Video Transmission Over Varying Bandwidth Links

Dissertation

submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Laxmikant Patil

(Roll no. 04329019)

under the guidance of

Prof. Sridhar Iyer

Kanwal Rekhi School of Information Technology

Indian Institute of Technology Bombay

2006

Dissertation Approval Sheet

This is to certify that the dissertation entitled

Video Transmission Over Varying Bandwidth Links

by

Laxmikant Patil

(Roll no. 04329019)

is approved for the degree of **Master of Technology**.

Prof. Sridhar Iyer

(Supervisor)

Prof. Om Damani

(Internal Examiner)

Prof. Ashwin Gumaste

(Additional Internal Examiner)

Prof. Vishal Sharma

(Chairperson)

Date: _____

Place: _____

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

CERTIFICATE OF COURSE WORK

This is to certify that **Mr. Laxmikant Patil** was admitted to the candidacy of the M.Tech. Degree and has successfully completed all the courses required for the M.Tech. Programme. The details of the course work done are given below.

Sr.No.	Course No.	Course Name	Credits
		Semester 1 (Jul – Nov 2004)	
1.	IT601	Mobile Computing	6
2.	HS699	Communication and Presentation Skills (P/NP)	4
3.	IT605	Computer Networks	6
4.	IT619	IT Foundation Laboratory	10
5.	IT623	Foundation course of IT - Part II	6
6.	IT694	Seminar	4
		Semester 2 (Jan – Apr 2005)	
7.	CS686	Object Oriented Systems	6
8.	EE701	Introduction to MEMS (Institute Elective)	6
9.	IT610	Quality of Service in Networks	6
10.	IT620	New Trends in Information Technology	6
11.	IT680	Systems Lab.	6
		Semester 3 (Jul – Nov 2005)	
12.	CS601	Algorithms and Complexity (Audit)	6
13.	CS681	Performance Evaluation of Computer Systems and Networks	6
		Semester 4 (Jan – Apr 2006)	
14.	CS630	Approximation Algorithms	6
		M.Tech. Project	
15.	IT696	M.Tech. Project Stage - I (Jul 2005)	18
16.	IT697	M.Tech. Project Stage - II (Jan 2006)	30
17.	IT698	M.Tech. Project Stage - III (Jul 2006)	42

I.I.T. Bombay Dy. Registrar(Academic)

Dated:

Abstract

Internet today is a best-effort network, with time-varying bandwith characteristics. A system for multicasting video over the Internet has to deal with heterogeneity of the receiver's capability and/or requirements. So adaptive mechanisms are necessary. Conventionally multi-layered transmission of data is preferred to solve the problem of varying bandwidth in multimedia multicast application. In todays scenario majority of the flows are highly bandwidth consuming and are bursty in nature. Consequently we observe sudden changes in available bandwidth, leading to lot of variations in received video quality. Here our aim is to minimize these variations in video quality. In this thesis we propose traffic pattern based adaptive multimedia multicast (TPAMM) scheme.In this scheme, link bandwidth values are available for prediction window length. Bandwidth prediction model is refined periodically using feedback from receiver. TPAMM scheme also maximizes the minimum quality video playout.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

In recent years the multimedia applications are rapidly increasing, support for video transmission is thus becoming a basic requirement of network architectures. People have desired end-to-end visual communications for many years. However, real video communications over the Internet has not been used widely and successfully in our daily life. The current best-effort Internet does not offer any quality of service (QoS) guarantees. Hence good quality of video can not be sustained over time. This motivates us to come up with a scheme which provides consistent video quality to user.

1.1 Key terms

Following are some the key-terms that are frequently used in this document.

- Playout rate : The rate at which video is shown at client side
- Startup-delay : The duration of time client is ready to wait before the playout starts
- Transmission rate: This the rate at which data is transmitted over the link
- Base encoding rate: This is the rate at which raw video is encoded initially. This is the highest quality rate.

