers.

3.4 Availability

We have designed our system such that it will have high availability and almost negligible down-time. We have multiple session management servers in each region so that a single session management does not get over-loaded. The session management servers are connected to by the clients randomly and hence, load balancing is achieved. Each region has multiple Session Managers with their back up servers as well. When a client has to connect with the Session Manager, he obtains a list of SMs in his region from the Web Server; he randomly selects an SM from the list and tries to establish connection, if the SM is not overloaded, he accepts the connection, else the client doesn't get a reply and repeats the process with another SM.

3.5 Modularity

The architecture is designed such that all the related functions are combined into a single module. We have separate servers which take care of audio and video and servers which set-up and maintain the connection between the clients. Such a design would help in achieving modularity so that in instances faults, the entire system does not get affected. It provides ease of design implementation and facilitates easy communication.

3.6 Location-Independent Architecture

Our Web-Conferencing tool is accessible via the web browser. The user need not download any client application on his computer to access the service. He just needs a system which is connected to the internet to access this tool. This makes the service highly location-independent and portable.

4. Component Description

The client can establish an audio/video chat session with multiple clients. It involves the following steps:

1) Client authenticates with web server.
2) Web server sends the list of SMs the client can connect to.
3) Web server also sends the list of friends this client has and also their status and region.
4) Client connects to one of the SMs, randomly selecting from the list.
5) The web server updates online friends about this client's status.
6) Client initiates a call setup with SM providing the friends and their regions with whom he wants to talk to.
7) This SM requests the corresponding SMs for a connection with those clients.
8) Connection is established with clients on the other end.
9) Client sends the audio/video stream to the SM.
10) SM forwards this to the audio/video stream for encoding.
11) The audio/video server responds to the SM with the encoded audio/video.
12) The SM sends the stream to the SMs on the other end.
13) The receiving SM will send the stream to the audio/video server in that region.
14) This audio/video server sends the encoded audio/video to the SM.
15) The SM sends this to the client on the other end.

The following sections describe the individual components in detail.

4.1 Web Server and Database

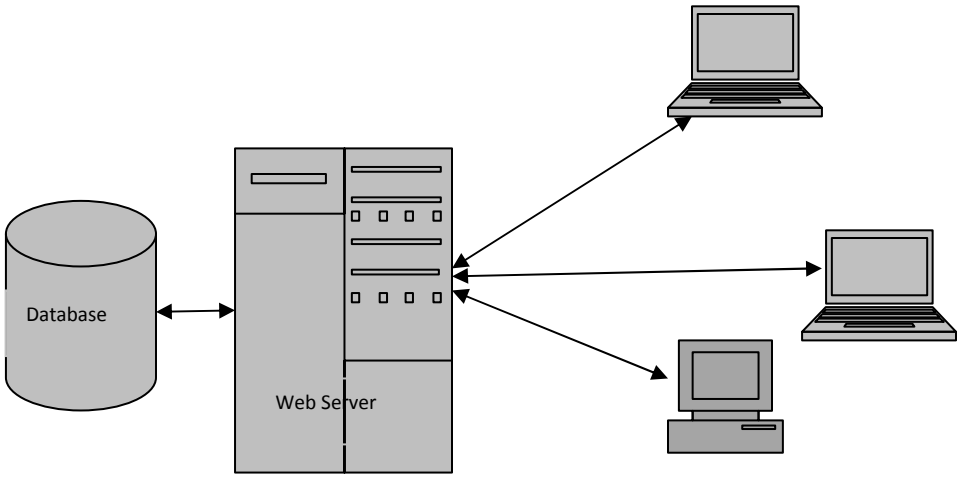


Figure 1 Web Server and Database Interaction

4.1.1 Description:

The web server is the main entity responsible for handling the connection of each of the clients who connect to the network for the first time. It is also responsible for authenticating the client and maintaining an entry for each client and its information within itself. This web server has a back-end database attached to it which keeps the static client information. The web server consults this database whenever it needs any information about the client or needs to verify any particular client's credentials.
The web server stores a dynamic 2-way mapping which consists of information of all the online clients. This mapping contains:

1) The location/region from where the client got connected to the network (this is found out using the whois command on the client IP address).
2) The list of friends of all the clients containing only online friends.
3) The IP address of the client.

The reason we chose the web server to keep such a dynamic table within itself is so that the database does not have to be queried for client location and friend's information or modified each time a friend of any client or the client logs off. In such an event, the web server would automatically delete the client information from its mapping and inform all its friends about the client's new status.

4.1.2 Functionalities:

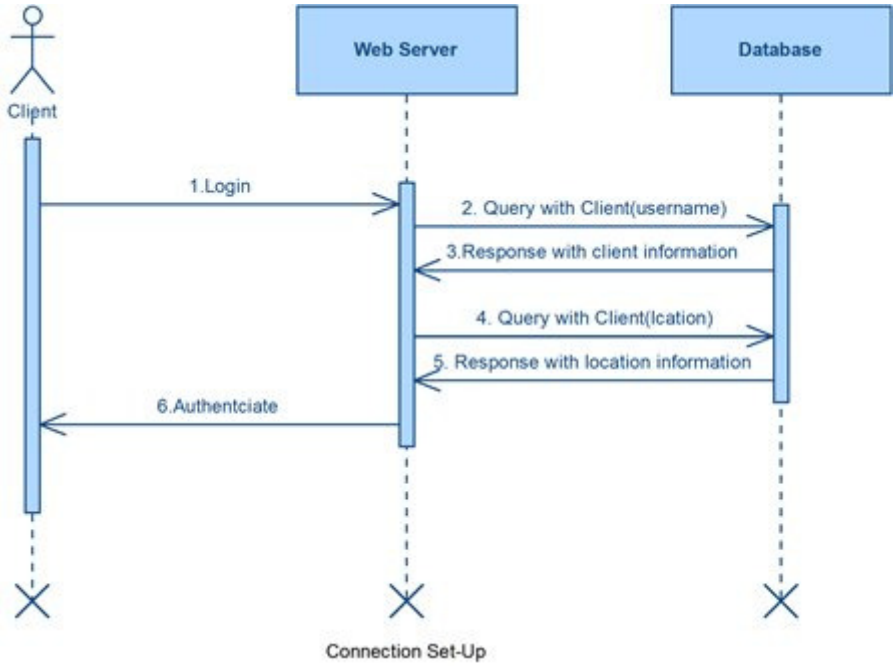
1) Get Login request message from the client.
2) Send authentication message to client.
3) Query the database to get client information.
4) Update online clients about his friend's status.
5) Add a new client to the database.
6) Delete a client from the database.
7) Store the dynamically mapped table which stores client information like client IP, location information, and online list of friends.
8) Update the back-up simultaneously, so it can easily take over in case primary dies.

