CONTROL MECHANISMS AND RECOVERY TECHNIQUES FOR REAL TIME DATA TRANSMISSION OVER THE INTERNET

Venkata Krishna Rao Battula, B.E.

Problem in Lieu of Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2002

APPROVED:

Dr. Roy Tom Jacob, Major Professor, Department Of Computer Science
C. Neal Tate, Dean of the Robert B. Toulouse School of Graduate Studies

ABSTRACT

Battula, Venkata Krishna Rao, Control mechanisms and recovery techniques for real-time data transmission over the Internet. Master of Science (Computer Science), August 2002, 31 pp., 13 illustrations, 21 titles in reference list.

Streaming multimedia content with UDP has become popular over distributed systems such as an Internet. This may encounter many losses due to dropped packets or late arrivals at destination since UDP can only provide best effort delivery. Even UDP doesn't have any self-recovery mechanism from congestion collapse or bursty loss to inform sender of the data to adjust future transmission rate of data like in TCP. So there is a need to incorporate various control schemes like forward error control, interleaving, and congestion control and error concealment into real-time transmission to prevent from effect of losses. Loss can be repaired by retransmission if roundtrip delay is allowed, otherwise error concealment techniques will be used based on the type and amount of loss.

This paper implements the interleaving technique with packet spacing of varying interleaver block size for protecting real-time data from loss and its effect during transformation across the Internet. The packets are interleaved and maintain some time gap between two consecutive packets before being transmitted into the Internet. Thus loss of packets can be reduced from congestion and preventing loss of consecutive

packets of information when a burst of several packets are lost. Several experiments have been conducted with video data for analysis of proposed model.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

1. INTRODUCTION

Increase in bandwidth and computational speed encouraging people interest in real-time data transmissions over the Internet. In the Internet packet carrying real-time data may be dropped or arrive too late because Internet is packet switched and real-time data uses best effort delivery of User Datagram Protocol (UDP) [10-13]. From a connection endpoint's point view, the best effort service amounts to offer a channel with characteristics such as available bandwidth, delay, and loss that vary over time. Due to unreliable nature of UDP delivery, quality of media streams on receiver side will have negative impact by packet loss occurred during the transmission. In previous work, many people limited their research to study of reliable transport, but not to real-time delivery [1]. It is very important to do research in providing some control mechanisms to make less impact of packet loss on quality of real-time data, which uses UDP transport. In recent market voice-over-IP and other streaming media applications in the Internet become more widely deployed the limitations of best-effort IP transport are becoming apparent and its necessary to provide these real-time media applications with some protection from worst impact of packet loss [11].

Internet ability to support real-time applications becoming very difficult since these applications are unresponsive to the network congestion and places unfair demands on the network particularly in huge volumes of traffic. In real-time delivery routing decisions and recovery from network outrage bursts are purely local choices in many cases, which do not have to be communicated back the source of the data packets or any

of expecting receivers. Overwhelming cause of packet loss is congestion at routers; therefore we can correlate the bandwidth used and amount of loss experienced during the data delivery. In some cases data has to delivery through high latency channels with higher delays, which may cause variations in the delay (Called jitter). This is one of the concerns for developing loss tolerant applications with facility discarding too long delayed packets to meet the application timing requirements, leads to packet loss. This problem is considered more acute for interactive applications. Playout delay may be inserted to allow delayed packets in applications for which interactivity is unimportant [1-10].

In this paper, literature about different control mechanisms on server side like interleaving, and forward error correction (FEC) and recovery techniques like error concealment, and retransmission mechanisms is reviewed to incorporate good reliable nature to the UDP transport delivery. Control mechanisms are normally classified into two categories: receiver-based, and sender-based. Sender-based, the sender to first process input streams in a way that receiver can better reconstruct missing data. Based on different ways of processing input streams, these schemes further classified into those that add redundant data and those do not. There are several ways for sender to add redundancy, include sending duplicate packets, or sending past packets coded in lower bit rate along with current ones or sending error correction bits in voice packets. All redundant control methods need extra bandwidth or long end-to-end delays. There are some mechanisms that do not add redundancy but utilize inherent redundancies in source data streams. Interleaving is one typical method for non-redundant type, interleaves data units into distinct packets and reconstructs lost units by interpolation using their neighbor

units [14-3]. Error concealment techniques rely on producing a replacement for a lost packet, which is similar to the original packet. These may be initiated by the receiver of real-time data and do not require assistance from the sender, of use when sender-based recovery schemes fail to correct all loss, or when the sender of data is unable to participate in the recovery [1].

In this paper, main focus is on implementation of interleaving with addition of time spacing between packets before pumping them into Internet for delivery. This time gap provides good handling of congestion in the Internet and interleaving improves the quality of real-time data and better human interaction towards packet bursts. At the end several experiments and their results are being discussed for better control and handling of packet loss and to reduce it effects.

1.1 DEFINITION OF TERMS

Congestion. A router cannot reserve memory or communication resources in advance of receiving datagrams since IP is connectionless. As a result, routers can be overrun with traffic, a condition known as congestion. Congestion may arise for two entirely different reasons; first a high-speed computer may be able to generate traffic faster than a network can transfer it. Second, if many computers simultaneously need to send datagrams through a single router, the router can experience congestion [21].

Latency. A synonym for delay, is an expression of how much time it takes for a packet of data from one designated point to another. In some usages latency is measured by sending a packet that is returned to the sender and the round-trip time is considered the latency.

Bandwidth. The width of a band of electromagnetic frequencies, is used to mean how fast data flows on a given transmission path.

Codec. A codec is a single-input single-output processing component. It reads data for individual track, processes the data, output the results.

1.2 PROBLEM STATEMENT AND SIGNIFICANCE

Most of the real-time applications, which are sensitive to network delay, use UDP as their transport protocol. Even though UDP is not a reliable protocol as TCP, UDP has its own properties to fit best for these applications. Few reasons to choose UDP than TCP are [15-16]

- **No connection establishment.** TCP uses a three-way handshake to establish a connection before it starts to transfer data. UDP just blasts away sending data without any formal preliminaries to setup any connection. Thus UDP does not introduce any "delay" to establish a connection, which is very key factor for interactive real-time applications.
- **No connection state.** TCP maintains connection state in the end systems during the transmission. This connection state includes receive and send buffers, congestion control parameters, and sequence and acknowledgment number parameters. On the other hand, UDP does not maintain connection state and does not track any of these parameters. For this reason, a server devoted to a particular application can typically support many more active clients when the application runs over UDP rather than TCP.

- **Small segment header overhead.** The TCP segment has 20 bytes of header overhead in its every segment, whereas UDP only has 8 bytes of overhead.
- **Unregulated send rate.** TCP has a congestion control mechanism that throttles the sender when one or more links between sender and receiver becomes excessively congested. This throttling can have a severe impact on real-time applications, which can tolerate some packet loss but require a minimum send rate. On the other hand, the speed at which UDP sends data is only constrained by the rate at which the application generates data, the capabilities of the source (CPU, clock rate, etc.) and the access bandwidth to the Internet. However, that the receiving host does not necessarily receive all the data - when the network is congested, a significant fraction of the UDP-transmitted data could be lost due to router buffer overflow. Thus, the receive rate is limited by network congestion even if the sending rate is not constrained.
- In UDP transport layer won't cause retransmissions for lost packets.
- Internet checksum can verify UDP header and data payload for validation.