Existing approaches for multimedia transmission over internet try to start the playout as soon as possible i.e. these approaches try to minimize the startup delay. But our focus is application which can tolerate some initial startup delay. These are call *delay-tolerant applications*. Some of the example applications include

- Distance education program

- Movie streaming over internet
- Live Concert streaming can also be started with some intensionally introduced initial delay and we can utilize this delay to provide better video quality.

1.2 Startup latency

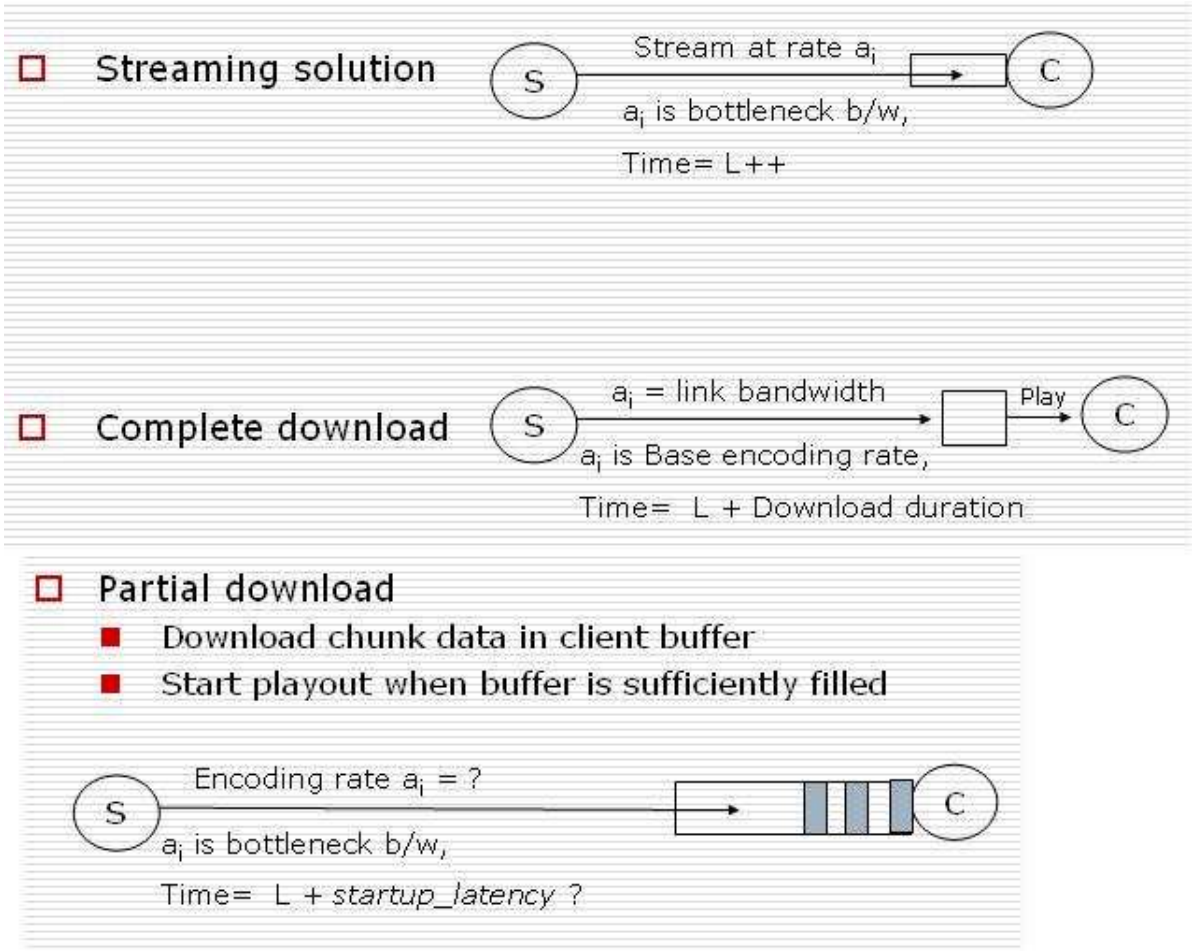

Figure 1.1: Three ways of data transmission

Suppose we want to transmit video file from Source to client, following are the 3 ways to do this. Refer 1.1

- *Streaming solution :* In this approach we can start the playout immediately but the rate at which data is transmitted is limited by bottleneck bandwidth on the path from source to client.

