ABSTRACT: - Most of the local area networks have been designed for non-real time transmission of digital data as text and images. To achieve audio data transmitting over the network in efficiency manner (real-time), the implementation of audio-on-demand using the Real Time Protocol will be discussed. The performance of the system implementation both of the resulting packet loss and delay jitter will be presented. This paper describes audio streaming transmission protocols that used to implement the system, the system architecture and how the system investigates and addresses the previous issues. The Java Media Framework library (JMF) and the Real-time Transport Protocol (RTP) for audio transmission were used as development tools.

Keywords: - Multimedia, Real-time Transport protocol, audio-on-Demand, buffering time, Java Media Framework, Media Locator.

1. Introduction

Streaming is media content [1] that is delivered to the viewer’s media player in real-time. Media content is transferred at the same rate as it is displayed. There is no intermediate storage or caching of the content between its origin and the player. The data is processed as it arrives at the player, and then discarded [2]. To make streaming media a successful technology, the following issues have to be address: (1) Initial delay of playing time that’s mean downloading time, audio and video clips are usually large compared to other types of data (e.g. text, images.) and it may take a long time to start playing, (2) Current Streaming Protocols which not concern well with network Congestion, (3) Compression algorithms efficiency, (4) Network bandwidth utilization (Network infrastructure) and (5) Security concerns of content owners.

The first three issues are of a technical nature that can be addressed with technical solutions. The other two issues are of a non-technical nature.

Downloading approach may be used to transmit the media content. In this method the user must wait for the entire file to finish downloading before it can play.

The current streaming protocol implementations do not deal well with network congestion at higher rates (DSL rates and above). The emphasis on “edge caching” or “edge streaming” of a number of products and services supports this assumption. The inability to deal with real-world networking conditions at higher bandwidths results in the annoying blinking congestion message in a player application. A streaming media protocol should be able to deal with the following three network traffic categories:

- Constant packet loss between 2% and 10%. Typically backbone routers drop packets in order to manage heavy traffic.
- Changing latencies between 50 and 300 ms. Latencies are caused by queued packets in a router. Each router between a client and a server adds extra latency.
- High packet loss events. If a router becomes too congested due to high traffic, the router will drop packets during a period between 1 to 10 seconds. A router can only hold a finite number of packets in its queue, dropping packets is a way to decrease the number in the queue.

Network bandwidth utilization, this issue is a complex, because it’s mainly depending on the network infrastructure. To reduce the effect of this problem, the best solutions at the moment are to use two different solutions: Caching and Content Delivery Networks (CDN). Caching is done by installing streaming media caching appliances (for example Cache Flow appliances) inside local ISP’s just before the backbone connection. The idea behind a CDN is
that it is cheaper not to stream over the backbone by installing streaming servers inside large ISP’s. These two solutions work only for certain streaming applications well: caching works well for popular content and CDN serve only content from content providers that pay the CDN to host their content.

When audio media [2] is streamed, a small buffer space is created as shown in Figure 1 on the user's computer, and data starts downloading into it. As soon as the buffer is full (usually just a matter of seconds), the file starts to play. As the file plays, it uses up information in the buffer, but while it is playing, more data is being downloaded. As long as the data can be downloaded as fast as it is used up in playback, the file will play smoothly. Usually there is a delay of only 10-30 seconds before the audio or video starts to play.

This paper, which describes a real-time streaming service system that is designed to experiment with audio-on-demand and broadcast issues using RTP. The streaming system contains two or more stations, one of them used as server and the other used as clients. The connections between the stations can be LAN or WAN. Live audio streams and stored audio data can be established and controlled in real-time based on IP addresses.

2. Audio Streaming Transmission Protocols

Streaming files are transmitted over computer networks using special protocols that enable the transmission of the files without the need to download them. Two of these protocols are the Real Time Transport Protocol (RTP), which is a transport protocol that was developed for streaming data including audio and video with real-time properties, and the Microsoft Media Server (MMS) protocol. MMS is an application protocol used to access unicast content from a Windows Media server [7]. MMS is a Microsoft proprietary protocol and its specifications are not open for public. The developed application that will be described in this paper is based on the RTP protocol, which is supported by Java Media Framework (JMF). So RTP protocol and JMF were used as development tools in this application.

Figure 2 shows the transmission protocol layers. The control signals are designed over TCP for controlling streaming service system session [3]. A streaming service system session is an association between a station that worked as client and a station that worked as a media server of our system starting from initial to the last request of the client. RTP/UDP is used for media transport and RTP control protocol, RTCP/UDP for QoS feedback.

Three logical communication channels are established during a single streaming service system session, a control channel for streaming service system session; an audio data channel; and an audio feedback channel. During initialization, the station that worked as client opens the control channel for communicating with the station that worked as media server.

The Realtime Transport Protocol (RTP) can be used on top of UDP. It provides additional presentation information (timestamps, payload type) for the price of some additional transmission and processing overhead.