UDP have some disadvantages over TCP transport. TCP has congestion control mechanism (slow-start designed and implemented by Van Jacobson) is there to let the hosts to adopt in cases of severe congestion. UDP is efficient for point-to-point (a user to user) applications; UDP is never designed to deliver large amounts of data from a central location. If an uncontrolled number of users start retrieving data on the Internet using UDP, it will cause severe congestion on links near the server, and the congestion will not ease as UDP does not have built-in congestion control. Quality of real-time data is determined by type of compression used and amount of data lost during the transmission.

Packets do get lost during the transmission over the Internet therefore it degrade quality of UDP based real-time applications [17]. It is understood that there will be some packet loss and congestion in networks while the usage of UDP for real-time applications. So before design some techniques to repair or control a real-time media streams subject to packet loss or congestion in networks, it is important to have some knowledge of the loss characteristics which are likely to be encountered. Number of experiments were conducted on loss characteristics, it was clear that some receivers will experience inevitable packet loss [18-19-13]. In majority of the experiments losses are single packets. Burst of losses occur less frequent that in case of transient congestion. So in this paper main focus is given to control and repair single packet loss since this is far the most frequent occurrence in real-time applications. Significance of error control mechanisms to provide control on packet loss or repair to the packet loss is very important since losses in real-time audio/video applications will have a great impact to an human eye perception.

2. LITERATURE STUDY ON CONTROL AND RECOVERY TECHNIQUES

The broad range of applicable techniques both sender-driven and receiver-based, have been implemented in wide range of real time applications, which give them operational experience and recovery from packet loss. Sender-driven and receiver-based recovery techniques are complementary and applications should use these to attain best performance. The techniques discussed in this section are not necessarily generalize to conferencing applications, majority of these are applicable to unicast IP [1].

2.1 SENDER-DRIVEN REPAIR

These techniques slit into different classes, which have subclass division depending on the nature (refer figure 1).

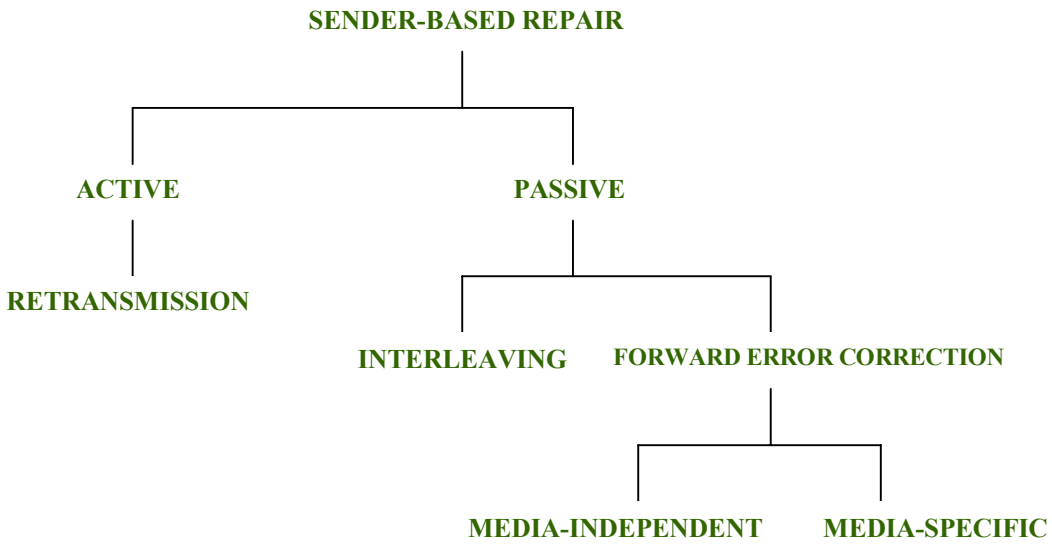

Figure1. Taxonomy of sender-based repair techniques

2.1.1 FORWARD ERROR CORRECTION (FEC)

Schemes rely on the addition of repair data along a data stream, from which lost packets may be recovered are named as forward error correction techniques. Repair data in these techniques are divided into two classes, those that are independent of data and those, which use knowledge of the stream data. Error control mechanism using forward error correction is implemented and evaluated in [2] to minimize the loss rate and maintain the good quality of transmitting multimedia.

Media-Independent FEC. Number of media-specific schemes proposed for use with streamed media. In these techniques addition of redundant data, which is transmitted in separate packets, will be used to a media stream. FEC techniques are described as loss detecting and/ or loss correcting in tradition. In media streaming, loss detection on receiver side is provided by the sequence numbers in Real time protocol (RTP) packets. Redundant data in these FEC schemes typically calculated using the mathematics of finite fields. The simplest of finite field is GF (2) where addition is just exclusive-OR operation. Basic FEC schemes transmit k data packets with n-k parity packets allowing the reconstruction of the original data from any k of the n transmitted packets [13]. In this class FEC using block, or algebraic, codes to produce additional packets which will be useful for correction of losses. There are two main cases which were discussed in [1], Parity coding and Reed-Solomon coding. Parity coding, the exclusive-or (XOR) operation is applied across many packet groups to generate corresponding parity packets. Reed-Solomon codes are known for their excellent error correcting properties, and in particular their resilience against burst losses. Parity-based FEC techniques have a significant advantage that they are media independent, provide exact repair for lost packets, and the computation required to derive the error correction packets is small and simple to implement. In addition processing requirements for these schemes are relatively light, especially when compared with some media-specific FEC schemes, which use very low bandwidth but very high complexity encoding. Disadvantages of these schemes are that the codings have higher latency, increased bandwidth, and difficult decoder implementation than the media-specific [13-1]. Reed-Solomon codes typically come at

expense of increased latency but dependent on observed loss patterns. These codes may give improved performance compared to parity-based FEC.

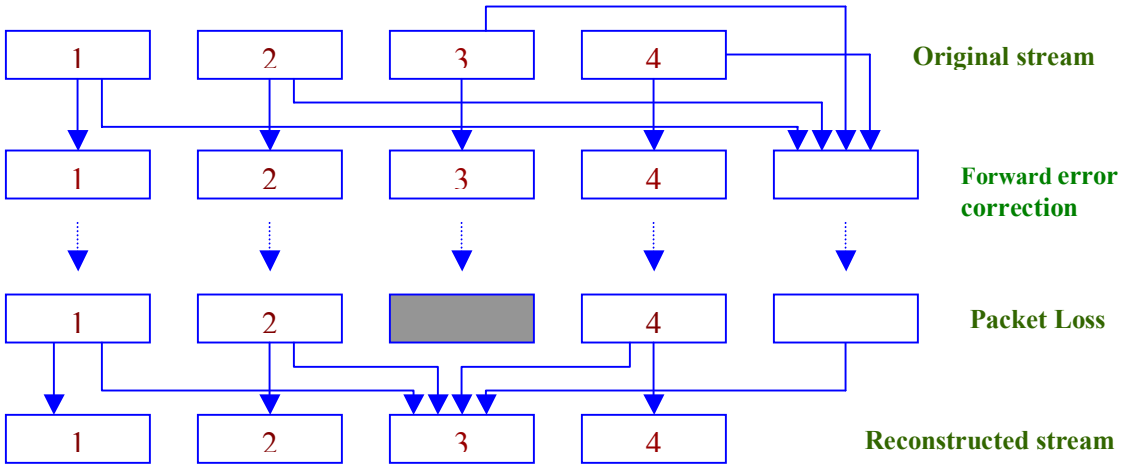

Figure2. Repair using parity FEC

Media-Specific FEC. Basis of media-specific FEC is to employ knowledge of a media compression to achieve more efficient repair of a stream than other ways. In this scheme also redundant data is added to repair lost packet in media stream. Here redundant data is some information added to media stream which is not required in the absence of packet loss, but which can be used to recover from that loss. If the units of media data are packets, or multiple units are included in a packet, it is logical to use the unit as level of redundancy and to send duplicate units. If a packet is lost it can cover that loss by getting the same unit from another packet, which it contains. Recording redundant copy of a unit can save significant bandwidth at the expense of additional computational complexity and approximate repair. If a media units span multiple packets, it is sensible to include