- *Complete download :* In this approach playout does not start until entire file is downloaded at client. Here we get best possible video quality but the startup delay is very high, it is equal to amount of time required to download entire video file.

- *Partial download :* In this approach we download data in chunks to client buffer. And when we have sufficient data to ensure continuous playout, we can start the playout at client. We do not need to wait for entire file to be downloaded. Also the encoding rate of data is not constrained by bottleneck bandwidth. Thus we have a trade-off between startup latency and encoding rate.

1.3 Need for Adaptive mechanisms

A system for multicasting video over the Internet has to deal with heterogeneity of the receiver's capability and/or requirements. Typically receivers and the path leading to them have different reception capacity. Even video displaying /rendering capabilities of client may be different e.g. one client watching video on desktop PC and other client watching same video on mobile phone, in such cases format conversions are necessary. In this work mainly focus on heterogeneity of the receivers in terms of bandwidth capabilities. So adaptive mechanisms are needed to address the problem of heterogeneous receivers.

Conventionally multi-layered transmission of data is preferred to solve the problem of varying bandwidth in multimedia multicast application. Data will be transmitted in layered fashion i.e. base layer and enhancement layers. Transmission rates are adjusted using layer-join and layer-leave experiments. When you have better bandwidth join more enhancement layer. If the bandwidth falls down, unsubscribe few of the enhancement layers.In todays scenario majority of the flows are highly bandwidth consuming and are bursty in nature. Consequently we observe sudden changes in available bandwidth, leading to lot of variations in received video quality. This variation in the video quality occur when layer-join and layer-leave experiment happen frequently. Our aim is to minimize these variations.

1.4 Problem Definition

Our objective is to utilize the startup latency to overcome the problem of link bandwidth variations and provide consistent quality client .

In this thesis we present an approach (TPAMM) Traffic Pattern based Adaptive Multimedia Multicasr. In this approach during transmission, we have a notion of available bandwidth using "bandwidth prediction model". We can build the initial model from the log files available at routers. This model is refined periodically using feedback from receiver. At receiver actual value of available bandwidth, packet loss are measured. And this feedback is given back to source and the prediction model is refined. TPAMM approach utilizes "startup latency" to overcome the problem of link bandwidth variation. Since we are having the prediction model, it is possible to make provisions for possible low bandwidth later on by pushing down more data to the client while we are having good-bandwidth. And we can consume this extra data when the available bandwidth is not good.

1.5 Thesis outline

In the chapter 2 we present related work on adative schemes for multimedia multicast. In chapter 3 we will explain the system architecture. In chapter 4 we describe our solution strategy with help of algorithm. In chapter 5 we discuss simulation and result. Finally chapter 6 outlines the future work and provide some concluding remarks.

Chapter 2

Literature Survey

2.1 Related Work

In this section we present existing approaches for video transmission over variable bandwidth links. A system for multicasting video over the Internet has to deal with heterogeneity of the receivers capability and/or requirements. Typically receivers and the path leading to them have different reception capacity. So adaptive mechanisms are needed to address the problem of heterogeneous receivers. Layering mechanism discussed in [1] [2] and transcoding mechanism discussed in [3] are important mechanisms to improve QoS for varying bandwidth links.

