

JQoS: Design and Implementation of a QoS-based Internet Videoconferencing System using the Java Media Framework (JMF)

Wenbiao Zhu and Nicolas D. Georganas

Multimedia Communications Research Laboratory

School of Information Technology and Engineering, University of Ottawa

{wenbiao, georganas}@mcrmlab.uottawa.ca

ABSTRACT

The heterogeneity feature of the Internet makes it hard for a single real-time multicast stream with unique or static quality of service (QoS) support to provide good service to all receivers. Applying network-level QoS guarantees or application-level QoS adaptations are often considered effective solutions to this issue. However, the current proposed protocols or schemes have problems in scalability and implementation. This paper proposes a simple but efficient application-level dynamic QoS control mechanism and demonstrates it via implementing a QoS-based Internet videoconferencing system using Sun Microsystems' Java Media Framework (JMF). The dynamic QoS self-adaptation model has achieved the goal in receiving clear audio and visually useful video. The intelligent QoS request-handling algorithm may adjust the source transmission to satisfy most of, if not all, session receivers' requests. The receiver's QoS self-adaptation ability allows session receivers to apply their individual interests on the receiving streams.

1. INTRODUCTION

With the rapid advances in computer and network technologies, especially with the emergence of Internet, Internet-based videoconferencing (VCoIP) has sparked a new future for videoconferencing applications. Comparing to the traditional videoconferencing system, the promised ability of the Internet Protocol allows VCoIP to transmit any data over virtually any network transmission media and communicate between virtually any system platforms. The multicast ability of the Internet allows a multipoint VCoIP system to direct and distribute a single data stream to multiple receivers without increasing the network bandwidth. Most importantly, the low cost on using a computer and the Internet, as well as its ubiquitous accessibility, make VCoIP applications have a great number of potential users.

However, this inexpensive and convenient approach is not coming without pain. The problems existing in the VCoIP system make it far from perfect. The following sections address two of them.

1.1. Heterogeneity Problem

Multicast (one-to-many) transport mechanisms allow a videoconferencing source to transmit data to a number of receivers simultaneously without the need to replicate data on single connections to each receiver, as in the traditional unicast (one-to-one) case. However, within multicast groups, deriving end-to-end satisfactory performance for a number of receivers poses a number of problems. The major problem lies in resolving heterogeneity issues that may include the diversities in end-user QoS requests, end-user system capabilities and end-user network connections.

For example, in a real-time videoconferencing session with many participants, some receivers may suffer congestion when a link or links on the path from the source to the receiver have limited bandwidth or become congested with other traffic, while some others are fine. As a result, a single multicast stream with unique or static quality of service support cannot always provide good service to all receivers.

1.2. Transmission Problem

The current Internet Protocol (IP) can only provide "Best Effort" service. It is based on the available network bandwidth and FIFO forwarding schemes¹. There is no guarantee as to timely or actual delivery. But the data of a videoconferencing application are all delay-sensitive and error-sensitive media streams. The bandwidth constraint often makes transmission experiencing unpredictable

¹ If the queue in the IP router for the next hop is long, the packets may be delayed to forward. If the queue is full or unavailable, an IP router is allowed to drop the late coming packets. [1]

delays and data loss that greatly degrade the receiver's receiving and playback quality.

1.3. The Effective Solution: QoS Controls

Providing QoS control to a VCoIP system is commonly considered an effective solution to those problems. Quality of Service or QoS is the ability of a network element (e.g. an application, host, router) to have some level of assurance that its traffic and service requirements can be satisfied [2]. The Internet Engineering Task Force (IETF) and other research institutes have proposed many QoS protocols and schemes. We may classify them into two categories:

- Network-level QoS controls are those usually provided on routers or hops to assure timely and actual delivery.
- Application-level QoS controls are those usually based on the specific application or end-users.

Notably among those are included: *Integrated Services*, *ReSerVation Protocol (RSVP)*, *Differentiated Services*, *Constraints based Routing*, *MultiProtocol Label Switching (MPLS)*, *Internet2*, etc. All of these are trying to present approaches to deliver network-level QoS controls over the Internet.

Besides providing the network-level QoS guarantee, with the underlying transport service, the application-level QoS adaptation can also deliver QoS to the heterogeneous end-users and network environment.

QoS-filtering mechanisms and layered-multicasting mechanisms represent two typical application-level QoS controls.