4.1.3 Interfaces:

Boot-up:

1) Initializes its dynamically mapped table.
2) Connect to the database.
3) Start listening for connection (HTTP) requests.

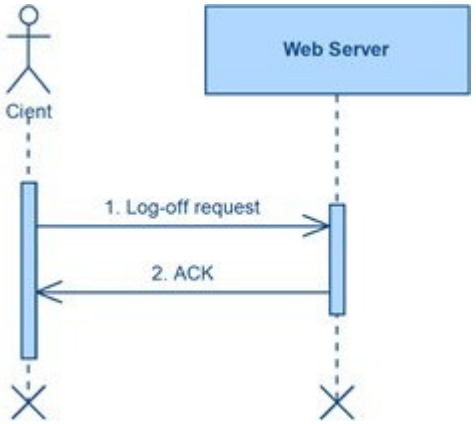
Connection Set-Up:



Connection Set-Up

1) Gets *Client (connection)* request from client.
2) Web server receives the request.
3) The web server queries *Client (username, region)* to the database to get client information.
4) The database responses with two messages: 1) *Client (username, password, friends)* 2) *Region (list of session management servers, IP addresses of each session management servers)* to the web server who then matches the client's credentials and makes a list of all his online friends using his dynamic table.
5) The web server makes an entry for the online client in its dynamically mapped table which contains the client IP address, location and all online friends of each client.
6) The web server authenticates the client and sends a response *Client* (*connected successfully)* to the client.
7) He also sends to the client the list of all session management servers along-with their IPs *Region (list of session management servers, IP addresses of each session management servers).*

Connection Tear-Down:



1) When a client wishes to log-off, it will send a *Client (connection cancel)* request to the web-server.
2) The web server on getting such a request will remove the client entry from its table.
3) It will inform the other online clients who are friends with this client about its offline status by sending *Client (client[i] offline)* message to them.
4) It will then send an acknowledgment back to the client *Client (cancel ACK).*
5) This terminates the connection.

If a client disconnects from the network abruptly, the session management server connected to that client will inform the web server of the incident and the web server will remove the client information from its table. It will also inform all the friends of the client's offline status.

Database

4.1.4 Description:

The Database unit is in direct contact with the web server. It issues responses to queries generated by the web server. The database is responsible for holding the client information like username, password, profile information, list of friends, etc. It also holds information of every region like region name, list of session management servers, IP addresses of the session management servers in each region.

The database information is modified only when any client joins the network for the first time or deletes his account. This command is issued by the web server. Our aim is to keep the database isolated from rest of the network and store only static information. As a result, the accesses to database will be in-frequent which otherwise would have increased the overall response time.

4.1.5 Functionalities:

1) Store client information consisting of client username, password, profile information, list of all his friends.
2) Store region information consisting of list of session managers and IP addresses of each session managers.
3) Accept queries from the Web Server.
4) Give back responses to the Web Server for each query.

4.1.6 Interfaces:

The following sequence of operations takes place for the following events:
Client Query:

1) Receive query from the Web Server *Client (username, region)*.
2) Database looks up the client information database and the region information and sends the response to the query. He also looks up the region information database and sends region information in that reply. The response is generated in two messages: 1) *Client (username, password, friends)* 2) *Region (list of session management servers, IP addresses of each session management servers)*.

Client Update:

1) Receive client update message from the Web Server *Client (username, update_message)*.
2) Database looks up the client information database and modifies the given data for the client in its database.

Client Insert:

1) Receive client insert message from the Web Server *Client (username, password, profile_information)*.
2) Database inserts the client information into its database for the new client.

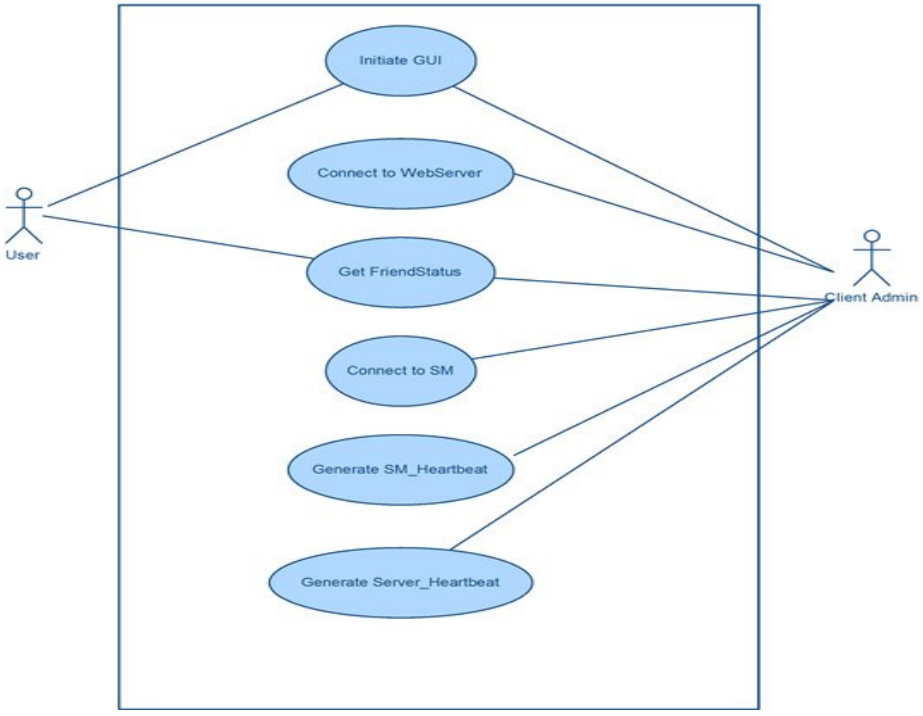
4.2 Client

4.2.1 Description:

The client interacts with the web server and the session manager. The client basically is a web browser which opens an applet. This applet acts as an interface for the user to interact with the web server. The client has a rich GUI which enables the user to see which of his friends are online. He can choose who and how many he wishes to speak with. The client allows a user to have a text chat session, audio conference, video conference. The client is a thick one, storing some state such as the list of session managers around him, his friends list along with who is online and their location. The state has been stored with the client so that the whole burden does not fall on the web server or the session manager. A user can connect to multiple users and establish audio/video multicasting conference.