2.1.1 Multilayered approaches

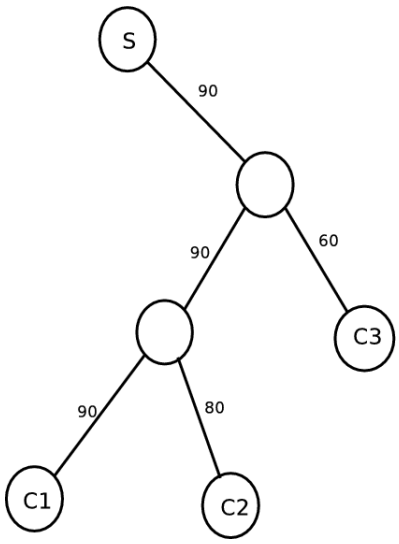

Figure 2.1: Multi-layering example

A multilayered video encoder compresses a raw video sequence into one or more streams, or layers, of different priority. Highest priority layer is called Base layer, this contains most important data of the video, other layers are called enhancement layers which contain further refinement of the base layer stream. When a network link gets congested, low priority packets are discarded. Given a network with support for priority-based packet discarding, the encoder may generate an enhancement layer of video for each unique multicast bandwidth constraint and thus ensuring all receivers obtain the quality of video proportionate with their available bandwidth. Consider the figure 2.1 here client-3 has 60kbps bandwidth whereas client-1 can support 90kbps and client-2 can support 80kbps. Now using multilayering approach, 3 layers will be formed. Base layer will be 60 kbps, first enhancement layer=20kbps,second enhancement layer=10kbps. C1 will get base layer and all enhancement layers. C2 will get base layer and one enhancement layers of 20 kbps. C3 will only get base layer.

There are two primary advantages of multilayered video encoding.

- First graceful degradation of video quality when loss occurs. Priorities are assigned to video layers, in the event of congestion packets from low priority layers are discarded first and thus protects the important base layer and other important enhancement layers.

- Second advantage is ability to support multiple destinations with different bandwidth constraints, for each such source-to-destination path with a unique bandwidth constraint, an enhancement layer of video may be added.

Simple multilayered approach is not sufficient as network bandwidth changes with time, so source must dynamically adjust number of layers to be generated and transmission rates of these layers.

2.1.2 Transcoding

A transcoder converts existing video stream into a new stream with a different format or rate. So we can decrease the encoding rate in the event of congestion. Transcoding is a technique employed by network proxies and servers to dynamically customize multimedia objects for prevailing network conditions and individual client characteristics. A simple approach for encoding is to decompress the video stream, process it and then recompress

it. This strategy is called spatial domain processing because it is performed on the original pixels. The efficiency of spatial domain processing is relatively low as it involves computationally intensive procedures for fully encoding and decoding. We can achieve fast transcoding by directly manipulating the compressed data in the frequency domain. Consider the figure 2.1 here using transcoding approach, we need to transcode just after the source-node and transcoded data of 60 kbps will be sent to C3 whereas 90kbps quality data will flow on other link. Again this 90 kbps quality data will transcoded to 80kbps and this 80kbps stream will goto client-2. Client-1 gets full 90kbps quality data.

2.1.3 Other adaptation based mechanisms

- Buffer-Based adaptation: Buffer based adaptation scheme [4] uses occupancy of a buffer on the transmission path as a measure of congestion. Goal of the control algorithm is to maintain buffer occupancy at a constant, desired level. When the buffers begin to fill up, the transmission rate is reduced in response and when the buffer begins to empty, the transmission rate is increased. Sender periodically receives explicit feedback from network giving buffer occupancy. Finding optimal buffer capacity is non-trivial task.

- Loss-based adaptation : Based on feedback information from a receiver,the sender assumes that the receiver is in one of three states unloaded, loaded, or congested. In the unloaded state, the sender progressively increases its transmission rate in an additive manner in response to feedback,until the network state is driven into the loaded state or the sender is sending the maximum useful rate. In the loaded state, the sender maintains a constant transmission rate. Depending on packet loss feedback, it can be driven into either the unloaded or the congested state. In the congested state, the sender progressively reduces its transmission rate multiplicatively until the reported loss decreases to the loaded state. Here determining these threshold is important activity.

- Rate based approach is discussed in [5]. This mechanism uses a closed feedback loop , where the source periodically multicasts feedback packet to each destination and destination returns the packet back to source. As the feedback packet traverse the network, intermediate nodes examine their current state of congestion and determine

the number of layers the video source node should generate as well as explicit rates of each layer. Source adjusts the encoding behavior accordingly.