In [3], fundamental to the QoS Filtering mechanism is the use of QoS filters to allow a stream to be transmitted to multiple clients with different QoS requirements. However this mechanism requires instantiating filter operations to complete a particular negotiation and renegotiation. The stream cannot be sent until the negotiation is done. This is not suitable for Internet videoconferencing real-time transmission because no matter how powerful a filtering engine may be, time will always be consumed. This affects both the transmission delay and the jitter.

In the layered-multicasting mechanism [4], a source simultaneously transmits several hierarchically layered streams, each on a separate multicast group. Each layer represents an increasing level of quality. The base layer describes a basic quality level. Adding incremental layers in order increases the quality of the representation. Each receiver can tune its own received quality by adjusting the number of layers it receives. However, applying this kind of QoS control mechanism, we have to know the exact process of the media transcoding otherwise there no way to establish such hierarchically layer-encoded streams.

In the following sections, we present our QoS control mechanism and demonstrate it via implementing a QoS-based Internet videoconferencing system by taking advantage of continuous media compression techniques, the real-time transport protocol RTP/RTCP, the Java Media Framework and Remote Method Invocation techniques. Section 2 describes the overall architecture and its design issues. Section 3 presents the implementation of the system in JMF. Section 4 gives experimental results and evaluation. Finally, conclusions and future work to be done are presented.

2. OVERALL ARCHITECTURE

The design goal of a QoS-based VCoIP system contains three major aspects. Firstly, the system has a dynamic QoS self-adaptation ability based on the current network transmission performance. Secondly, the system has an intelligent QoS requests handling algorithm that takes care of diverse session receivers' requests and makes a proper QoS adaptive decision not just based on the requests but on the current system QoS adaptation status as well. Thirdly, as a session participant, the receiver has somehow itself a QoS adaptation ability to express its individual interests on the receiving streams.

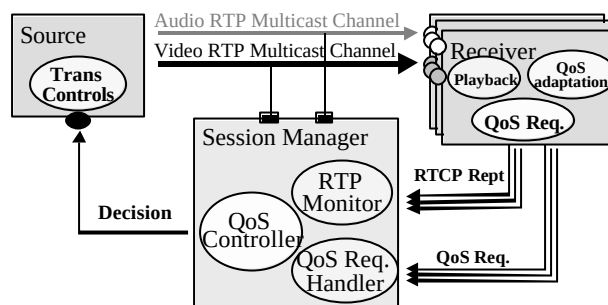


Figure 1. Overall architecture

As shown in Figure 1, a typical QoS-based Internet videoconferencing architecture can be described as follows: a multipoint videoconferencing system has a *Source* and a group of session participants – *Receivers*. The *Source* is capable of establishing a videoconferencing session by sending compressed live audio and video streams to the Internet via separate RTP multicast channels. The RTP multicast channel is identified as a multicast IP address associated with the port number. The *Receivers* can join the session by subscribing to those channels. After successfully joining the session, the receivers are able to receive the RTP streams and play them back. While receiving the coming streams, each receiver periodically issues a series of RTCP reports to the network to report its most recent receiving status. In addition, *Receivers* are able to perform QoS self-adaptation

and issue specific QoS requests if they want. Obviously, if every receiver sends QoS request directly to the *Source*, it may cause source overload and crash. Therefore, a *SessionManager* has been developed. The *SessionManager* serves like a gateway logically sitting between the *Source* and *Receivers*. In the *SessionManager* module, the *RTP Monitor* is responsible for tracing the transmission performance and monitoring each session receiver's receiving status by collecting the receiver's individual RTCP reports. *QoS Request Handler* is used to take care of each receiver's QoS request. The request could be either quality improving or quality degrading. The *QoS Controller* will make a proper QoS adaptive decision based on the information provided by the *RTP Monitor* and *QoS Request Handler*. The decision may be sent to the *Source* if necessary. The *Source* provides a couple of transmission control interfaces that may accept the decisions and adjust the transmission accordingly.

The challenge in system design is to establish a proper dynamic QoS control mechanism. To simplify the problem, we may decompose the mechanism into two sub-models: a dynamic QoS self-adaptation model and a QoS Request Handling model.

2.1. Dynamic QoS Self-adaptation Model

The dynamic QoS self-adaptation model is very much analogous to a feedback control system.