redundancy directly within the output of codec. Proposed RTP payload for H.263 includes multiple copies of key portions of the stream separated to avoid the problems of packet loss of key portions. Advantage of this approach is efficiency since the codec designer knows exactly which portions of the stream are important to protect from losses and also its low complexity. Use of media-specific FEC has the advantage of low-latency with only single packet delay being added. This is suitable for interactive applications, where large end-to-end delay can be tolerated. In a unidirectional and non-interactive environment it is possible to delay sending redundant data to achieve improved performance in the presence of burst losses at the expense of additional latency [13-1].

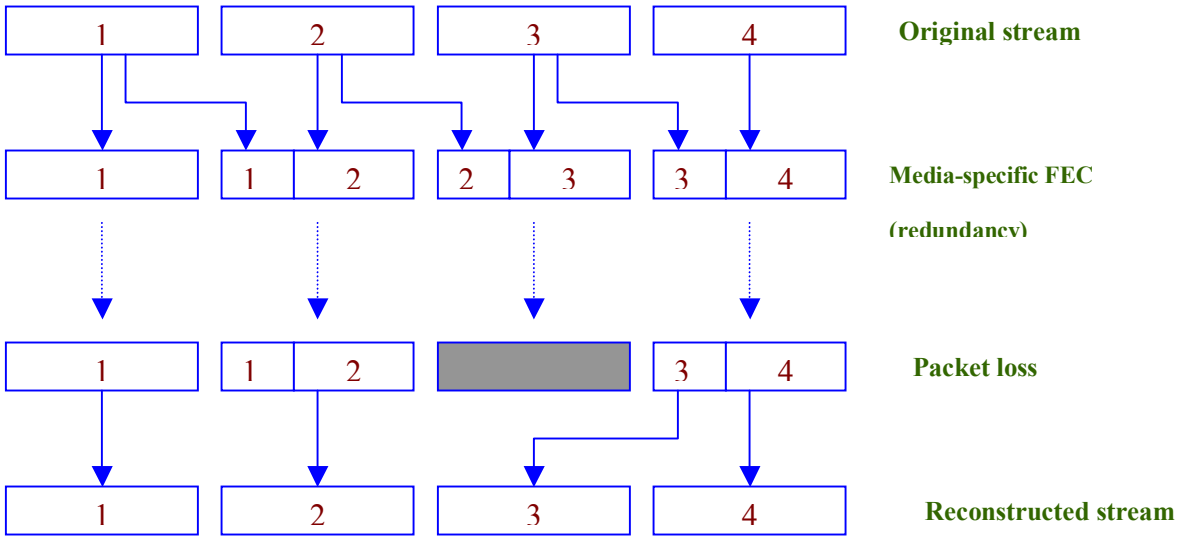

Figure3. Repair using media-specific FEC

FEC provides a promising solution to the problem to correct errors at the end receiver with out waiting for the retransmission packets. FEC schemes are useful to

transmit the date reliably without any acknowledgements used in TCP as described in [2]. Also experiments results from [2] showing that using FEC over UDP for real-time data transmission can reduce packet loss and provide low latency and jitter than TCP. Limitations of FEC schemes, additional repair data along with original data stream may cause more usage of available bandwidth and which may worsen the network congestion leads to more data loss in networks. So some congestion control mechanism should be expected to incorporate with FEC schemes or the schemes, which will have congestion control, must be used for real-time data transmission to protect from packet loss over UDP [1].

2.1.2 INTERLEAVING

Unit size is smaller than the packet size and end-to-end delay is unimportant, interleaving is a useful technique for reducing effect of loss. Units are resequenced for reducing the effects of loss, before transmission so that originally adjacent units are separated by a guaranteed distance in the transmitted stream and returned to their original order at the receiver. This technique is known as interleaving which will disperses the effect of packet losses. The loss of single packet from an interleaved stream results in multiple small gaps in the reconstructed stream, where as the single large gap, which would occur in uninterleaved stream. Spreading of the loss is important because of the small gaps wont effect much the quality of data transferred at the receiver and error concealment techniques can be better correction for small gaps [1][3]. In [3] two-way interleaving and reconstruction was discussed, where sender in the new algorithm transforms an input stream according to the interpolation method used at receiver and the predicted loss

behavior, in order to enable better reconstruction quality. Layer Coding for UDP video packets, separating important information and key video data as base layers and less important data as enhancement layers was discussed in [4]. Base and enhancement data layers are scrambled and interleaved before transmission, so that consecutive blocks of data can be spread as far as possible to prevent loss of consecutive and important data from burst of several base packets loss. Even though interleaving technique increases the latency for interactive applications, it performs well for non-interactive use and does not increase the bandwidth requirements of a stream.