4.2.2 Functionalities:

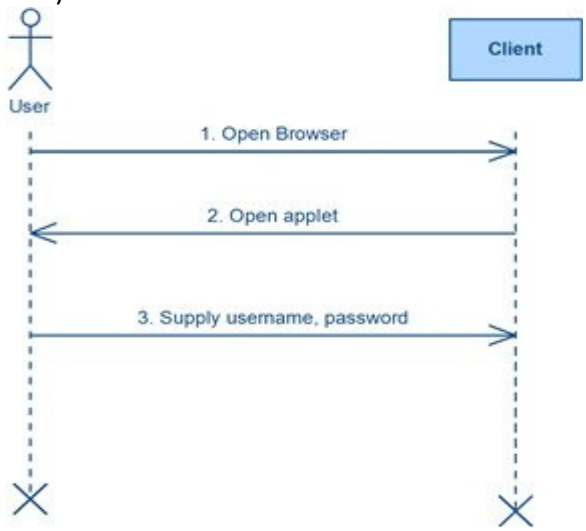
1) Provide a GUI interface to a user where he can log in to vConnect
2) Establish a connection with the Web Server and authenticate the user.
3) Provide the user with a list of his friends, their status and location.
4) Connect with the Session Manager.
5) Periodically inform the Session Manager that user is alive.
6) Periodically inform the Web Server that user is alive.
7) User can connect to multiple clients using the multicast feature.



4.2.3 Interfaces:

Provide GUI interface to a user where he can log in to vConnect (showApplet):

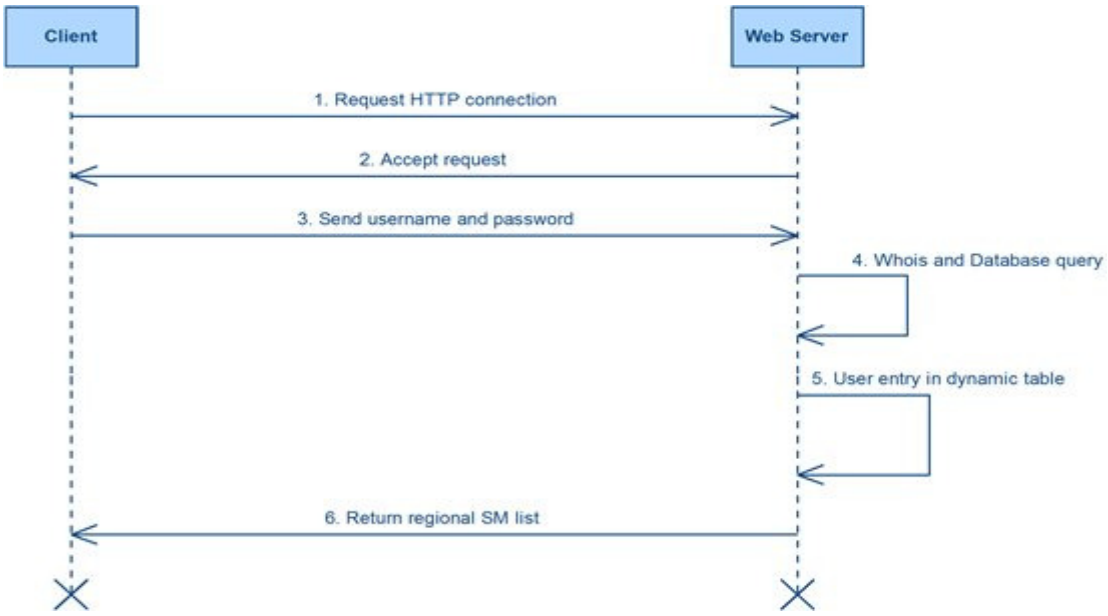
1) User opens the web browser and reaches http://www.vconnectcmu.com
2) Client opens an applet for the user where he can enter his username and password.
3) Client stores this to send to the web server.



Sequence Diagram for User Connecting to Client

Establish connection with the Web Server Client (Connection)

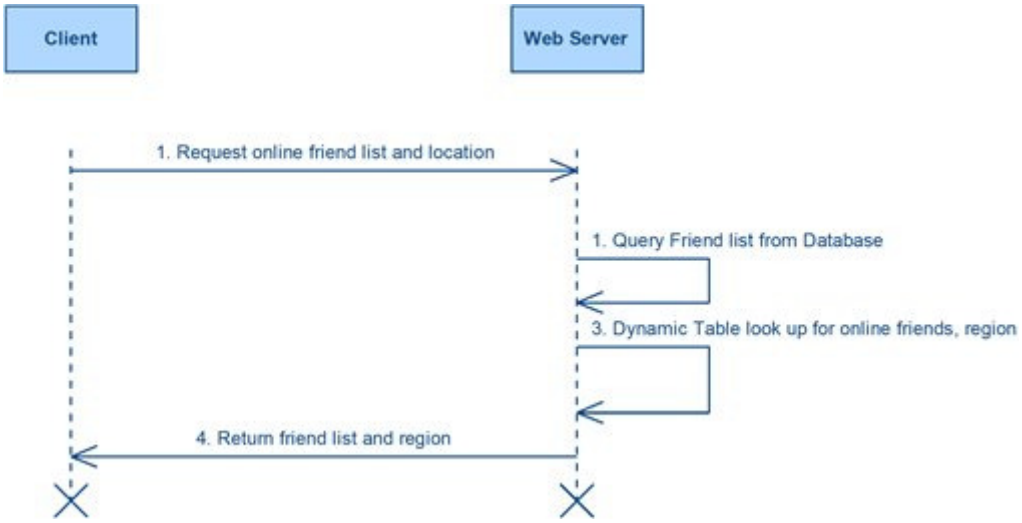
1) Client requests for an HTTP connection with the Web Server.
2) Web Server accepts the request.
3) Client establishes an HTTP connection with the Web Server.
4) Client sends username and password to the client.
5) Web Server performs a whois query to get the client location.
6) Web Server queries the database for a match.
7) In case of a valid combination of the username and password, user gets authenticated.
8) Web server makes an entry in the dynamic table to store user's location.
9) Web Server returns a list of Session Managers in the region.



