2003

An Integrated Videoconferencing System for Heterogeneous Multimedia Collaboration

Ahmet Uyar
Syracuse University, Department of Electrical Engineering and Computer Science ; Indiana University, Community Grid Labs, auyar@syr.edu

Wenjun Wu
Indiana University, Community Grid Labs

Hasan Bulut
Indiana University

Geoffrey C. Fox
Indiana University, Community Grid Labs

An Integrated Videoconferencing System for Heterogeneous Multimedia Collaboration

Ahmet Uyar[1,2], Wenjun Wu[2], Hasan Bulut[2], Geoffrey Fox[2]
[1]Department of Electrical Engineering and Computer Science, Syracuse University
[2]Community Grids Lab, Indiana University
auyar@syr.edu, wewu@indiana.edu, hbulut@indiana.edu, gcf@indiana.edu

Abstract

We have developed an integrated conferencing system, Global Multimedia Collaboration System, which enables heterogeneous multimedia clients to join the same real-time sessions. Our system provides support for a variety of protocols and applications, including H.323 clients, SIP clients and Access Grid rooms. In this paper, we would like to show the features and design principles of our conferencing server which facilitates audio and video communications among participating clients in a real-time conference.

Keywords: videoconferencing, XGSP, Collaborative Systems and Applications, Distributed Multimedia Systems.

1. Introduction

Today Internet provides a very convenient and cost effective medium for audio and video communications. Highly sophisticated audio and video codecs have been designed to transfer audio and video content efficiently and standards have been developed to initiate and manage real-time multimedia sessions. However, these standards come from different communities and they tend to solve the same problem in quite different ways, partly because of some historical reasons and partly because of differences on objectives. Currently there are three major video conferencing systems, H.323[1], SIP[2] and Access Grid[3]. Every community has its own set of protocols and products. A user of one system can not talk to a user of another system, although both have the means to send/receive audio and video. They are like three islands in Internet without a bridge among them.

We have designed an architecture and implemented the prototype system to integrate all these different communities in the same real-time session in an easy-to-use fashion. Since these systems are not identical in functionality and their way of solving the same problem is quite different, it is not easy to translate messages from one protocol to another. Therefore, we have developed a more general XML based session management protocol (XGSP) which covers a wide range of collaboration functions and interacts easily with these systems.

Although there has been some work done to interoperate these systems, they exclusively focused on the interactions between two communities. [4,5] presents a solution to interoperate SIP and H.323 systems. [6] is another videoconferencing system which targets the interoperability between H.323 and Access Grid communities. Our approach is more general and provides a general session management protocol to accommodate more protocols and applications with an easy to use web interface. It is even not limited to multimedia sessions, it can be extended to be used for any kind of real time application from simple chat to online gaming.

In this paper after giving a brief overview of our Global-MMCS, we will provide a detailed description of our audio and video conferencing solutions and their design principals, and discuss the performance results.

2. Global-MMCS Overview

Global-MMCS[7,8,9] is an integrated video conferencing solution which enables heterogeneous clients to join the same real-time multimedia sessions. It provides a flexible architecture to support even more standards and applications. There are five main components of this architecture (Figure 1); NaradaBrokering(NB) servers, XGSP Session Server, Gateways, Media Server, and Web server. NB is a distributed publish/subscribe messaging system which delivers all messages. XGSP Session Server manages real-time sessions, namely starts/stops/modifies them. It receives messages from gateways and the web server, and performs appropriate actions on the media server. Gateways receive protocol specific messages from different clients, and pass them to the XGSP session server after converting to XGSP messages. There should be a dedicated gateway for each supported protocol. Currently we have H.323 and SIP gateways. Media Server facilitates the audio and video communications among participants in a meeting. We will cover the details of the media server in following sections.

The web server provides an easy-to-use web interface for users to join multimedia sessions and for administrators to perform administrative tasks. In addition, users can start some audio and video clients

through these web pages such as VIC, RAT and Real Player.