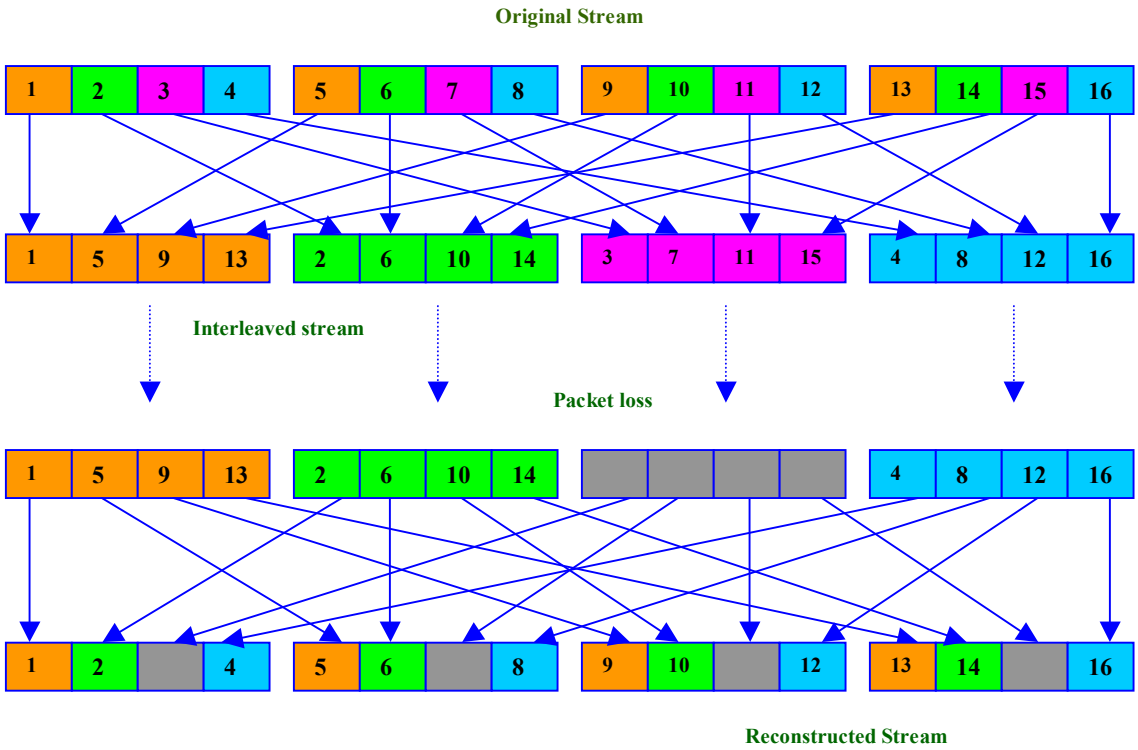

Figure4. Interleaving units across multiple packets

In figure4, example with 4 units in each packet described for interleaving. After interleaving on sender side the packets will be, 1st packet contains 1,5,9,13; the 2nd packet contains 2,6,10,14; and so on. It can be observed that loss of a single packet from

interleaved stream result in multiple small gaps in the reconstructed stream instead large gap, which would occur in a non-interleaved stream. It is very easy to reconstruct a stream with such loss patterns although this is clearly media and codec dependent. The size of the gaps is dependent on the degree of interleaving used can be made arbitrarily small at expense of additional latency [13]. In this paper implementation of interleaving is presented with results and their description in later sections. Disadvantage of interleaving is that it increases latency. It limits this technique for interactive applications and works well for non-interactive applications. The advantage of interleaving is that it does not increase the bandwidth requirements of a steam.

2.1.3 RETRANSMISSION

In general retransmission based recovery has been inappropriate for continuous media applications since it requires at least one additional round-trip delay to recover lost packets. But retransmission recovery is feasible where roundtrip delay is small, in particular if a playout buffer is used to increase the time available for recovery. Retransmission is still an attractive because of modest bandwidth and less processing costs when compared to FEC like techniques. Playout buffer is an important component of any retransmission scheme in a latency-constrained environment and also costs associated with playout buffer are small. The problem of interaction between retransmission and congestion can be tackle in one way is to set aside some bandwidth for retransmissions. During the time of congestion the application reduces its rate enough so that new data plus restransmissions do not exceed the reservation of bandwidth. Retransmission can be scaled very well in multicast environments where lost data can be

recovered locally from other receivers in multicast group [5][1]. Although retransmission based error control not suited for all continuous media applications, still an attractive technique regarding low-cost solution and remains a serious solution for continuous media error control. Retransmission is used when receivers experiencing burst losses, and willing to accept the additional latency to repair and recover the losses. In order to reduce overhead of retransmission, the retransmitted units may be piggybacked onto the ongoing transmission, using a payload format. This allows for the retransmission to be recoded in a different format to further reduce the bandwidth overhead [13].

2.2 RECEIVER-BASED REPAIR (ERROR CONCEALMENT)

These techniques are initiated by receiver for error concealment of a data stream with out assistance of sender. Generally these are useful when sender based recovery fail to correct all loss, or when sender unable to participate in recovery process. Error concealment techniques rely on producing a replacement for a lost packet, which is similar to original one, was discussed for audio streams in [1]. Taxonomy of receiver-based techniques is given in figure 5.

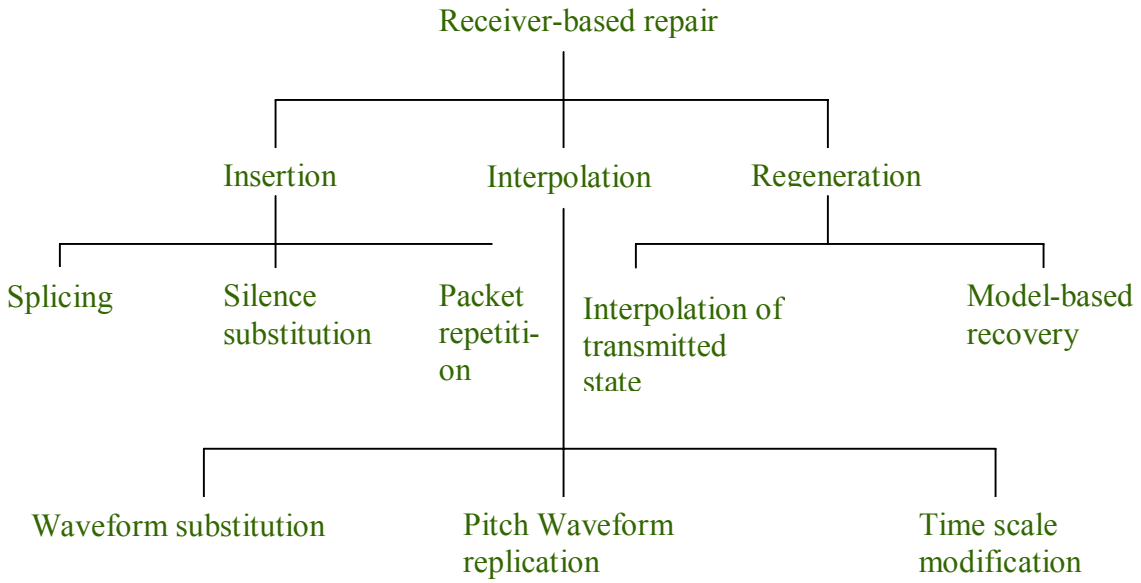

Figure5. **Taxonomy of error concealment techniques for audio streams.**

Silence Substitution, fill-in silence in the gap left by lost packet to maintain the timing relationship between the surrounding packets. This is only effective for short length packets and low loss rates.

Noise Substitution, some back ground noise is inserted for audio data loss to make up the missing component.

Repetition, lost units can be replaced with copies of unit immediately arrived before the loss. This is good trade off between poorly performing insertion based concealment techniques and complex interpolation based and regenerative concealment methods.

Interpolation-Based Repair, Various interpolations based repair techniques were discussed for audio data input in [1], wave substitution, and pitch waveform replication, time scale modification.

Video error concealment techniques. A single bit error during the transmission of compressed encoded video stream can cause severe degradation to the picture quality due to spatial and temporal error propagation. If error control techniques like FEC fail to stop errors then decoder on receiver side applies error concealment techniques to estimate the lost data by taking advantage of the spatial and temporal redundancy inherent in the adjacent error-free frames. Error concealment can be done only if error detection algorithm at the receiver side marks out errors. Spatial error concealment techniques are used for I-pictures for which no motion information exists. These techniques make use of spatial similarity in a picture. Spatial techniques are iterative and computationally intensive. This scheme utilizes only macro blocks that are located spatially above and below the damaged macro block. This scheme works well if the error is limited to one slice and that the adjacent rows are intact. Temporal error concealment, technique by which errors in p-pictures (predictive coded using the previous frame) are concealed. Macro blocks in P-pictures are coded by locating the best matching block in the previous frame, using which a displacement vector and transform coded difference signal are transmitted. Simple temporal error concealment technique is to copy the corresponding macro blocks from the previous frame. This technique works well if the damaged slice includes only one row of macro blocks [6-20].