Sequence Diagram for Client Connection to Web Server

Provide the user with a list of his friends, their status and location FriendStatus()

1) With the user being authenticated, the client requests for the following things from the Web Server: List of friends online, their location.
2) The Web Server on the basis of the user name looks up the database to get his list of friends.
3) On getting the list of friends, the web server looks up the dynamic table to see who all are online and returns their region to the client.



Sequence Diagram to get Friend Status

Connect with the Session Manager Connect_SM()

1) The Client looks at the list of Session Manager (SM) sent by the Web Server based on his location.
2) It chooses an SM randomly thus providing load balancing.

3) It contacts the chosen SM and sends a connection request.
4) If the SM is down or heavily loaded, it denies the request, else SM accepts the request and acknowledges it.
5) In case a connection is denied, the Client repeats the steps 2, 3 and 4 till he connects with an SM.

Sequence Diagram for Client to connect to Session Manager

Periodically inform Session Manager and WebServer that he is alive HeartBeat_SM()

1) Client sends KeepAlive message to the Session manager periodically.
2) Client sends KeepAlive message to the Web Server periodically.

4.3 Audio and Video Servers

4.3.1 Description:

We have separated the audio and video processes on different servers so that even if video servers fail, the audio streaming could continue to provide communication between the clients. The main priority of the conferencing toll is audio conferencing. In addition to this, separating the functions provide modularity, since different processes are done on audio and video streams.
Presentation quality of a media stream depends mainly on the compressing scheme used, the processing capability of the playback system, and the bandwidth that is available.
Figure 1 below summarizes the basic steps taking in media processing as “Input Stage”, “Process Stage”, and “Output Stage”. Firstly, the contacting Session Management Server sends the stream to the corresponding media server with certain specifications for processing. After the processing is handled at the media server for the input parameters, it is sent back to the Session Management Server.

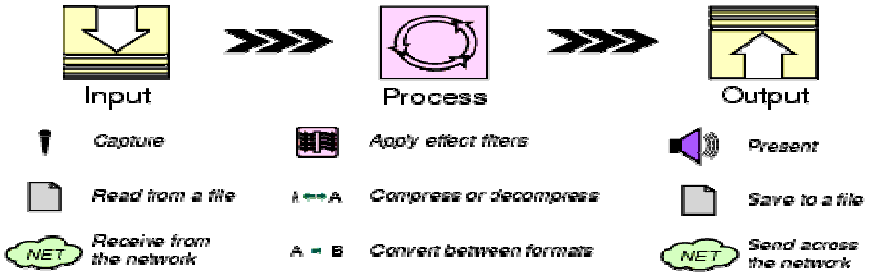


Figure 1 – The Basic Steps in Media Processing

4.3.2 Functionalities:

1) Decoding: In this step, each track is decompressed at the server.
2) Processing: In this step, some effect algorithms are applied to the tracks.
3) Mixing: In this step, several tracks are interleaved into a single track for audio output at the client side.
4) Encoding: In this step, each track is compressed at the server.

To be able to apply Quality of Service to vConnect Audio/Video Conferencing tool, the controller interface provided within the JMF will be very helpful.

The usage of TrackControl interface enables the controlling what processing a Processor object performs on a particular track. The Effect, Codec, or Renderer plug-ins to be used by the processor can be selected using TrackControl methods.

JMF defines several codec controls to enable control over encoders and decoders:

1) *BitRateControl* : Used to control the encoding bit rate

2) *FrameProcessingControl*: Enables the specification of frame processing parameters that allow the codec to perform minimal processing when it is falling behind on processing the input stream
3) *FrameRateControl*: Enables the modification of the frame rate and thus bandwidth usage and quality of the stream.
4) *Quality Control*: Enables the specification of a preference in the trade-off between quality and CPU usage in the processing performed by codec.
5) *SilenceSupressionControl*: Enables specification of silence suppression parameters for audio codecs, to be able to alter the bandwidth usage, and thus the quality of the audio stream.

4.3.3 Interfaces:

1) Receive media from SM Server (*ReceiveStream)*
2) Receive QoS Specifications from the SM Server from the Controller Interface:
   - Controlling Quality of Service parameters (*QualityControl*)
   - Controlling the encoding bit rate (*BitRateControl)*
   - Controlling of Silence Supression (*SilenceSupressionControl*)
   - Controlling H.263 video-codec parameters (*H263Control*)
   - Controlling of frame processing parameters *(FrameProcessingControl)*
3) Send the processed media to SM Server (*SendStream*)

4.3.4 Sample Scenario:

1) Locate the incoming stream and get the mediaLocation, which corresponds to the Session Management Server
2) Call the PlugInManager.getPlugInList method to see what plug-ins are available
3) Create a processor to deal with the incoming stream
4) processor = Manager.createProcessor(mediaLocation);
5) Add a controller listener to get any controller inputs from the Session Management Server
6) processor.addControllerListener(this);
7) Respond to controller events by implementing the function below:
8) controllerUpdate(ControllerEvent event)
9) Call processor.getTrackControls to get TrackControl for the track in the media stream
10) Call the TrackControl setCodecChain to specify the plug-ins to be used for each track
11) Get the codec controls by calling TrackControl getControls method. This returns codec controls such as *H263Control*, *QualityControl* and apply the control given by the SM Server
12) Get the output DataSource from the Processor by calling getDataOutput
13) Construct a DataSink to send the processed stream back to the SM Server by calling Manager.createDataSink and pass in the output DataSource and a Media Locator that specifies the SM Server as the location

4.4 Audio Server

4.4.1 Description:

The audio servers are responsible for receiving input audio streams from the Session Management Servers, and processing these streams according to Quality-of-Service specifications input by the contacting SM Server itself. After processing, the audio server is responsible for sending back the processed audio stream back to the SM Server for delivery to the client.