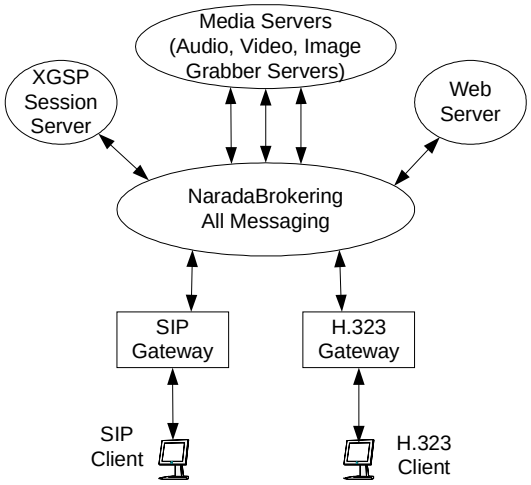

Figure 1. The architecture of Global Multimedia Collaboration System

3. Media Server

Media server facilitates audio and video transfer among participants in real-time multimedia sessions. It receives RTP streams from source clients and redistribute it to interested parties, sometimes by replicating the received media, sometimes by mixing, and sometimes by transcoding. In addition, it provides an XML based message interface through which audio and video sessions are managed by the session server. It has three main components; audio server, video server and image grabber server. Each of these three components are independent of one another and they run in different machines. This makes our system more scalable and flexible.

Both Audio and Video Servers implement a meeting management concept and an interface to manage these meetings over the network. Session Server manages these meetings by sending/receiving messages back and forth over the network. They provide functions to create/delete meetings, and to add/remove participants. In addition, they also provide a method to give the list of audio and video streams in meetings. These are all XML messages and defined in an XML schema. Both Audio and Video servers support many meetings at a time and each meeting can have any number of participants.

We have implemented the media server in Java using Java Media Framework[10]. It provides a flexible and extendible architecture. Since Java is relatively slower than C/C++ for multimedia processing, all CPU intensive encoders and decoders are implemented in native C/C++ code. This makes JMF a good candidate for developing multimedia applications, since it provides the performance of the native code and the flexibility of Java platform. By default it supports the most commonly used codecs such as H.261, H.263, JPEG video codecs and G.711, G.723.1, GSM audio codecs. All these codecs come with both encoders and decoders except H.261. It only has the decoder. This was an important missing part for us and we have developed an encoder for H.261.

Today most of the conferencing servers are based on H.323. There are hardware and software implementations. Polycom and Radvision are two companies which provide hardware based conferencing servers. IBM Lotus Sametime, VCON Media Exchange Manager, and OpenMCU H.323 conference server are some of the software based H.323 conferencing servers. There are also some projects[11] to implement SIP based conferencing servers.

4. Audio Conferencing

Contrary to human eyes, human ears are very sensitive to distortions in voice. In addition, human ears are also sensitive to the delay of voice from a speaker to the listener. Studies show that to give the feeling of one is having a direct conversation with the other party, the delay in the transmission of audio should not exceed 400ms. Therefore we need to pay close attention to these factors when designing an audio conferencing system. The delay is introduced by a combination of factors, but the most important ones are the delays caused by transmission, buffering and mixing. The distortion in the voice can be caused either by the package loss during the transmission or by the accidental deletion of speech data when suppressing the silence. Although the loss in the transmission is not in the scope of this paper, silence suppression is a very important part of any audio conferencing solution.

Audio conferencing over Internet can be implemented in many ways[11]. Here we give four different architectures.

A. Endpoint Mixing: All audio streams are delivered from audio sources to recipients without modifying or mixing them in server side. Audio middleware redistributes audio packages to interested parties by replicating them whenever necessary. A client may get more than one audio stream at a time and it should be able to mix and play them. IP multicast or software-based multicast audio conferencing solutions[12] use this mechanism. The advantage of this solution is that the audio middleware introduces minimal transmission delay since there is no mixing or transcoding during transmission. In addition, this solution scales very well, since the only computing performed in server side is to replicate the data. On the

other hand, this solution requires more bandwidth for each client since they may receive more than one audio stream at a time. Moreover, for this solution to work efficiently, each audio sender client should implement a silence suppression mechanism so that when a participant is not speaking, no audio data should be sent from that client. Otherwise, the scalability of this system will be severely affected and bandwidth sensitive clients will end up loosing some valuable audio data. Furthermore, each client should support all the codecs used in a session by all participants, since there is no transcoding during transmission.