- Hybrid streaming mechanisms are discussed in [6] [7]. Here one part of the network is well provisioned with respect to bandwidth whereas in the other part there are data losses. This work mainly focuses on DEP applications using satellite communication.

- Simulcast approach is discussed in [8]. Here source maintains different quality stream and receivers switch across streams depending upon available bandwidth. This approach is combination of single-rate multicast and multiple-unicasts.

A combination of above mechanisms can also be used. Majority of the existing work for video transmission focuses on, starting the playout as early as possible. And sudden variation in available bandwidth result in varying video quality at client. In our approach we try to minimize these variations. Next section explains the solution in detail.

Chapter 3

TPAMM Architecture

In this chapter, we present "Traffic pattern based adaptive multimedia multicast (TPAMM) architecture". Aim of this architecture is to minimizing abrupt changes in video quality during playout, even in the presence of variations in available bandwidth. There are three different modules in this architeture:

- Input Modules
- Functional Modules
- Output Modules

3.1 Functional Modules

There are four important functional modules.

3.1.1 Network monitoring and feedback module

This module is responsible for monitoring available bandwidth across each link in the network and providing this feedback to the source. Available bandwidth values are sent to source after every feedback interval. This module runs at all the nodes in the network. This feedback is used to refine the bandwidth prediction model. E.g. Tools like perfmon can be used to monitor network interfaces.

3.1.2 Bandwidth prediction module

This module predicts available bandwidth values for near future. The amount of time values are predicted is called prediction window. Using the network logs we can identify