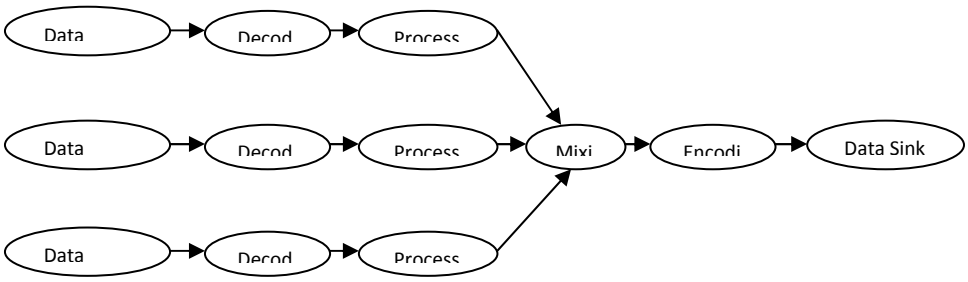


Figure 2 – Figure Showing the Steps in Functions of the Receiving Audio Server

In the receiving side the several audio streams are mixed together. In the transmitter side, no mixing is done; the processed audio stream is directly encoded and sent to the data sink.

4.4.2 Functionalities:

1) Decoding: In this step, decompression is applied on each track to be able to process the streams.
2) Processing: In this step, some effect algorithms are applied to the tracks according to the control parameters input by the SM Server. Filtering out high frequency components would be suitable, since human vocal cords can generate sound within the frequency range 300 Hz – 3.4 KHz. So, instead of using a sampling rate of 44 KHz, as in the case of music players, 8 KHz sampling rate would be sufficient. This would result in using less bandwidth for transmission of the audio stream, thus providing lower-quality audio streaming for QoS purposes.
3) Mixing: In this step, several tracks are interleaved into a single track, so that there is one audio stream to be transmitted to the target client.
4) Encoding: In this step, compression is applied on each track considering the bandwidth and QoS specifications.

4.4.3 Interfaces:

1) Receive media from SM Server (*ReceiveStream)*
2) Receive QoS Specifications from the SM Server from the Controller Interface:
   - Controlling Quality of Service parameters (*QualityControl*)
   - Controllinging the encoding bit rate (*BitRateControl)*
   - Controlling of Silence Supression (*SilenceSupressionControl*)

3) Send the processed media to SM Server (*SendStream*)

4.5 Video Server

4.5.1 Description:

The video servers are responsible for receiving input video streams from the Session Management Servers, and processing these streams according to Quality-of-Service specifications input by the contacting SM Server itself. After processing, the video server is responsible for sending back the processed video stream back to the SM Server for delivery to the client.

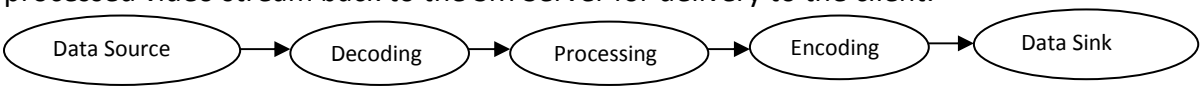


Figure 3 – Figure Showing the Steps in Functions of Video Server

4.5.2 Functionalities:

1) Decoding: In this step, decompression is applied on each track to be able to process the streams.
2) Processing: In this step, some effect algorithms are applied to the tracks according to the control parameters input by the SM Server. Filtering out high frequency components would result in using less bandwidth for transmission of the video stream, thus providing lower-quality video streaming for QoS purposes. The sampling rate could be altered for different QoS levels. For high quality video streams, noise reduction, detail enhancement and upscaling algorithms may be used.
3) Encoding: In this step, compression is applied on each track considering the bandwidth and QoS specifications.

4.5.3 Interfaces:

1) Receive media from SM Server (*ReceiveStream)*
2) Receive QoS Specifications from the SM Server from the Controller Interface:
    a. Controlling H.263 video-codec parameters (*H263Control*)
    b. Controlling Quality of Service parameters (*QualityControl*)
    c. Controllinging the encoding bit rate (*BitRateControl)*
3) Send the processed media to SM Server (*SendStream*)

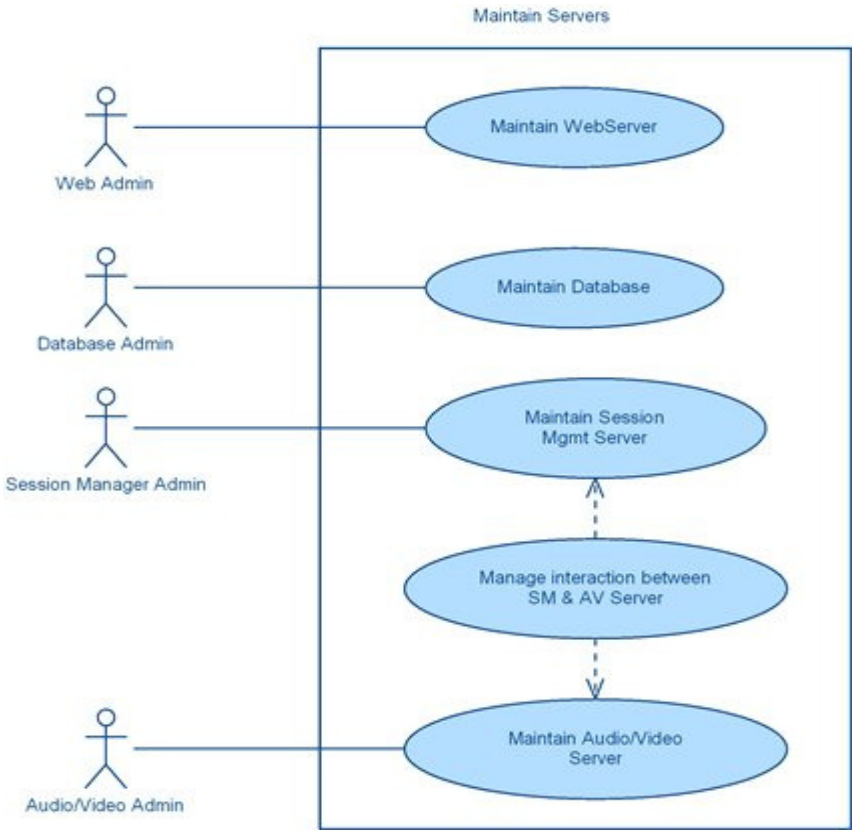
4.6 Session Manager

4.6.1 Description:

Session Manager will establish connection with the clients and provide the Audio/Video streaming functionality to the system. The SM will be responsible of interacting with clients, Audio/Video servers and other SMs. Clients interact with SMs and not directly with the Audio/Video servers as we can keep the processing at the client to the minimum. Another design feature to consider here is the separation of Audio and Video servers as this will enable us to keep both the streams separate. Hence we can deliver the audio irrespective of the video breakdown.