Audio transport in our system is done by RTP session. It reads audio file from media data store locates the data offset from the header information contained in the file, reads audio file data starting from the data offset and packetizes into RTP packets. In this case, RTP header is followed by audio data, no extra payload specific header is required. An audio RTP packet typically contains 40 ms of audio data. Market bit is set to zero in all audio RTP packets. Audio RTP session in the client reorders the RTP packets based on sequence number information.

The Network Adapter component mainly encapsulates RTP packets in UDP packets and extracts RTP packets from UDP packets. The components are implemented as an instance of RTP Socket class, whose Send () and Receive () methods make it possible to transmit and to receive RTP packets. The defined software architecture is shown in figure 3:
3. System Architecture
The system is developed using client-server architecture and it is designed to support unicast streaming (Audio-on-Demand and Broadcast transmission) as shown in Figure 4. The application program as illustrated in Figure 5 contains transmitting audio stream program, which send the audio stream in one station.

During a streaming service system session, the process of state transition in the media server is illustrated in Figure 7:

- **IDLE**: an initial state of the server where it waits for a new client register request, after which it transits to **WAIT** state. When end request is received it comes back to IDLE from any other states.
- **WAIT**: the station that works as media server waits for additional requests.
- **PLAY**: an active transmission state. The station that work as media server enters this state upon receiving a valid plays or resume request.
- **STOP**: the station that works as media server transits from PLAY state to this state when transmission is halted.

To send a Data Source (Audio stream) over the network, Data Sink method is used to transmit one stream. On the receiver side, a Session Manager is used to present the audio clip on the JMF Player. The complete system design is shown in Figure 6.
4. Implementation

The hardware environment for the development of Streaming Audio System consists of one station as a media server and the other stations as clients. The machines are connected by a 10 Mbps Ethernet. The programming language used to develop the application programs is a Java Media Framework JMF because of its multimedia capabilities. JMF is an extension of Java language and it offers an end-to-end solution, starting with the creation of multimedia presentations (Player) and allowing their on-demand transmission with RTP protocol. It is an application-programming interface (API) for incorporating time-based media into Java applications. JMF is developed by Sun Microsystems, Intel Corporation, Silicon Graphics, and IBM. The clips transmitted over the network with mp3 and wav formats.

As mentioned before, this system is based on the Real-time Transport Protocol (RTP), which is the Internet-standard protocol for the transport of real-time audio data and can be implemented by using Java Media Framework. RTP over UDP is used for real-time transfer of audio rather than reliable TCP because when packet losses occur retransmission and congestion control methods used in TCP result in gaps during media presentation. The main features of our system are real-time transmission, scalability, use of standard protocols, flexibility, and user interactivity. Also this paper investigated the potentiality of the JAVA platform in the design and implementation of network protocols supporting the real-time transmission of multimedia information.

The development system is implemented firstly to examine and analyze the current state of the art in streaming audio over network, and secondly to suggest and develop a means by which the Java Media Framework may be used to implement a client/server approach to streaming audio to obtain the best subjective experience for the listener. The overall evaluation of the efficiency of the development system is performed both in terms of resulting packet loss and delay.

The GUI is developed using Java Swing classes API but the Player implemented by using JMF API. To support user interactivity during playback, the Player has the following features: pausing and restarting playback of the stream at any time, positioning of playback for streams read from files, control over the sound volume, ability to disable sound (mute), for video playback ability to adjust the screen size. For each media data type, there is control provided, which can display stream processing statistics, encoding type, playback duration, sound frequency, frame rate, and other stream specific data.

5. Results and Discussion

In these experiments, the developed system was used to investigate the following parameters: bandwidth, delay, and delay jitter, packet loss. Delay, jitter and bandwidth are varying with traffic load. In case low traffic load, low delay, jitter is small, situation under control. Increasing traffic load causes larger delay, jitter variations increase, packet loss, bit errors, so larger playback buffer needed to avoid large packet drop ratios. Also for fast-Start Streaming, the buffer must be filled as fast as possible, JMF Player has this feature, but if buffer too small packets must be dropped.

For the first demonstration, the station that contains the transmitting application sends audio streams to the other station(s) that contain receiving application via a 10MBit/s LAN environment that is implemented in the Intelligent Systems and Robotics Laboratory – ITMA, UPM University.

The second demonstration setup inside UPM campus, the audio streams were transmitted from Multimedia and Intelligent System Laboratory in the Department of computer and communication systems engineering to Intelligent System and Robotics Laboratory in ITMA over Ethernet as shown in figure 8.

Figure 8: Audio Streams are transported via RTP, UDP, IP and Ethernet.