Since the effectiveness of concealment techniques depends on the amount and correct interpretation of received data, concealment becomes much harder with bursty loss [5]. Final technique for error resilience of any multimedia data, error concealment is used to estimate the lost data by taking advantage of the spatial and temporal inherent in the adjacent error free macro blocks or frames as described in [6].

2.3 CONGESTION CONTROL

In present Internet all end-systems are expected to react to congestion by adjusting their transmission rates, not only to avoid congestion collapse but also to keep the bandwidth usage high. Also interprotocol fairness is important, rate adjustment should result in a fair share of bandwidth for all coexist flows along the same path. The main idea of separating congestion control from error control was discussed in [8], since congestion control is application specific. In RAP [8] protocol architecture is built to control congestion and to find loss detection. Adjusting quality with layered coding, quality control module transmitted streams based on the rate specified by RAP module. Quality control tries to deliver the maximum number of layers that can be fit into available bandwidth. Rate adaptation happens on time scale of round trip times, layers will be added and dropped based on slower time scale by using receiver buffering to accommodate mismatch between transmission and consumption rates. Buffering at client side also provides the selective retransmission as determined by quality control. Server including retransmission shouldn't exceed the bandwidth specified by RAP. Congestion avoidance discussed in [7], the network must signal the transport end points to inform about congestion. End points must have a policy that decreased the utilization of network of the signal received and increases if the signal isn't received. Window based approach is also discussed along with rate based approach in [9] to provide end-to-end TCP-friendly congestion control. Window based approach is derived from additive increase multiplicative decrease (AIMD) rule by adjusting the window size in response to the acknowledgements from receiver or packet loss detection.

3. INTERLEAVING PROCESS AND ITS IMPLEMENTATION

Interleaver is a device, which permutes the order of sequenced data packets. Device that reconstructs the original order is called deinterleaver. Interleaver is employed into transmission process whenever random distribution of errors needed in reception. After this interleaving process burst of losses will be distributed into isolated gaps. A block interleaver is an interleaver where the permutation function repeats periodically. If matrix interleaver block is of size *n* x *m* represent the *mn* successive real time data packets that are sorted in a buffer prior to transmission as described in figure6.

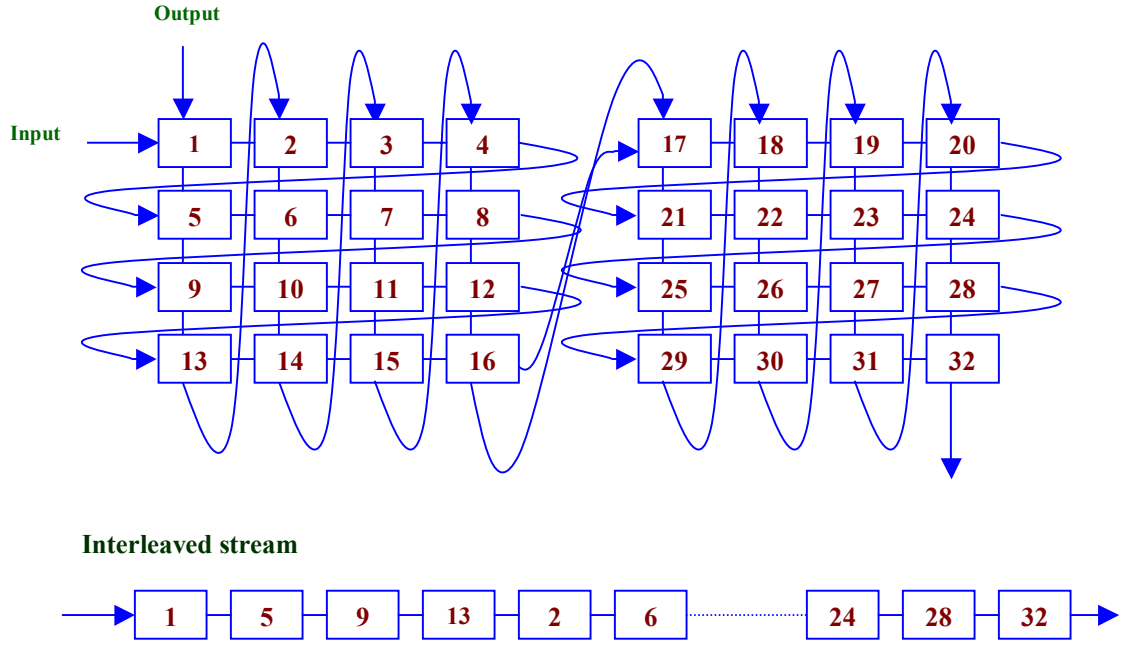

Figure6. Server Side construction of packets for interleaving with 4x4 matrix interleaver

In this paper interleaving process is adopted from [11] for implementation. Data packets are read into the matrix by rows starting from "input" and read out from by columns starting from "output". To make continuous interleaving two matrices are needed as described in [11] with packets being feed into the one matrix while reading the packets out from the other. Considerable delay will be there in the interleaver with out put of packets from the buffer matrix being delayed until all packets have been read in. Rearrangement of packets by the interleaver in a way if m or fewer symbols are lost from a block each original group of n packets after deinterleaving will contain at most one loss.

As shown in figure4, a burst of consecutive loss in an interleaved stream will result in multiple gaps in the reconstructed stream. Loss of a single packet will have a large effect on intelligibility of real-time data. If the spreading of losses is done by interleaving small parts of loss will spread among different packets resulting in improved perceived quality for a given loss rate. Also error concealment techniques perform significantly better for small gaps or losses [11].

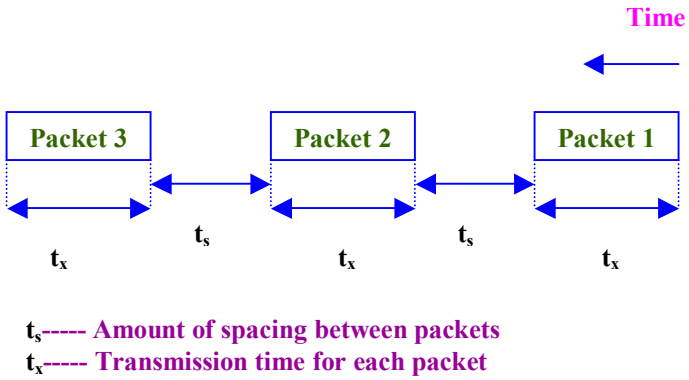

Figure7. Packet Spacing

In implementation of this paper for interleaving, packet spacing mechanism is added to avoid congestion, which may occur in networks as, described in [10]. Packet spacing, the delay introduced between two consecutive packets as in figure7. By this spacing in between packets, bursts can be spaced out, resulting in fewer packet drops at intermediate routers and potentially higher throughput at the end host. Packet spacing is smoother and work well than equation based congestion control to adapt the sending rate as needed based on available network resources.

Reconstruction of interleaved packets on receiver side will be done as show in figure8. Upon arrival of interleaved packets on receiver side it will feed them into similar matrices similar to sender side interleavers by row wise. Once the first buffer matrix is full, out put process of packets will starts by column wise, which will be the original order of the streams. These reconstructed streams will be feed as input to codec.