4.6.2 Functionalities:

1) Listen to Client connections.
2) Accept connection from a Client.
3) Call setup with another SM on the network. This is done by searching the list of SMs provided by the client.
4) Negotiate a RSVP connection with the other server.
5) Accept media stream from the client.
6) Forward the stream to the Audio/Video server for encoding.
7) Receive the response from Audio/Video server.
8) Perform QoS capabilities through Differentiated service. This can be done by having multiple data queues for different media. Also through Gold/Silver/Bronze kind of service.
9) Different formats can be: Video – H.261(Low), H.263(Med).
   Audio: MuLaw(Low), ADPCM(Med), GSM(Low), G723.1(Low).



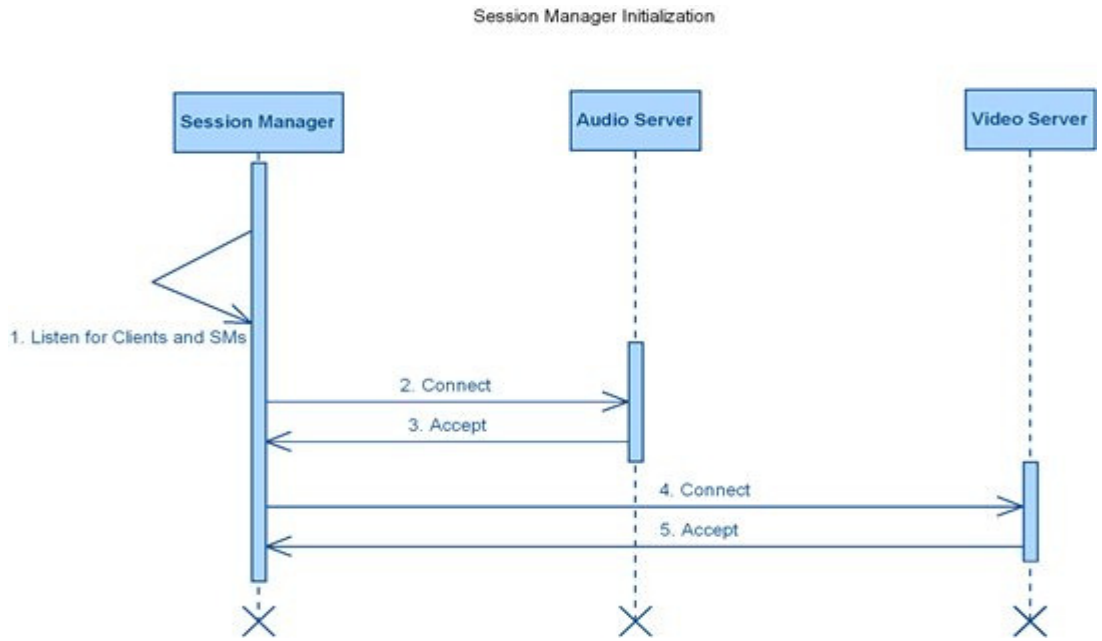
4.6.3 Interfaces:

Boot up:

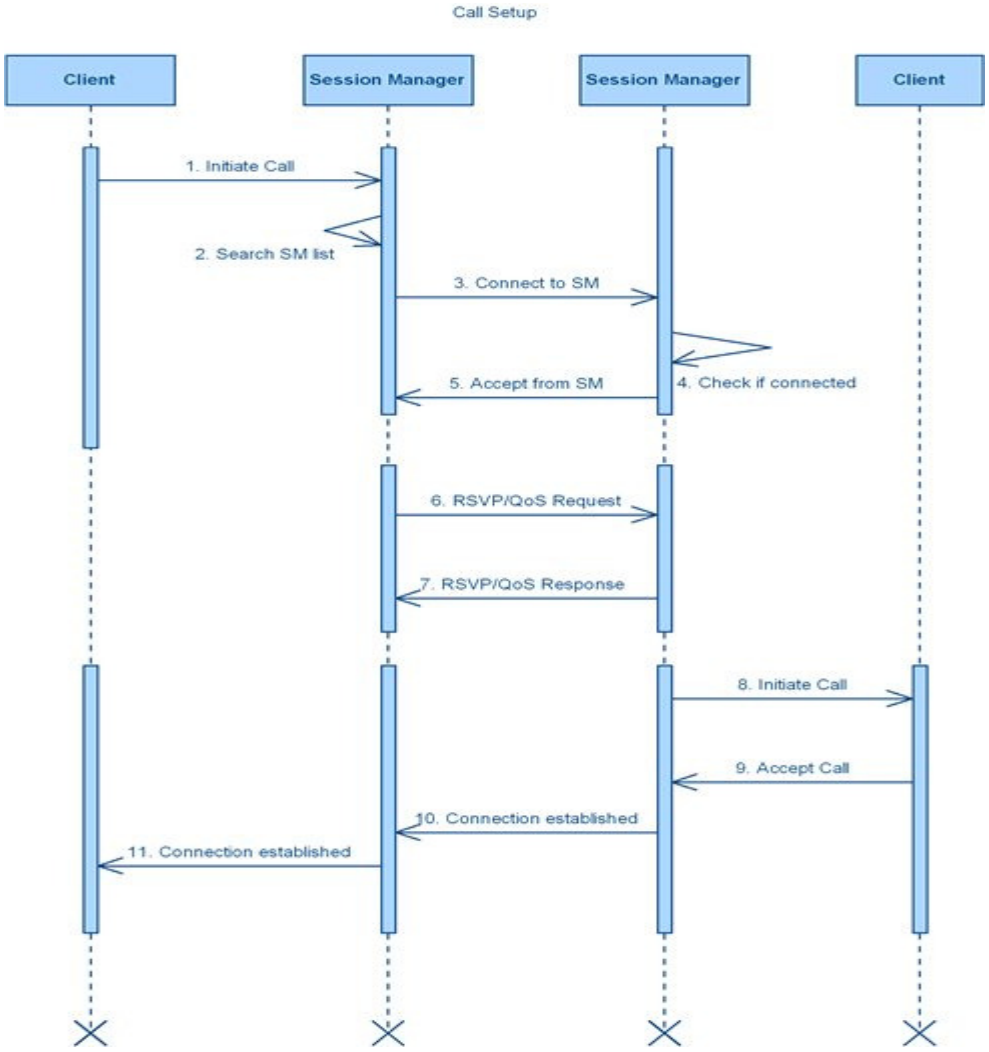
1) Connect to Audio/Video Server *(ConnectAVServer)*: Establish connection with Audio and Video server.
2) Listen for connections from clients.
3) Listen for connections from other SMs looking for a connection for session initialization.