B. Distributed Mixing: Another way of implementing audio conferencing over Internet is to move audio mixing process from audio endpoints to the server. This architecture will be very similar to the previous one except it will handle audio mixing on the server. There will be one audio mixer for each participant in a meeting and each participant will receive exactly one audio stream. This mechanism reduces the bandwidth requirement for the end user significantly. In addition, users can be given an option to choose the audio format they want. Moreover, audio streams can be silence suppressed before mixing. Therefore this architecture can be more flexible to deploy than previous one and a diverse set of audio clients can be supported with different network bandwidth capabilities. The scalability of this architecture will be similar to the previous case, but nonetheless it is a challenging task to develop such a distributed audio conferencing solution. It is particularly difficult to distribute computing load among different machines and to maintain such a complicated system Another disadvantage of this solution is that the mixer in the server will introduce some delay to the transmission of packages. However, our experiments show that if there is only one audio mixer along the way from audio sources to the target, the delay introduced by the mixer can be tolerable.

C. Hybrid Model: One can also imagine a hybrid architecture in which some clients will have a mixer dedicated for them on the server and some will receive all audio streams from servers directly and handle by themselves. This architecture will have the advantages of both systems and provide services according to the capabilities of the end user. Nonetheless this is also a complicated system and not very easy to develop and maintain.

D. Central Mixing: In this model there will be one central audio mixer in the server and all participating clients send their audio to that mixer. This mixer will mix all audio streams and then send the mixed audio to each participant. This solution is less computationally intensive compared to the second approach, since there is only one audio mixer rather than N audio mixers in a meeting. It also has the same advantages as the second approach, since it is providing each participant with one mixed stream. The main disadvantage of this solution is its scalability. Since there is only one mixer in the system, it can not support thousands of clients in a session. Nonetheless, our experiments show that such a system can support up to 300 clients easily.

4.1 Audio Server Implementation

We have chosen the centralized audio mixing model (Figure 2) from the listed alternative conferencing architectures above. It is easy to implement and flexible enough to support variety of clients with different capabilities. Its scalability is also good enough as it can be seen from performance figures.

One important point we need to consider while designing an audio server is the silence suppression. Although it is best to suppress the silence in audio streams at the source clients before sending it over the network, we assume that the incoming streams may not be silence suppressed. Because we do not have any control over audio clients. Therefore, it is essential for us to suppress the silence in each incoming audio stream in the server. Silence suppression is particularly important because when audio streams are mixed, silence packages add up and produce a lot of unwanted noise. In addition, silence suppression saves CPU time on the server machine when mixing audio packages, since the silence packages will not get mixed. Moreover, most of the time there is only one speaker in a meeting, and we can avoid mixing the audio streams if we suppress the silence properly.

Various silence detection algorithms proposed with different computational complexity and accuracy; HAM algorithm, Exponential Algorithm, Absolute Algorithm, Differential Algorithm [13], zero crossing rate algorithm[14], refined block oriented algorithm [15], Silence Compression Scheme of G.723.1 [16]. We use the refined block oriented algorithm for its simplicity and reasonable accuracy. This algorithm compares the average energy of each package to a threshold and decides that package as silence if enough consecutive packages are below the threshold. Since we do not employ a comfort noise generator, we wait half a second before deciding a package as silence to avoid deleting the silence packages between the words of a speaker. In addition, to avoid missing the beginning of a speech we examine the sub blocks of each package and decide it as speech if the average energy of a sub block is higher than the threshold.

Figure 2 shows how the audio server works. First audio streams are decoded to raw data, then repackatizer adjusts the sizes of audio packages if necessary. Since our system supports a variety of clients, not all of them use the same package size. While Polycom client uses 60ms packages, Rat 4.2.2 uses 20ms packages. Currently we use 60ms as our systems package size.