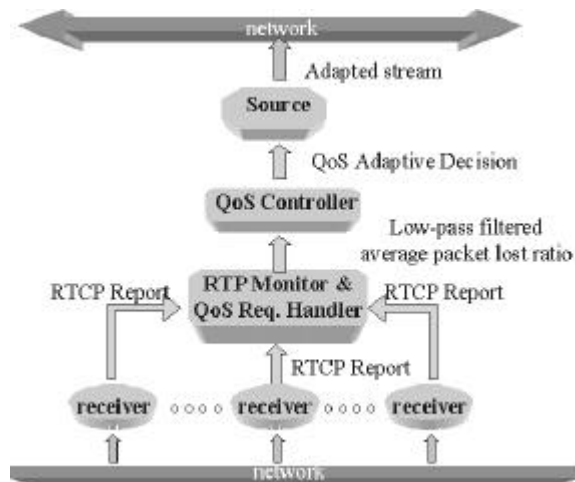


Figure 2 Dynamic QoS self-adaptation flow chart

As shown in Figure 2, after a videoconferencing session has been established, each session receiver issues its RTCP report to the network periodically. The RTCP report contains the receiver's id and current receiving status including the packet loss ratio. The *RTP Monitor* collects these reports and uses the low-pass filter to

calculate the average packet loss ratio². The *RTP Monitor* then sends the low-pass filtered average packet loss ratio to *QoS Controller*. The *QoS Controller* makes a proper QoS adaptive decision and sends it to the *Source* if necessary. The *Source* dynamically adjusts its transmission rate accordingly and sends an adapted stream to the network.

If we let R_i stand for a receiver's packet loss ratio. λ_n is an n^{th} average packet loss ratio. With the total number of N session receivers, it can be computed using the following formula:

$$l_n = \frac{\sum_{i=1}^N R_i}{N}$$

$$l_n = \alpha l_n + (1 - \alpha) l_{n-1}$$

$$(0 \leq \alpha \leq 1)$$

where α is a low-pass ratio. Increasing α increases the influence of the current value while decreasing α results in a higher influence of the previous values. Using a low-pass filter we may smooth the statistics and avoid QoS oscillations. [5]

As a result of QoS self-adaptation, the system may stay in two distinct states: one is called *DOWN* status that indicates the system is currently in QoS degrading status, while the other is called *AVAIL* status that indicates the system is either in QoS upgrading status or holding status. The system may be in QoS upgrading status under the receivers' request.

2.2. QoS Request Handling Model

Our QoS-based VCoIP system allows session receivers to issue specific QoS requests to the system. These requests are handled by the QoS request-handling model. The following diagram demonstrates the handling sequence.

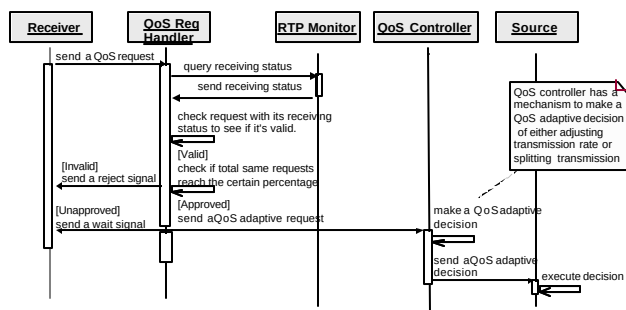


Figure 3 Sequence diagram of request handling model

A *Receiver* issues a QoS request to the *SessionManager*. The *QoS Request Handler* receives the

² In the current model only the packet loss ratio is used as congestion indicator.

request and sends a message to the *RTP Monitor* to query this receiver's receiving status. If the feedback from the *RTP Monitor* shows this receiver is suffering high packet loss, but the request requires quality upgrading, then the request will be rejected because quality upgrading will result in increasing the transmission rate, which in turn will worsen this receiver's receiving status. On the other hand, if the receiver's current receiving status is good, the request is then considered valid. But the *SessionManager* won't issue a QoS request to the source until it receives a sufficient number of similar requests. The *QoS Request Handler* first checks if the request is redundant to ensure the receiver sends one request at a time and then checks the total number of the corresponding requests. If the number does not reach a predefined percentage, this request will be kept in a safe place, say a vector in Java. If the *QoS Request Handler* gets the enough number of the corresponding valid requests, it will forward the request directly to the *QoS Controller*. The *QoS Controller* checks the current system status and makes a proper QoS adaptive decision. If the current system is in *AVAIL* status, the system may accept any QoS adaptive requests. If the current system is in *DOWN* status and the request is requiring degrading quality, the system will stay in degrading status. But if the request is requiring increasing quality, the *QoS Controller* will make a decision to split the source transmission.