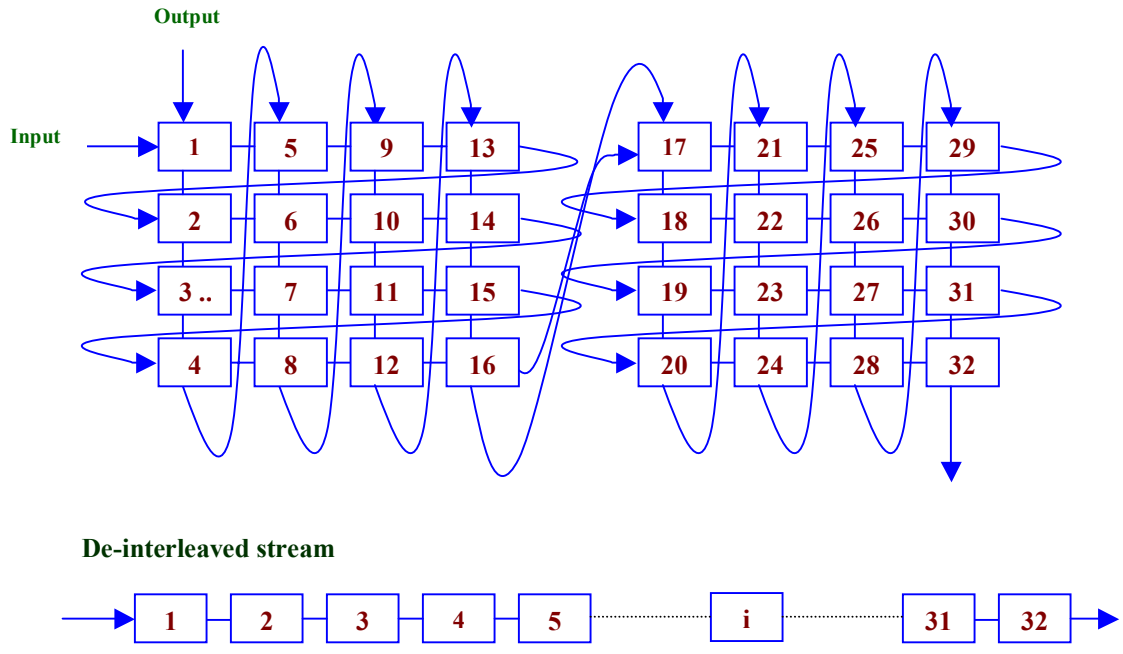

Figure8. Client Side reconstruction of interleaved packets to get the original stream with 4x4 matrix interleaver

3.1 IMPLEMENTATION

This problem implementation supports MPEG format movie clips only. Necessary changes can be made to support other type of movie formats also. This paper has three main modules for the problem implementation.

1. Server to handle requests from the clients to send video packets across the network.
2. Decoder to parse the movie file and extract buffers to send across the network.
3. Client to accept the video buffers and feed to a player.

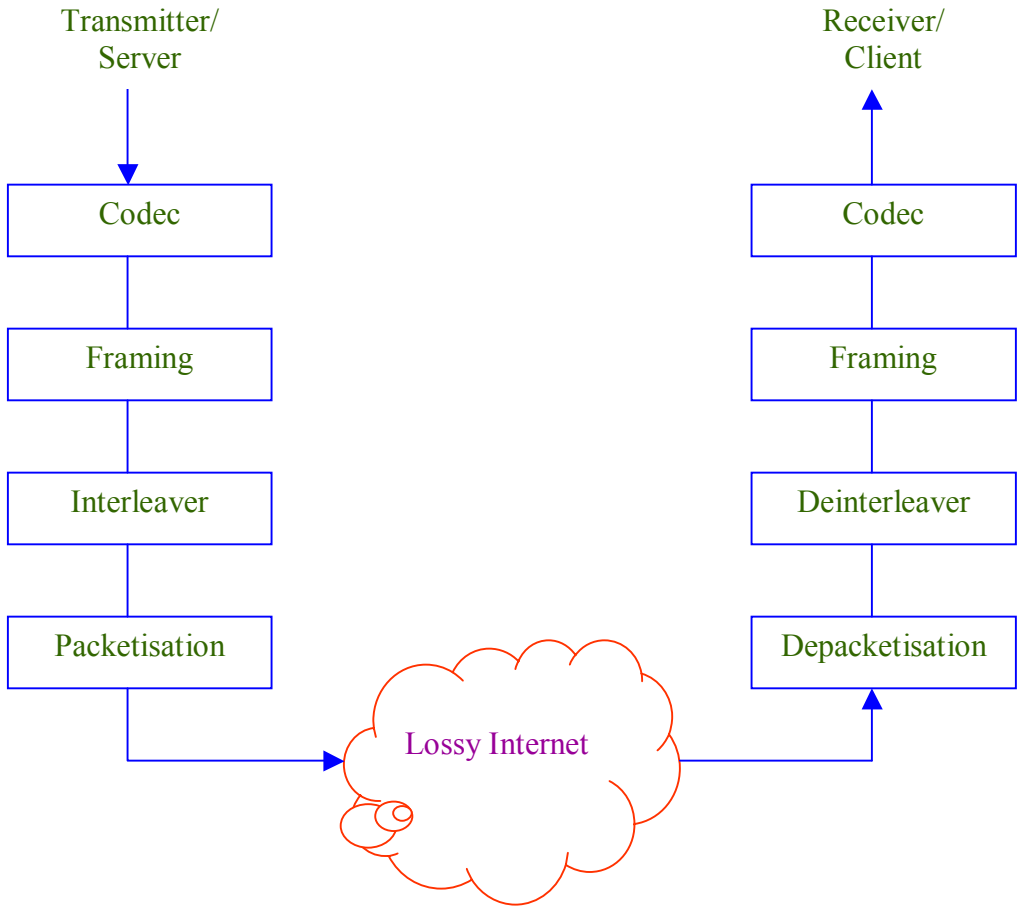

Figure9. Transmission process with interleaving [11]

Server side. Consists of a server waiting on port 8089 to accept connections from the clients, decoder which parses the movie file, a video thread to transfer data to the client. Video buffers are converted into a serialized from and sent across the network.
Java's JMF v 2.1.1 toolkit is used for decoding and extracting buffers from the movie file.

Functioning of the decoder. The decoder takes an input URL of the movie clip. Create a data source for the given media locator. Given the data source, create a processor to extract the video buffers.

Configure the processor and add a controller listener to control the processor. Get the raw output from the processor by setting the content descriptor to RAW. Realize the processor to initiate decoding of the video. Create a data source handler and add data sink listener to catch the processed video buffers. Pre-fetch the processor and start the processor for data initiation. The DataSourceHandler class implements a data sink to which the processed video data is sent. It contains loop threads to pull data from a PullBufferDataSource. There is one thread per each PullSourceStream.
Two buffers are used to store the retrieved video buffers from the video clip in form of an m by n matrix. Different buffer sizes are used to measure the performance of the decoder. One buffer is used to store to the extracted video buffers and other buffer is used to transfer the already stored buffers across the network. Buffers are used in the data flow path to store the retrieved buffer information and data. A server thread is called when the buffer is full, to send the buffered packets across the network

The server thread converts the buffer into serialized form (Vector) to transport across the network. The buffer object is broken down and stored as a vector that is transported across the network. The buffer is reconstructed at the client side from the vector. The buffers are interleaved before sent across the network. The interleaver used is an m by n matrix. The video buffers are stored in the matrix in rows consecutively and read out by columns. Continuous interleaving is done as we use two matrix buffers with data being written into one matrix buffer while they are read out of the other matrix buffer. The rearrangement of the video buffers by the interleaver is such that if m or fewer buffer packets are lost from a block, each original group of n buffer packets after de-interleaving will contain at most one loss. Group of interleaved packets sent across the network are time spaced i.e. between sending of each set of interleaved packets there is some time gap.