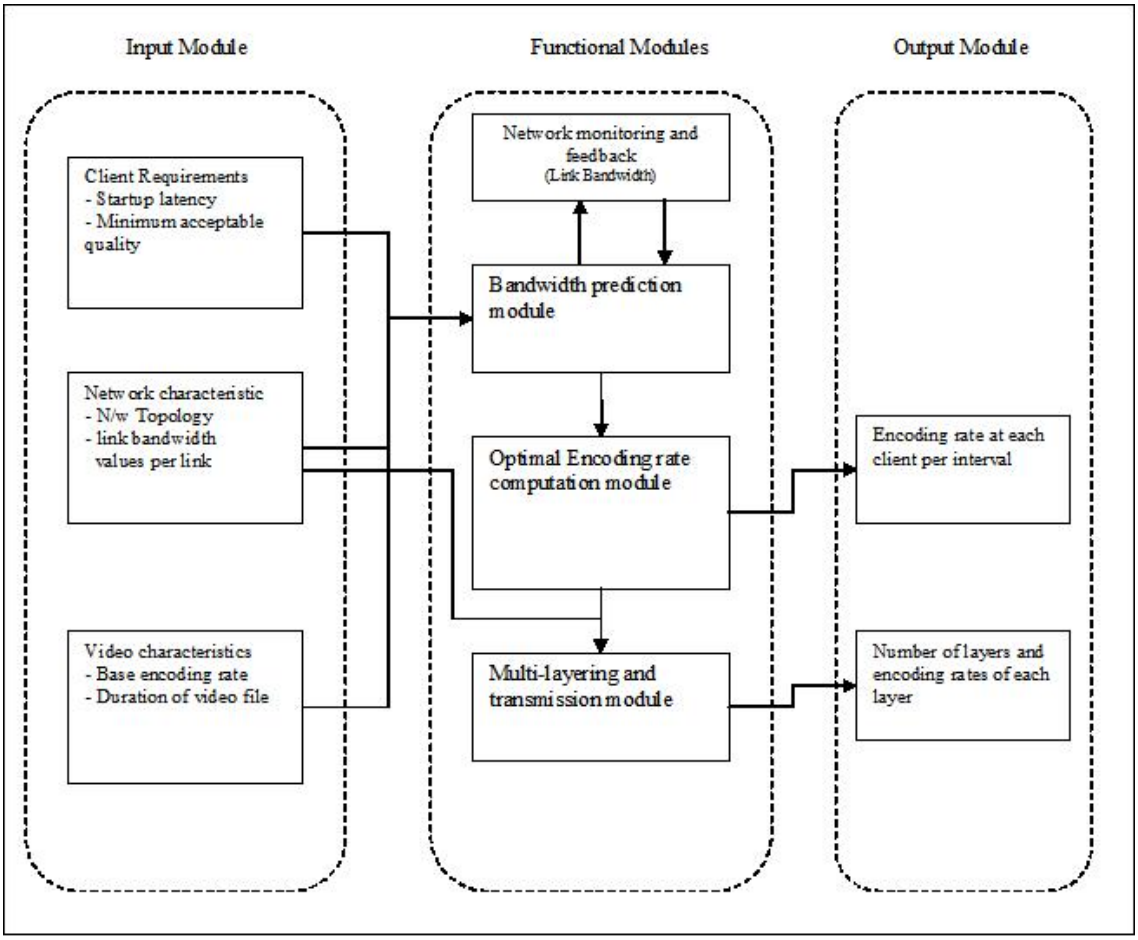

Figure 3.1: TPAMM Architecture

certain traffic patterns over time and build initial prediction model. This prediction model is refined using the feedback from the receivers after every feedback interval. These prediction values are used by optimal encoding rate computation module.

3.1.3 Optimal encoding rate computation module

This module computes encoding rate that each client can get per interval. These encoding rates are computed with an objective of minimizing the variation in video quality during playout. Client requirements, video file characteristics and bandwidth prediction model are used to compute encoding rate at clients. Algorithm for this module is given in detail in next chapter for various scenarios.

3.1.4 Multi-layering and transmission module

This module is responsible for deciding the number of enhancement layers and encoding rate for each layer. After the encoding rates of each client are decided by "Optimal encoding rate computation module", this information is combined with network topology information to decide number of layer and encoding rate of each layer.

3.2 Input modules

There are three major inputs to the system:

3.2.1 Client Requirements

Client Requirements comprises of two parts

- **Startup latency :** This is the maximum amount of time client is ready to wait before playout starts.
- **Minimum acceptable quality :** This is the minimum video quality that client is ready to accept; this is specified in terms of encoding rate.

3.2.2 Network characteristic

Network topology: This indicates total number of nodes and an adjacency matrix link bandwidth: This consists of available link bandwidth values for each link in network per interval

3.2.3 Video characteristics

Base encoding rate: This represents encoding rate of original video file. When we are transcoding to other encoding rate, target encoding rate will be less than base encoding rate.

Duration of video file: This represents the total duration of video file.

3.3 Output modules

There are two important output parameters.

- **Encoding rate at each client per interval :** This is provided by optimal encoding rate computation module
- **Number of layers and encoding rates of each layer :** This is provided by multi-layering and transmission module.

In the next chapter we discuss algorithm for "Optimal encoding rate computation module". Various scenarios are considered are single hop transmission, multihop transmission and transmission over multicast tree.

Chapter 4

Solution Strategy

In previous chapter we have discussed overall system architeture. We can summarize important functions as

- Gathering the network traffic characteristics and client requirements
- Computing encoding rate at which data should be sent to clients
- Transmitting the video file data over multicast tree

In this chapter we focus on algorithm for *Optimal encoding rate computation*. The available bandwidth for each link varies with time at every feedback interval. Now if we adjust encoding rate of our video file as per changing link bandwidth, client will see frequent variation in the video quality during playout. Our objective is to minimize this variation in the video quality during playout.

According to client requirements and available link bandwidth, encoding rates are decided per feedback interval and data is sent to clients at this encoding rate. Here we will explain our solution strategy with example scenarios and finally we will provide the overall algorithm for video multicast over a tree topology. Here we explain 3 different scenarios.

- Single hop scenario (S-C): Here client node is at one hop distance from Source node
- Multihop scenario (S-R1-R2-...-C): Here client node is at multiple hop distance from source
- Multicast tree scenario: Here the client is at leaf node of the tree and source is the root of this tree

We now explain each scenario in detail. In each scenario we explain topology, available inputs, solution approach and output.