Silence detector passes the speech packages to the package queue and packages wait in this queue to be picked up by the audio mixer. Packages are buffered in this queue for a while to avoid missing the late arriving packages because of the jitter in the transmission time. Audio mixer polls all queues regularly and passes a copy of mixed audio data to subtracters. Mixer just adds the values of all available data and store the result in a short array instead of byte to avoid overflow or underflow. Then subtracters subtract the data of themselves from the received mixed data if there is any, store the result in a byte array and pass it to the encoder. If the mixed audio sample value is out of range for byte type, the maximum or the minimum byte value is assigned accordingly. We have not experienced any distortion of audio because of this value conversion.

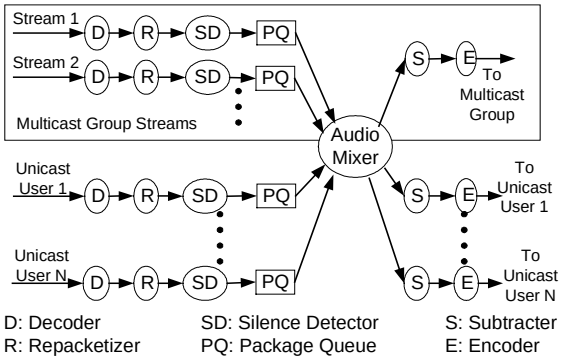

Figure 2. Centralized Audio Mixing

Since our system is designed to bridge Access Grid and other videoconferencing systems, we also support multicast groups in our audio server. It is similar to unicast support, but in this case usually multiple audio streams are received. The handling of these streams are the same up to the mixer, but after the mixer the subtracter subtracts the values of all streams from that multicast address. Therefore it sends the mixed audio of all unicast clients to the multicast group in a session.

We expect our system to work with any H.323 or SIP based multimedia client which supports any of the audio codecs, G.711, G.723.1, GSM. We have tested our audio server using an H.323 based Polycom client, SIP based HearMe and Windows Messenger clients and RAT from MBone tools. It works well with all these clients.

4.2 Audio Server Performance Tests

The performance of the audio server depends on the number of participants and active speakers in a session, and the number of audio sessions at a time. First we test the scalability of our audio server for one audio session. Audio server runs in a 2.5GHz Pentium 4 CPU, 512MB memory, Windows XP machine. All machines involved in this test run in a 100Mbps subnet. We tested it with one active speaker and two active speakers at a time. Speakers sent a 64kbps ULAW audio stream. When there is only one active speaker, no mixing is performed. This is the most common case in video conferencing environments since most of the time only one speaker talks. As it can be seen from Table 1, in this case, it provides a good quality audio for up to 300 participants. After that it starts dropping packages since it can not process it on time. Although we did not include the results for two active speakers case, it provides good quality audio for up to 275 participants and then it starts dropping packages. In this case two active speakers were always sending audio, so there have been continuous mixing activity.

We have also tested the effects of having multiple sessions at a time. We created audio sessions with 50 participants and two active speakers each. Our tests show that audio server can support 5 concurrent sessions (250 participants in total) without any package droppings. Since there are more mixers and more incoming audio streams, the supported users are a little less than the previous one session cases.

Number of users	CPU Usage	Mem. Usage (MB)	Total BW (Mbps)	Quality
50	% 3	39	3.2	good
100	% 12	60	6.4	good
150	% 24	80	9.9	good
200	% 38	101	12.8	good
250	% 49	121	16.0	good
300	% 56	141	19.2	fair
350	% 56	161	22.4	poor

Table 1. Audio Server performance results for one audio session with one active speaker

We should note the fact that even when the CPU utilization is quite high such as %50, audio server can still deliver a good quality audio because mixer runs regularly and CPU is utilized linearly. As long as it finishes processing a package during the given time interval, it will not delay the processing of the next package. But if it can not finish it on time, then it will delay processing of the next package. Since next package also will not be finished on time, it will delay the processing of the one after that. This will cause regular package drops from the package queue causing audio quality degradation for the end user.