2.3. QoS Adaptation Policy

To efficient manipulate the QoS adaptation for our VCoIP application; we have defined the following policy:

- An Audio stream always has higher priority to use network bandwidth, CPU and other related resources than a video stream.
- To avoid the QoS oscillation, the system's QoS self-adaptation only performs quality-degrading activity.
- The system may perform quality-upgrading adaptation under receivers' requests.
- If the QoS adaptive requests conflict with system status, source transmission may be split into several separate sessions with different QoS levels to satisfy various QoS requests.

3. IMPLEMENTATION

Our dynamic QoS control mechanism is demonstrated via implementing a QoS-based VCoIP system using the Java Media Framework (JMF). As a media-developing tool, JMF provides a set of easy-to-use APIs that make manipulating multimedia streams much easier than developing them from scratch. JMF RTP APIs enable working with real time media streams. In the next part, we will introduce the

implementation of the system *Source*, *Receiver* and *SessionManager* by taking advantage of JMF.

3.1. Source

In JMF, to transmit an RTP stream, we have to use a *Processor* to produce an RTP-encoded *DataSource* and construct an *RTP SessionManager* to manage the session. Figure 4 shows the procedure to produce an RTP stream.

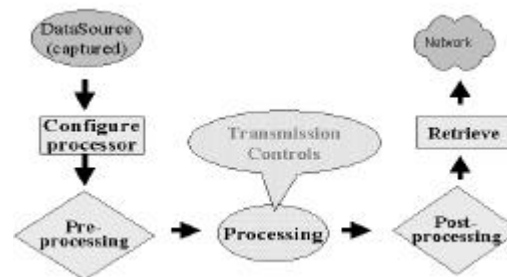


Figure 4 The procedure of producing an RTP Stream

Our job is adding transmission controls to the processing in order to affect the media transcoding. The following snippet shows how to retrieve supported transmission control interfaces from each stream track in JMF.

```

TrackControl[] trackCtl = myProcessor.getTrackControls();

for (int i=0; i<trackCtl.length; i++) {
    format = trackCtl[i].getFormat ();
    if ( format instanceof VideoFormat ) {
        videoControl = trackCtl[i].getControls();
    }
    if ( format instanceof AudioFormat ) {
        audioControl = trackCtl[i].getControls();
    }
}

```

After getting the transmission controls, one may define and implement a transmission adjustment scheme and provide corresponding RMI interfaces to allow the remote object (e.g. the *QoS Controller* in the *SessionManager*) to call. In JMF, the common control APIs for the stream used in our VCoIP application include:

- G723 or GSM Audio Stream
 - *BitRateControl()*, *SilenceSuppressionControl()* ...
- H263 Video Stream
 - *BitRateControl()*, *KeyFrameControl()*,
 - *QualityControl()*, *PacketSizeControl()* ...

While at the split situation, we may replicate the captured *DataSource* by calling the *createClone()* method. Through a similar processing procedure a new session with different QoS level will be established.

3.2. Receiver

JMF *Players* and *Processors* provide presentation mechanisms for the RTP streams. A separate player is used for each receiving stream. You may construct a *Player* for an RTP stream through the standard Manager *createPlayer()* mechanism. Figure 5 is an example for audio and video players.



Figure 5 Audio and video players

Besides receiving and presenting the RTP stream, Receiver also has QoS self-adaptation ability. Figure 6 presents its controller panel. By clicking the *ImproveQuality* or *ReduceQuality* button, the Receiver may send a QoS request to the *SessionManager* via a pre-established RMI connection. If the source transmission is split, the color of the *Split* button will turn to red. The Receiver may click it to get the new session's information such as its new session address, transmission rate, etc. The *BufferControl()* API allows the receiver to adjust the actual receiving buffer size and threshold to reduce the delay jitter.