Finally overall algorithm will be explained where bandwidth predictions values are available only for some time-window.

4.1 Single Hop Scenario

In single hop scenario, client is directly connected to source as shown in figure. For this link available bandwidth values are known using bandwidth prediction model.

4.1.1 Inputs parameters

Inputs parameters for single hop scenario are

- Base encoding rate of original video file
- Minimum acceptable video quality at client
- Feedback interval = 10 sec
- Length of video file(L) = 100 sec
- Client Delay tolerance value $\delta = 20$ sec
- Available bandwidth at each feedback interval

4.1.2 Single hop solution strategy

Our basic solution strategy is to send data at average encoding rate. This average is computed over available bandwidth values for many feedback intervals, it supresses the sudden changes in available link bandwidth. Compute accumulated bandwidth over time $\int_0^{\delta+L} a_i$ where a_i is available bandwidth during interval i. This accumulated data is consumed only during the playout duration (L), so while computing the average rate we divide only by L and not by $\delta + L$.

$$A_{avg} = \frac{\int_0^{\delta+L} a_i}{L} \tag{4.1}$$

4.1.3 Output Parameters

Output is A_{avg} encoding rate to which original video file should be transcoded for the client for the given delay tolerance value.

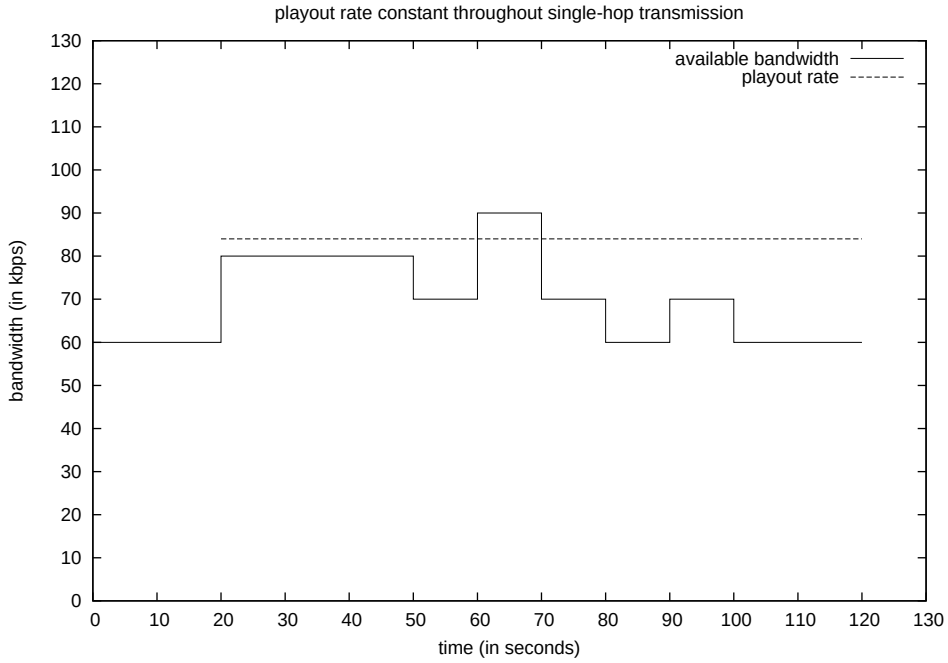

Figure 4.1: Singlehop eg1: Available Bandwidth Vs Playout Rate

4.1.4 Single hop scenario examples

In figure 4.1 we can see the variations in available bandwidth from source to client, whereas the encoding rate during playout does not change. This is because from figure 4.3 we observe that during playout, accumulated data is always greater than consumed data till that time. Thus continuous playout at average encoding rate is feasible. So in example 1 4.1 video is played at 84 kbps.