5. Video Conferencing

The requirements for video transmission are significantly different than audio transmission. Video

streams entail much more network resources, and its encoding/decoding requires much more computing power. Moreover, there is no trivial way of mixing video streams as mixing audio. Although one can merge a number of streams into one, it is not easy to merge an arbitrary number of video streams into one.

Video conferencing can be implemented in many different ways. Here we give three different architectures and discuss the advantages and disadvantages of them:

A. Multicast Style: In multicast style conferencing, all participants send video streams to a group address and network or middleware servers deliver these streams to participants by replicating the data whenever necessary. This approach can be implemented either using IP multicast supporting routers or software based multicast implementations such as a distributed brokering system[12]. Access Grid uses ip-multicast for video conferencing. The advantage of this solution is its simplicity and ease of use for the end user. The main disadvantage of it is that it requires much more network bandwidth than many users can accommodate, since all video streams are delivered to all participants. Even if the network bandwidth is available, the computers which receive these video streams may not have the means of processing them.

B. Multicast Gateways: This approach is similar to first one, but for those users who do not have multicast support or who do not have enough bandwidth, or who do not have the capability of processing multiple streams, gateway(s)[17] can be placed in the edge of a multicast network. These gateways receive video streams from multicast network and forward the requested video streams to proper destinations. In addition, these gateways can also provide video merging and transcoding services. Moreover, they can implement a session management protocol to give users an option to choose the video stream(s) they want. H.323, SIP or a proprietary protocol can be used for this purpose. [18] uses gateways to merge many video streams into one and requires users less bandwidth and less computing power.

C. All Unicast: Third approach would be to avoid using multicast altogether and route the video streams through software or hardware based servers. The main advantage of this solution would be not requiring multicast support and can run anywhere. Although some sophisticated servers can handle very large scale meetings, this solution can be used effectively for small scale sessions.

5.1 Video and Image Grabber Server Implementations

Our video server(Figure 3) is effectively a multicast gateway with a general session management interface. Its main functions are to redistribute the received video streams, and to mix up to four video streams into one and send out to registered destinations. Every session in video server has a multicast group address, a video mixer and one or more unicast users. Each unicast video stream is forwarded to the multicast address and through this multicast address image grabber server receives all video streams in that session.

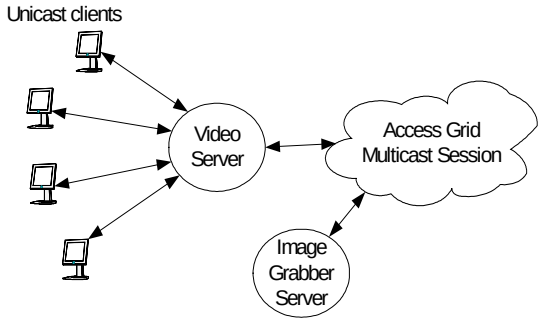

Figure 3. Video and Image Grabber servers

Image grabber server is independent of the video server and runs in another machine. Because it processes the incoming video streams continuously, it requires a lot of computing resources. It first decodes the received video streams into YUV format and then encodes them into JPEG format, and regularly saves the snapshots of these streams in JPEG file format. Web server accesses these pictures and provides them to users through a web interface. These pictures gives the end user a pretty good sense of what each stream is about, before deciding to choose a video stream to receive.

Figure 4 shows the video mixing algorithm in video server. Because of the space limitations, the details of video streams 2 and 3 are not shown but they are processed in the same way as the other two streams. After receiving a to be mixed video stream, a replicator filter duplicates video packages and passes one copy to RTP Transmitter which sends out to proper destinations and another copy to a video decoder which decodes the received stream into YUV format. Video Mixer first