Figure 6 Receiver's QoS controller GUI

3.3. SessionManager

The *SessionManager* plays a key role in our QoS-based VCoIP system. The overall GUI of the *SessionManager* contains an RTP Monitor, a real-time BitRate chart, a real-time low-pass filtered average packet loss ratio chart and a H263_RTP QoS Control Panel. An RTP Monitor GUI contains three parts. Parameters and Status part not only lets you initialize the QoS parameters such as low-pass ratio but also indicate the current status of the system. The Global Statistics part shows the current transmission status. The last part comes with a list of session receivers' feedback reports that indicate each receiver's receiving status (see Figure 7).

Username	Fraction Lost	Jitter	Packet Lost
Wenbiao@c3po	0.0	7286	137
Wenbiao@luke	0.0585	6063	173
Wenbiao@guigan	0.0703	8398	125
Wenbiao@r2d2	0.0898	8301	144

Average Packets Lost ratio per interval: 0.05400390625

Figure 7 Receivers' feedback reports in the RTP Monitor

In JMF, there is an interface called *RTCPReport* to represent both RTCP SR and RR packets. The method *getReports()* returns a vector containing the last SR or RR reports sent by the participants. The *RTCPReport* also has a method called *getFeedback()* that returns a vector of *RTCPFeedback* objects. Each *RTCPFeedback* object conveys information about the reception of one incoming stream. Feedback information includes fraction lost, cumulative number of packets lost, etc.

A QoS decision based on the information of the feedback reports and receivers' requests is sent to the session source via an RMI connection.

4. EXPERIMENTS

The experimental results are produced from different testing scenarios. Although the testing was held on our in-house test bed, since it can simulate different network environments elegantly, we believe the conclusions are similar to those from testing over the real world Internet.

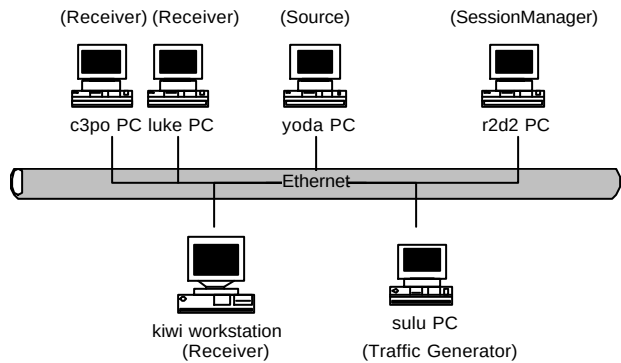


Figure 8 Test bed

Figure 8 shows the layout of our test bed. It is an isolated Fast Ethernet LAN with one Sun SPARC 20 workstation and 5 PCs running windows NT. "yoda" serves as a videoconferencing *Source* while "r2d2" plays the role of the *SessionManager*. "sulu" is a network traffic generator. The other machines act as session receivers.

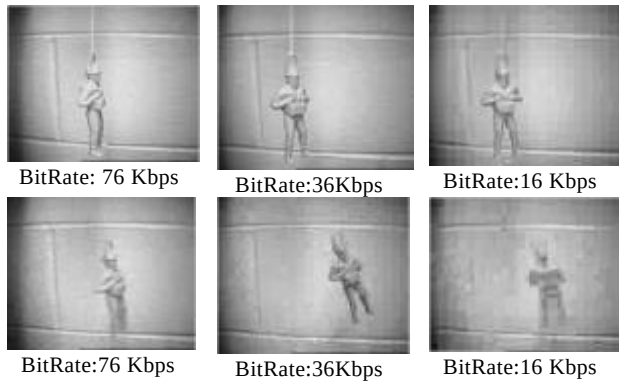


Figure 9 Upper row: still object. Bottom row: moving object

We first investigated the transmission characters of an H263/RTP video stream. Through testing on the H263/RTP video stream under different transmission rates (see Figure 9), we conclude that H263/RTP is very promising for very low-bit rate video transmissions, especially when the video has not very much fast moving objects such as videoconferencing video.

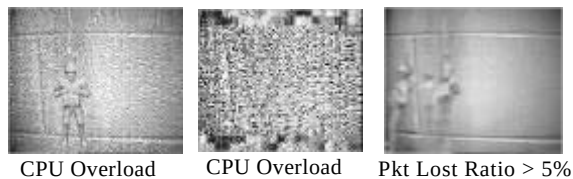


Figure 10 CPU overload and high packet loss scenarios

Figure 10 demonstrates that without the QoS adaptation, high transmission bit rate may cause CPU overload and heavy packet loss. As a result, the playback quality will suffer unpredictable bad effects.