Client Side. The client side reads the interleaved video packets from the network de-interleave them, write it to a temporary file and feed it to the player to play the video. The client program consists of a PullBufferDataSource, which supplies the packets from the temporary file to the player. The packets are read from the network, de-serialized and de-interleaved, which are then fed to the data sink through which the player reads the video packets and displays them in an AWT frame.

There are basically two ways to access the individual frames and feed to the player.

1. Get the data from the output Data Source of the Processor.
2. Using a pass-through plug-in codec as a callback when individual frames are being processed.

In this program the second approach is used. Build the pass-through codec. Create a processor; get the Track Controls from the processor. Set the codec on the video track: TrackControl.setCodecChain (codec []). This way, the codec's process call will be the "callback" whenever a video frame goes through the plug-in.

process sender

1. ***while (true) do***
2. ***Create the processor, decode the video file and buffer the packets***
3. ***Interleave the buffered packets***
4. ***Serialize and Packetize each buffered packet***
5. ***Send the packets to the receiver.***
6. ***end-while.***

end.

process receiver

1. ***while (true) do***
2. ***De-serialize each packet***
3. ***De-interleave all the packets received.***
4. ***Write the packets to a data sink.***
5. ***Feed the packets through a data source to the player created.***
6. ***end-while.***

end.

Figure10. Pseudo-code showing the steps carried out by the sender and receiver using interleaving with packet spacing.

4. EXPERIMENTAL RESULTS AND DISCUSSION

Experiments have conducted on the proposed model in this paper; one experiment is conducted on various buffer sizes (interleaver block sizes) at different times of the day. Two graphs (figure11, figure12) are presented in this paper, which shows the optimized buffer size range of 12-20. This range of buffer sizes better and consistent transmission time was come out for the proposed model. Second experiment was conducted with varying packet spacing for throughput time. In figure13, graph shows that optimized and consistent range of packet spacing 950 micro seconds-2000micro seconds. If the buffer size is too small there is change that loosing of two nearest packets even in case of interleaving, and if the buffer size too big it takes long delays for buffering so, from experiments conducted, optimized size range 12-20 is found. Packet spacing provides good time gap to slow the traffic reasonably in the Internet to avoid congestion and gives good throughput times. So it is realized that the proposed model have good performance towards the throughput times provides good removal of jitter with packet spacing.

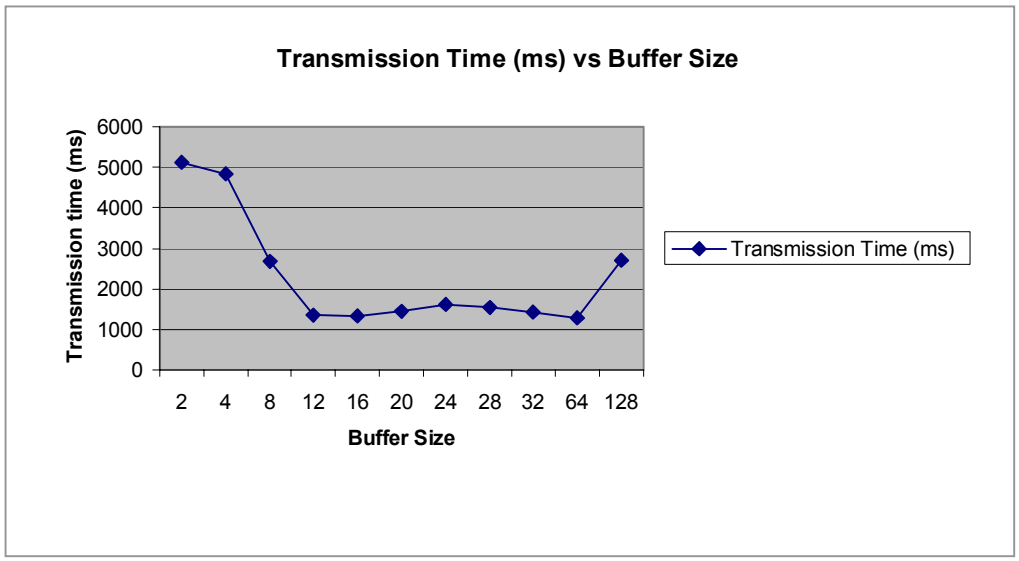

Figure11. Experiment done on proposed model on 06/15/02 at 4a.m for diff buffer sizes for transmission time

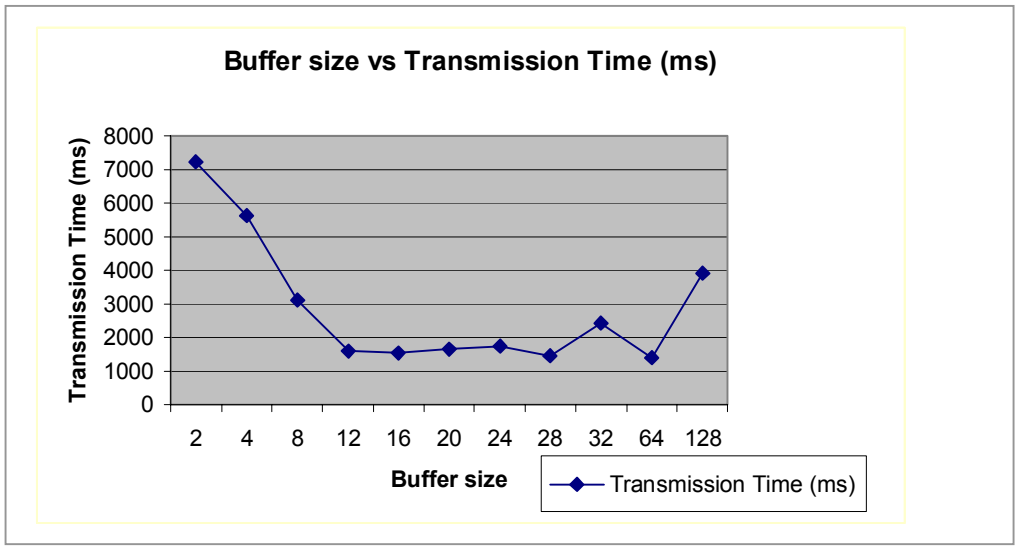

Figure12. Experiment done on proposed model on 06/15/02 at 12p.m for diff buffer sizes for transmission time