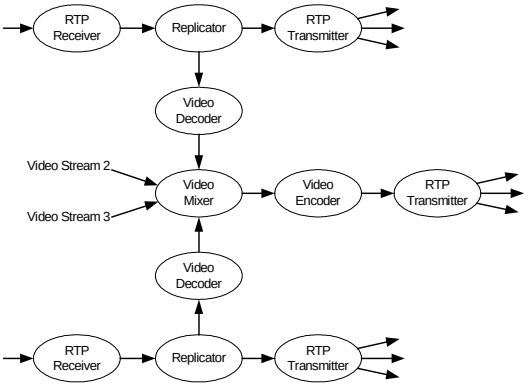

Figure 4. Video Mixing in Video Server

reduces the size of these incoming four video streams to one fourth of their original size. Then it combines these streams into one full picture. It places every stream to one corner of the merged video picture(Figure 5). In the next step, this newly mixed video stream is encoded either into H.261 or H.263 format and then it is transmitted to proper destinations by the RTP transmitter. Since Video mixing is a CPU intensive application, instead of giving each user an option to create his/her own mixed video stream, we create only one video mixer for each session and the system administrator has the right to choose the video streams to be mixed. Our web server provides an interface to add/remove video streams to/from the mixer.

Figure 5 shows the mixed video in VIC, Real Player and Polycom windows. In background a snapshot of our web page is also seen. Mixed video streams are particularly important for users who can not receive more than one video stream such as a Polycom client. These users get the pictures of four participants in a meeting through one video stream.

Figure 5. Mixed video streams in various windows

Since our system supports H.261 and H.263 codecs we expect it to work with any H.323 or SIP compatible multimedia client which supports any one of these codecs. We have tested it with a H.323 based Polycom client and VIC MBone tool. It works well with both of them.

5.2 Video Server Performance Tests

The testing of video server is more complicated since there are some video streams which are only forwarded and there are also some streams which are mixed. We have tested the performance of forwarding and mixing separately.

In forwarding case, one user sent a H.263 video stream to the server machine -1.2GHz Intel Pentium III dual CPU, 1GB MEM, RedHat Linux 7.3, and server forwarded it to many clients. Instead of measuring the CPU load of the server, we have calculated the delay, jitter and loss rates for each package over a period of time. This way we get a more precise way of assessing the perceived performance of our system by users. Although video server distributed the packages to hundreds of destinations, we gathered the results from 12 clients for the ease of testing. The sender client and 12 receiver clients, from whom we gather results, were running on a 2.4GHz Intel Pentium 4 CPU, 512GB MEM, RedHat Linux 7.3 machine. The video stream had an average bandwidth of more than 600 kbps. The sender application sends 2000 packages in each test, 23 second part of a movie. We used the same video stream for each test. All machines involved in this test reside on a gigabit subnet. For every package we calculated the transit delay, (receivedTime – sentTime), for all 12 clients and then we get the average of these 12 delay values in milliseconds. We also calculate the average jitter for each package based on the formula explained in RTP RFC [19].

Number of clients	Avg. Delay (ms)	Avg. Jitter	Loss rate	Total bandwidth (Mbps)
50	3.08	1.10	0.0%	30
100	10.72	3.34	0.0%	60
200	27.69	7.56	0.4%	120
300	60.86	11.84	0.7%	180
400	229.2	15.55	6.0%	240

Table 2. Video forwarding performance results

Table 2 shows that our video server is capable of supporting 300 clients if there is only one video sender. When it sends to 400 clients, it starts dropping packages and also latency becomes pretty high. We should also note the fact that this video stream had an average bandwidth of 600kbps which is quite high. In a normal video conferencing setting, the average bandwidth of a video stream is much lower.

In mixing test, we ran the video mixer in a 1.2GHz Intel Pentium III dual CPU, 1GB MEM, RedHat Linux 7.3 machine. The video mixer mixed four identical H.261 video streams. The bandwidth of this stream changed from 100kbps to 200kbps. In this case, since it is not easy to measure the latencies and jitters for each video package, we measured the CPU load on the server.

Table 3 shows that video mixing is a CPU intensive process and this machine can only handle three video

mixers at a time. These results also suggest that video mixing should be done in a separate machine than video forwarding. Although currently these two processes are running in the same machine, we plan to separate these functions and distribute them among more servers.

.