5.1 Streaming Stored Audio over Laboratory LAN

The first part of this experiment is streaming stored audio, which it’s means streaming the media data in Media-on-Demand mode. Client requests a certain audio file to be transmitted from the server to be played out at this client. Client creates a signal channel to deliver some control data to the server as follows: Client requests from the server a list of clips, then Server sends it to that client, after that Client makes its selection and sends it to the server, Server interprets the request and processes it, Server reads the clip and
sends it over the network, finally, Client can start playing the selected clip.

Table 1 lists audio streams parameters of our experiment conducted over LAN. Figure 9 shows the buffering and playing operations during streaming process experimental. There is no interruption or jerky, because the bandwidth is so high (155 Kbps), also the changing buffering time has no effect in this case, so it’s proved that any streaming system is limited primarily by bandwidth.

Table 1: Streaming Stored Media Numbers

<table>
<thead>
<tr>
<th>IP Transmitting</th>
<th>202.184.36.212</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Receiving</td>
<td>202.184.36.97</td>
</tr>
<tr>
<td>File size</td>
<td>1.21 MB</td>
</tr>
<tr>
<td>Format type</td>
<td>Wav</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>151 Kbps – 155 Kbps</td>
</tr>
<tr>
<td>Buffering time</td>
<td>250 , 500 , 1000 msec</td>
</tr>
<tr>
<td>Playing time</td>
<td>276 sec</td>
</tr>
<tr>
<td>Media locator</td>
<td>rtp://202.184.36.97:49150/audio</td>
</tr>
</tbody>
</table>

Figure 9: Streaming Audio Media With Wav Format

For the audio stream, at the server side, packets are sent at the constant bit rate. A buffer is implemented at the client side to smooth out jitter. Two events can occur: a new packet arrives, detected by using a select statement, and/or the buffer is not empty. In the case that a new packet arrives, it is added to the buffer. If no packet arrives, then the buffer is checked to see if it contains any old packet that has not been played yet. If the buffer is not empty, then a packet is taken out of the buffer to be played.

The audio packets are added to the buffer according to the rate specified previously plus an amount of time equal to the delay in the network. Packets leave the buffer to be played out at a time derived from the specified rate, arrival time of the first packet in the stream, and current time. But the buffering limitations in the client end systems are becoming a non-issue.

5.2 Streaming Stored Audio over UPM Campus

In this experiment, the audio streams transmitted over UPM campus from Multimedia and Intelligent System Laboratory in the Department of computer and communication systems engineering to in Intelligent System and Robotics Laboratory in ITMA. Table 2 lists audio streams parameters of our experiment.

Table 2: Streaming Stored Media Over Upm Campus

From the Table 2, we note that the bit rate during the playing is not stable, because the bandwidth transmission depends on network traffic load.

Figure 10: Buffering Time And Playing Operation

The buffering time effect in the playing operations during streaming process experimental is shown in the
As mention in the previous section, Initial delay of playing time results from buffering on the player side. The size of the buffer is configurable in JMF player. The reason for using a buffer is to absorb network jitters (changing network latencies). So we can choose between a long start-up time and smooth media playback (larger buffer size) or a shorter start-up time and more interruptions in the media playback. The relation ship between playing time and buffer length is shown in figure 13.

Figure 11: Buffer And Playback Relation

Also as we note in Figure 12, the relation between buffer length and initial delay of playing is approximately linear. The buffering length values are 250 ms, 500 ms and 1000 ms (max value), while initial delay of playing time is also function of network traffic. So initial delay of playing time is caused by queued packets in a router (each router between a client and a server adds extra latency) and buffering time.

Figure 12: Relation Between Buffer Length And Initial Playing Delay

6. Conclusion
The developed audio streaming system successfully addresses the three issues: Initial delay of playing time; streaming protocols and Security concerns of content owners. The quality of streaming audio is low and sometimes the playing is jerky. But this technology is always improve in years to come.

Streaming technology is limited primarily by bandwidth and compression quality, so the playback quality depends on the network bandwidth. So when solutions for these issues have been found, streaming media technology has the potential to dominate media distribution and as a consequence replace the current broadcast TV distribution, cable TV distribution and physical video rental. The developed system showed clearly that interruptions to service were more disturbing than a lowering of sound quality.

A good streaming protocol should be able to deal with all different types of Internet congestion and should be close to the upper bound of the average throughput. Some TCP implementations are actually pretty good getting close to the upper bound under different networking conditions. The upper bound is the average bandwidth between a server and a player, regardless of any latencies. Streaming media will be accepted as a mature technology if a streamed media at DSL rates can be viewed without any visual degradation despite intense network congestion.
7. Acknowledgements
This work was implemented in Intelligent Systems and Robotics Laboratory – Institute of Advanced Technology (ITMA), so we would like to thank the staff members of this Lab especially Ms. ROSIAH OSMAN – Science Officer and Mr. MUHAMAD SAUFII KASSIM for their assistance and for provided us by all facilities that we needed to complete this work.

References
[7] Lammi, J., 2001, Internet streaming media platform, University of Technology, Department of Information Technology, Software Systems Laboratory