Connection establishment with Client:

1) Accept connection *(ClientRequest)*: Receives a connection request from Client and store the information received in its datastructure.
2) Send response *(ClientAccept)*: Sends an accept message to the client.


Session Manager Initialization
Session Manager
Audio Server
Video Server
1. Listen for Clients and SMs
2. Connect
3. Accept
4. Connect
5. Accept

Call setup:


Call Setup
Client
Session Manager
Session Manager
Client
1. Initiate Call
2. Search SM list
3. Connect to SM
4. Check if connected
5. Accept from SM
6. RSVP/QoS Request
7. RSVP/QoS Response
8. Initiate Call
9. Accept Call
10. Connection established
11. Connection established

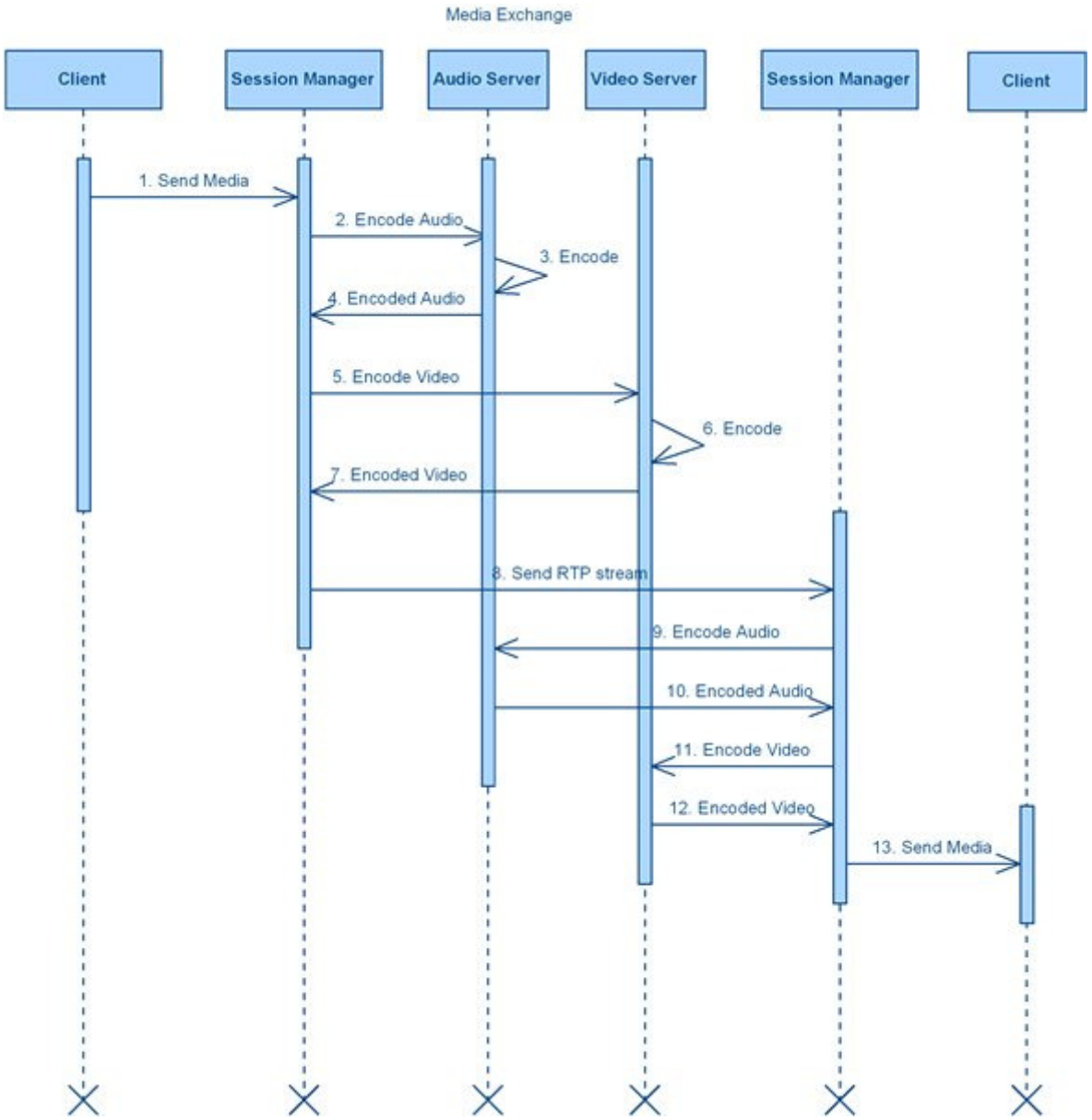
1) Receive Call Request *(InitiateCall)*: Receive the region the destination client can be connected from.
2) Search for SMs with the destination client *(SearchSMs)*: Search the list of SMs that can be present in that region from its database.
3) Send connect messages to list of SMs *(ConnectToSM)*: Send to all the SMs in that region.
4) Receive response from SM *(ReceiveSMResponse)*: If the client is connected to a particular SM, then that SM replies with a positive response.
5) RSVP connection establishment *(RSVP Request/Response)*: Send an RSVP request with the specified parameters. Receive a positive response if accepted, else RSVP fails.
6) Setup QoS parameters *(SetupQoS)*: Set up DiffServ parameters for QoS control.