Table 1 Playback Quality of Various JMF Audio Formats

Encoding	Sample Rates	Bits per Sample	Transmission BitRate	Playback quality
DVI/ RTP	8000	8	~ 32 Kbps	clear
G723/RTP	8000	8	~ 6.325 Kbps	clear
GSM/ RTP	8000	8	~13.44 Kbps	clear
ULAW/RTP	8000	8	~65 Kbps	noisy
MPEGAUDIO/ RTP	48000	16	~64 Kbps	longer delay

Table 1 shows the comparison of playback quality among various audio formats supported by JMF. We may conclude that G723/RTP and GSM/RTP are more suitable than other JMF audio formats in our Internet videoconferencing application, since they use less transmission rate but still get clear playback quality.

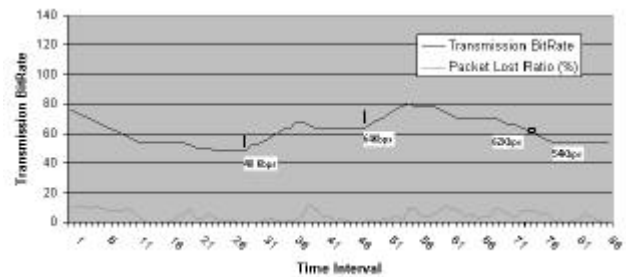


Chart 1 Transmission BitRate vs Packet loss Ratio (%)

Our dynamic QoS control mechanism works well in our application. Chart 1 demonstrates the transmission rate decreases based on the packet loss ratio, while it increases under receivers' requests. The red lines indicate that the system got valid QoS upgrading requests and the red circle shows that the source splits its transmission session, once the system in *DOWN* status has received a QoS upgrading request.

5. CONCLUSIONS AND FUTURE WORK

This research project set out to discover a way to support application-level QoS control for videoconferencing applications over the highly heterogeneous Internet. The QoS-based VCoIP system has demonstrated that our dynamic QoS control mechanism is feasible and indeed essential to providing an intelligent QoS adaptation to satisfy most, if not all, of session receivers' requests.

JMF is an easy-to-use media-developing tool, while problems in lack of robustness and flexibility require extra efforts on improvements.

There are many QoS related issues that could be considered as continuation work for this research, such as audio-visual synchronization, resource reservation, QoS mapping, etc.

ACKNOWLEDGEMENTS

This work was supported in part by the Telelearning Network of Centres of Excellence, under a research grant.

6. REFERENCES

- [1] White Paper - "The Need for QoS," *QoS forum*, Stardust.com Inc., California, USA, 1999.
- [2] Xipeng Xiao and Lionel M. Ni, "Internet QoS, the Big Picture", Michigan State University, USA, 1999.
- [3] Nicholas John Yeadon, "Quality of Service Filtering for Multimedia Communications", Ph.D thesis, Lancaster University, UK, 1996.

[4] SanKu JO and Pierce E. Cantrell, "*A dynamic QoS control scheme for videoconferencing in heterogeneous Internet*", Lawrence Berkeley National Laboratory, Texas A&M University, USA, 1999.

[5] Ingo Busse, Bernd Deffner and Henning Schulzrinne, "*Dynamic QoS control of multimedia applications based on RTP*", in Proc. First International Workshop on High Speed Networks and Open Distributed Platforms, St. Petersburg, Russia, June 1995.

[6] Elan Amir, Steven McCanne and Randy Katz, "*Receiver-driven Bandwidth Adaptation for Light-weight Sessions*", Proceedings of ACM Multimedia '97, Seattle, WA. Nov 1997.

[7] Francisco Afonso, "*Virtual Reality Transfer Protocol (V RTP): Implementing a Monitor application for the Real-time Transport Protocol (RTP) using the Java Media Framework (JMF)*", Master's thesis, Naval Postgraduate School, USA, 1999

[8] Guojun Lu, "*Communication and Computing for Distributed Multimedia System*", Artech House Publishers. ISBN 0-89006-884-4, 1996

[9] "*Java Media Framework API Guide*", Sun Microsystems, USA, 2000