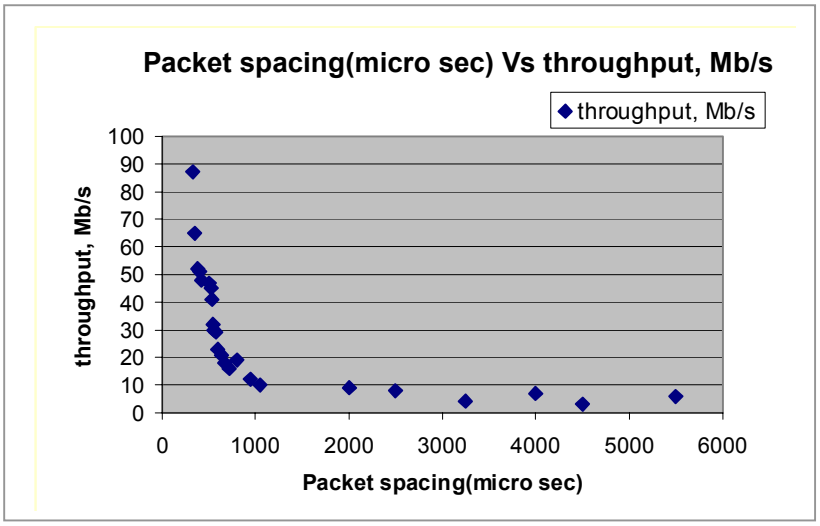

Figure13. Throughput time with packet spacing

5. RECOMMENDATIONS AND FUTURE DEVELOPMENTS

If desired scenario is a non-interactive unidirectional transmission, like radio or television broadcast, latency is considerably less important than reception quality. In this kind of applications interleaving is appropriate. If approximate repair is acceptable, interleaving is clearly preferred since it does not increase the bandwidth of a stream. In this paper, focus is given to server to client transmission; extension of this project will be considered for multicast applications like videoconference and live streaming of video audio presentations. This paper doesn't discuss implementation details of any error recovery techniques in the proposed system architecture. Future extension of this work can be developed with error concealment and repair provision at receiver side for small

bursts of packet loss. Security is very important issue for the real-time data transmission over the Internet these days. So this project can be designed for securities issues with secure communication by encryption and authentication algorithms incorporated on server and client side.

6. CONCLUSION

In this project vast literature study has been done about various control mechanisms explained in this paper. Real-time applications like video and audio used UDP best effort delivery, where loss may occur during the transmission. To avoid those losses some mechanism were discussed in this paper, which were sender-driven and incorporating some congestion feed back rate adjustment mechanisms and receiver based. Error concealment or recovery of error or lost packets some error concealment techniques were discussed. In this paper experiments are conducted on implementation of interleaving with packet spacing for realizable throughput is found even when sender rate is less. In today's multimedia applications blast packets as fast as possible. Slowing down the pumping of packets, by packet spacing, into the Internet, congestion is alleviated at the intermediate routers; this results in a net increase in throughput. Interleaving provides an acceptable visual quality, shorter burst loss lengths and fewer isolated losses, which can make error concealment job easy. Multimedia data, both audio and video, don't require complete reliability in delivery across the Internet, it can tolerate small errors. Loss of a packet in video would flicker on the screen, for audio, it would just be an imperceptible silence. The buffering (two matrices) in this paper

implementation at both sender and receiver side provides the video play smoother, also minimizes jitter. From experiment results, this implementation requires a reasonable size buffer (not too small or not too big) for interleaving and deinterleaving resulting in delay, which is utilized to remove jitter.

REFERENCES

1. Colin Perkins, Orion Hodson, and Vicky Hardman; University College London, A Survey of Packet Loss Recovery Techniques for Streaming Audio. http://www.east.isi.edu/~csp/publications/IEEE-Network-1998.pdf

2. Ray Fang, Dan Schonfeld, Rashid Ansari, Jason Leigh; Forward Error Correction For Multimedia and Teleimmersion Data Streams. http://www.euro-link.org/images/PDF/RayFangFEC1999.pdf

3. Benjamin W. Wah, Fellow, IEEE, and Dong Lin; Transformation-Based Reconstruction for Real-Time Voice Transmissions Over the Internet. http://manip.crhc.uiuc.edu/Wah/papers/Dirs/J68/J68.pdf

4. Chin, Suk Kim, Rogers, Glynn; Layered Coding for Interleaving and Scrambling Of UDP Video Packets. http://www.diee.unica.it/pv2000/proceedings/papers/51.pdf

5. Christos Papadopulos, Gurudatta M. Parulkar; Retransmission-Based Error Control for Continuous Media Applications. http://netweb.usc.edu/cs551f00/papers/nossdav96.pdf

6. Desmond yeo (University of Michigan), Digital Video Spatial Error Concealment Techniques. www.engin.umd.umich.edu/~what/DVP.doc

7. Van Jacobson, Michael J. Karels; Congestion Avoidance and Control. http://www-nrg.ee.lbl.gov/papers/congavoid.pdf

8. Reza Rejaie, Mark Handley, Deborah Estrin, Marina Del Rey; RAP: An End-to-end Rate-based Congestion Control Mechanism for Real-time streams in the Internet. http://www.cse.ogi.edu/~wuchi/580/Readings/Papers/proxy.rejaie98rap.pdf

9. Young-Gook Kim, Yon Jun Chung, JongWon Kim and C.C. Jay Kuo; An End-to-end Congestion Control Mechanism for Internet Video.

10. Apu Kapadia, Annette Feng, and Wu-chun Feng; The Effect of Inter-Packet Spacing on the Delivery of Multimedia Content. http://public.lanl.gov/radiant/website/pubs/rapid/icdcs01.pdf

11. Colins Perkins, Jon Crowcroft, Department of Computer Science, University College London, Effects of Interleaving on RTP Header Compression.
http://www.east.isi.edu/~csp/publications/InfocomInterleaving.pdf

12. G.A. Miller and J.C.R. Licklider, The Intelligibility of interrupted speech, Journal of the Acoustical Society of America, vol. 22, no. 2, pp.167-173, 1950.

13. C. Perkins, O. Hodson, University College London, Options for Repair of Streaming Media, Network Group, RFC 2354, June 1998.
http://www.faqs.org/rfcs/rfc2354.html

14. V.Hardman, M.A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the Internet" in Proc. Int. Networking Conf. June 1995, pp 171-178.

15. J.Postel, "User Datagram Protocol," RFC 768, August 1980.
http://www.faqs.org/rfcs/rfc768.html

16. Lars-Ake Larzon, Mikael Degermark, Stephen Pink Lulea University of Technology, Lulea, Swedish Institute of Computer Science, Stockholm; UDP Lite for Real Time Multimedia Applications.
http://www.hpl.hp.com/techreports/1999/HPL-IRI-1999-001.pdf

17. Kenneth Lee and Woong-kyo Suh 12/9/1996,University of Pennsylvania, Professor Keith Ross, Real-time audio on the Internet.
http://www.seas.upenn.edu/~ksl/Classes/TCOM500/InternetAudio/

18. J-C. Bolot and T. Turletti, Experience with Control Mechanisms for packet video in the Internet.
http://www.acm.org/sigcomm/ccr/archive/1998/jan98/ccr-9801-bolot.pdf

19. M. Handley, An Examination of Mbone performance. USC/ISI Research report: ISI/RR-97-450, April 1997.

20. Aravind Raman and Murali Babu, Ittiam Systems pvt Ltd, A Low Complexity Error Concealment scheme for MPEG-4 Coded Video Sequences, Annual Symposium on Multimedia communications and Signal processing, Nov22-24, 2001, Bangalore, India.
http://www.ittiam.com/pages/competency/video_mmcsp_2001.pdf

21. Douglas E. Comer, Internetworking with TCP/IP principles, protocols, and Architectures.