Receive media from the Client:

1) Receive media stream *(ReceiveClientStream)*: Receives a media stream from the client.
2) Send audio stream to Audio server *(SendToAudioServer)*: Send audio stream for encoding to the Audio servers.
3) Send video stream to Video server *(SendToVideoServer)*: Send audio stream for encoding to the Video servers.
4) Receive audio stream from Audio server *(RecvFromAudioServer)*: Receive audio stream after encoding from the Audio servers.
5) Receive video stream from Video server *(RecvFromVideoServer)*: Receive video stream after encoding from the Video servers.
6) Send media stream *(SendStream)*: Send the media stream to the SM on the other end through RTP.

Send media to the Client:

1) Receive media stream *(ReceiveStream)*: Receives a media stream from the RTP connection.
2) Send audio stream to Audio server *(SendToAudioServer)*: Send audio stream for decoding to the Audio servers.
3) Send video stream to Video server *(SendToVideoServer)*: Send audio stream for decoding to the Video servers.
4) Receive audio stream from Audio server *(RecvFromAudioServer)*: Receive audio stream after decoding from the Audio servers.
5) Receive video stream from Video server *(RecvFromVideoServer)*: Receive video stream after decoding from the Video servers.
6) Send media stream *(SendStreamToClient)*: Send the media stream to the client.



Connection Management:

1) Send Keep Alive messages *(SendKeepAlive)*: Send Keep Alive messages to Clients, Audio server, Video server and other SMs that it is connected to.
2) Receive Keep Alive messages *(ReceiveKeepAlive)*: Receive Keep Alive messages from Clients, Audio server, Video server and other SMs that it is connected to.
3) Tear down the connection if Keep Alive not received.

Interaction with Backup Server:

1) Send messages to back up server *(SendMessageToBackup)*: Send messages to backup server.
2) Receive messages from back up server *(ReceiveMessageFromBackup)*: Receive messages from backup server.
3) Interact with backup in choosing the server that interacts with other entities *(BackupAlgorithm)*: Implement the primary backup algorithm to provide replication. That particular SM will interact with other clients and SMs.

5. Project Schedule

Dates	Weeks	Web Server	Client	Audio/Video	Session Management
29-Feb	Week 1	Initialization Connection with Client	Initialization Connection with Web Server Connection with SM	Initialization Connection with SM	Initialization Connection with Client Connection with Audio/Video
7-Mar	Week 2	Connection with Database	Authentication Exchange Text Messages with SM	Exchange Audio Stream with SM Exchange Video Stream with SM	Exchange Text Messages Exchange Audio Stream between Client and A/V Exchange Video Stream between Client and A/V
14-Mar	Week 3	Authentication	Exchange Audio Stream with SM	Process Audio Stream	Exchange Audio Stream with SM
21-Mar	Week 4	Query and Update Data Storages Integration Mid-Term Demo	Integration Mid-Term Demo	Integration Mid-Term Demo	Exchange Video Stream with SM Integration Mid-Term Demo
28-Mar	Week 5		Exchange Video Stream with SM	Process Video Stream	QoS Parameter Exchange with A/V and SM
4-Apr	Week 6	Primary-Backup Approach		Primary-Backup Approach	Primary-Backup Approach
11-Apr	Week 7				
18-Apr	Week 8	Testing	Testing	Testing	Testing
25-Apr	Week 9	Documentation	Documentation	Documentation	Documentation
2-May	Week 10	Final Demo	Final Demo	Final Demo	Final Demo

6. Demo Sequences

1) Initialization of modules with setting up of internal parameters and basic interfaces.
2) Show client connection with Web server.
3) Show client connection with the SM.
4) Show SM connecting with other SMs.
5) Authenticate client with Web server. Web server interacts with database and verifies the authentication.
6) Exchange text messages between SM and client. Send these text messages to the SM on the other end. Send the text messages with the client on the other end.
7) End to end audio exchange between two clients:
    a. Generate audio stream from the client to SM.
    b. Send this stream to the Audio server.
    c. Audio server processes the received audio stream. Performs encoding of streams according to QoS parameters.
    d. SM sends it across to the SM on the other end.
8) End to end video exchange between two clients:
    a. Generate video stream from the client to SM.
    b. Send this stream to the Video server.
    c. Video server processes the received video stream. Performs encoding of streams according to QoS parameters.
    d. SM sends it across to the SM on the other end.
9) Demonstrate connection with multiple clients.
10) Demonstrate primary backup approach of web server, audio/video servers and Session Management server by showing failure of the primary, take over by the back up. When the primary comes back up, the back up continues to act as the primary, making the old primary as the new back-up.
11) Demonstrate load balancing of modules, with redirection of connections to other SMs or web servers.

7. Implementation

1) Development: J2EE and JMF API, Apache Tomcat
2) Database: MySQL, JDBC Database Access
3) Modelling: UML Diagramer
4) Languages: Java, HTML, JSP
5) RAD Tool: NetBeans

8. Responsibilities

1) Anoop Jaishankar(President): Session Management Server, connection management with Audio/Video Servers, other SMs, QoS, load balancing and primary backup approach
2) Nazmi Can Anik: Audio/Video servers, connection with Session Managers, QoS, load balancing and primary backup approach
3) Kalpana Chatnani: Client GUI, connection with SM, connection with Web Server, Audio/Video server, Audio/Video Servers
4) Priyanka Warade: Web Server interaction with client, authentication, Database module, web server and database interaction, maintain status of users

9. References

1) RFC 1889 – RTP A Transport Protocol for Real Time Applications
2) Service-Oriented Architecture for Building a Scalable Videoconferencing System; Ahmet Uyar, Wenjun Wu, Hasan Bulut, Geoffrey Fox,
3) Java Media Framework – API Guide
4) The Server Array: A Scalable Video Server Architecture; Christoph Bernhardt, Ernst Biersack, March 1996
5) Traffic Identification and Overlay Measurement of Skype; Yanfeng Yu,Dadi Liu, Jian Li, Changxiang Shen; 2006
6) Scalable Service Oriented Architecture For Audio/Video Conferencing; Ahmet Uyar; May 2005
7) Java2 - Complete Reference, Herb Schildt
8) J2EE 1.4 Tutorial; Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock; December 7, 2005
9) A QoS Guaranteed Multimedia Conference Service over Packet Based Network; Sang Gil Kim , Yongmin Choi, Yong Sun Ksm; October 1998
10) Audio Video Conferencing in Distributed Brokering Systems; Ahmet Uyar, Shrideep Pallickara, Geoffrey Fox