Number of Mixers	CPU load
1	15-25%
2	30-50%
3	50-70%

Table 3. Video mixing performance results

6. Conclusion and Future Work

In this paper we have presented our conferencing system which provides services to a diverse set of clients. We have also given a detailed description of the architectures of our servers and performance results. The performance results show that our current implementation supports hundreds of participants in audio and video sessions. It also shows that JMF can be used to implement such conferencing systems.

In the next step, we are planning to develop a distributed conferencing system which will be implemented on top of a distributed brokering system. We plan to support thousands of clients at the same time.

7. References

[1] International Telecommunication Union, "Packet based multimedia communication systems", Recommendation H.323, Geneva, Switzerland, Feb. 1998.

[2] J. Rosenberg et al., "SIP: Session Initiation Protocol", RFC 3261, Internet Engineering Task Force, June 2002, http://www.ietf.org/rfc/rfc3261.txt.

[3] The Access Grid Project, http://www.accessgrid.org.

[4] K. Singh and H. Schulzrinne. Interworking between SIP/SDP and H.323. In Proceedings of the 1st IP-Telephony Workshop (IPtel 2000), Berlin, Germany, Apr. 2000.

[5] Jiann-Min Ho, Jia-Cheng Hu, Peter Steenkiste, A conference gateway supporting interoperability between SIP and H.323, Proceedings of the ninth ACM international conference on Multimedia, 2001 , Ottawa, Canada

[6] Virtual Rooms Video Conferencing System, www.vrvs.org

[7] Geoffrey Fox, Wenjun Wu, Ahmet Uyar, and Hasan Bulut, A Web Services Framework for Collaboration and Audio/Videoconferencing, The 2002 International Multiconference in Computer Science and Computer Engineering, Internet Computing(IC'02), June 2002, Las Vegas

[8] Wenjun Wu, Ahmet Uyar, Hasan Bulut, Geoffrey Fox, Integration of SIP VoIP and Messaging Systems with AccessGrid and H.323, (to appear) the proceedings of The 2003 International Conference on Web Services (ICWS'03), June 2003, Las Vegas, ND, USA.

[9] Geoffrey Fox, Wenjun Wu, Ahmet Uyar, Hasan Bulut, Shrideep Pallickara, Global Multimedia Collaboration System, 1st International Workshop on Middleware for Grid Computing, June 2003, Rio de Janeiro, Brazil.

[10] Sun Microsystems, Java Media Framework 2.1, http://java.sun.com/products/java-media/jmf/2.1.1/index.html, 2001

[11] K. Singh, G. Nair, and H. Schulzrinne. "*Centralized conferencing using SIP*." In Internet Telephony Workshop 2001, New York, Apr. 2001.

[12] Ahme Uyar, Shrideep Pallickara, Geoffrey Fox, "*Towards an Architecture for Audio/Video Conferencing in Distributed Brokering Systems*", To appear in the proceedings of The 2003 International Conference on Communications in Computing, June 23 - 26, Las Vegas, Nevada, USA.

[13] Claypool M., Riedl J., "Silence Is Golden? - The Effects of Silence Deletion on the CPU Load of an Audio Conference", IEEE Proceedings of the International Conference on Multimedia Computing and Systems, May 1994, Boston.

[14] J. Junqua, B. Mak and B. Reaves, "*A Robust Algorithm for Word Boundary Detection in the Presence of Noise.*" IEEE Transactions on Speech and Audio Processing, vol.2, No.3, July 1994

[15] Gerischer F., "Design and Implementation of Audio Components for a Corba-based Multimedia Platform", Diploma Thesis, 1997, France.

[16] International Telecommunication Union, "*Recommendation G.723.1 Silence Compression Scheme*", Annex A, 1996.

[17] Elan Amir, Steven McCanne, Hui Zhang, An Application Level Video Gateway, In Proceedings of ACM Multimedia '95 (Nov. 1995), ACM.

[18] A. Mankin, L. Gharai, R. Riley, M. Perez Maher, and J. Flidr, "The design of a digital amphitheater," In Proceedings of The 10th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2000), Chapel Hill, NC, June 2000.

[19] RTP: A Transport Protocol for Real-Time Applications (IETF RFC 1889) http://www.ietf.org/rfc/rfc1889.txt.