But in figure 4.4 we observe that rate of data consumption at client is greater than data accumulation rate. Hence at time t=100 second, average encoding rate can not be sustained. We can not have continuous playout at this average encoding rate. Reason being large portion of bandwidth is accumlated towards end of playout, which increases the average encoding rate but we can not sustain this rate in initial phase of transmission. Hence we enhance our algorithm to take care of these cases and find such critical points

(e.g. t=100) Thus in second example figure 4.2 we observe that change in encoding rate, from time t=20 to t=100 video is played at 82.5 kbps and at 90 kbps for t=100 to t=120

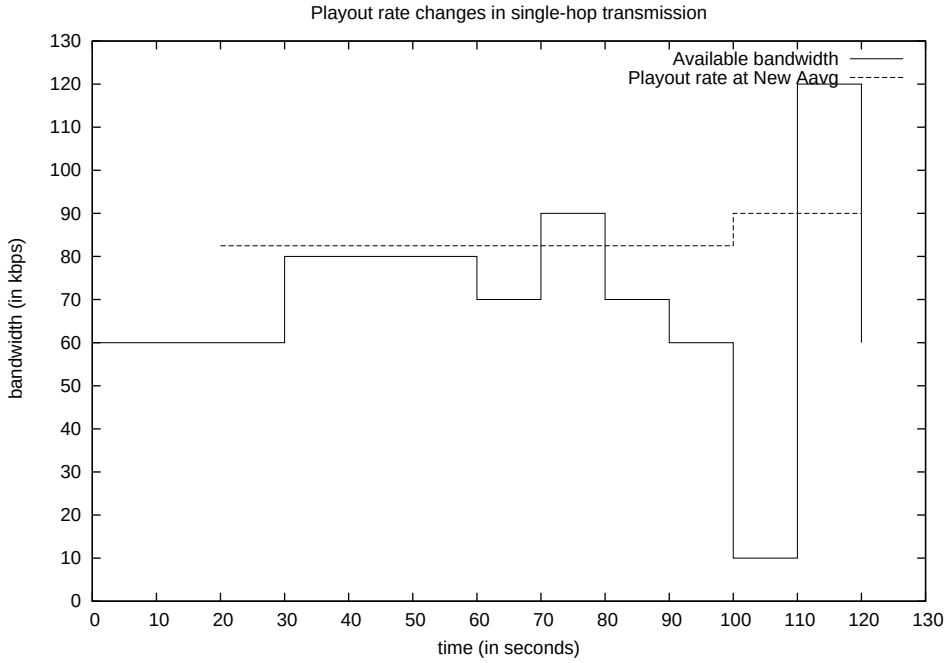

Figure 4.2: Singlehop eg2: Available Bandwidth Vs Playout Rate

4.1.5 Single hop algorithm

Finally we present the algorithm for single hop module in algorithm 1.

4.2 Multihop scenario

In multiple hop scenario, client is connected to source through one or more relay nodes as shown in figure. Each link on the path from source to client show different varition in available bandwidth per feedback interval. So our objective here is to find effective amount of data received by client per feedback interval.

4.2.1 Input parameters

Available bandwidth values per feedback interval, for each of these links from source to client are known using bandwidth prediction model. So inputs for multihop algorithm are

Algorithm 1 Algorithm to compute data received after singlehop

δ is startup latency

L is duration of video file

initially critical_point and new_critical_point both are (L+delay_tol)

critical_point is time at which (rate of data consumption > rate of data accumulation)

while critical_point < last_critical_point **do**

compute cumulative_total = $\int_0^{\delta+L} a_i$

compute

$$A_{avg} = \frac{\int_0^{\delta+L} a_i}{L} \tag{4.2}$$

new_critical_point = critical_point

for t=critical_point;t ¡= last_critical_point;t+=feedback_interval **do**

update accumulated_bw

update consumed_bw

if accumulated_bw < consumed_bw **then**

if critical_point > delay_tol **then**

new_a_avg = (float)(accumulated_bw * feedback_interval)/ (float)(t_critical_point)

else

new_a_avg = (float)(accumulated_bw * feedback_interval)/ (float)(t_delay_tol)

end if

if new_a_avg < a_avg **then**

a_avg=new_a_avg

new_critical_point = t

// it will be chked whether this a_avg is supported throughout

end if

end if

end for

end while

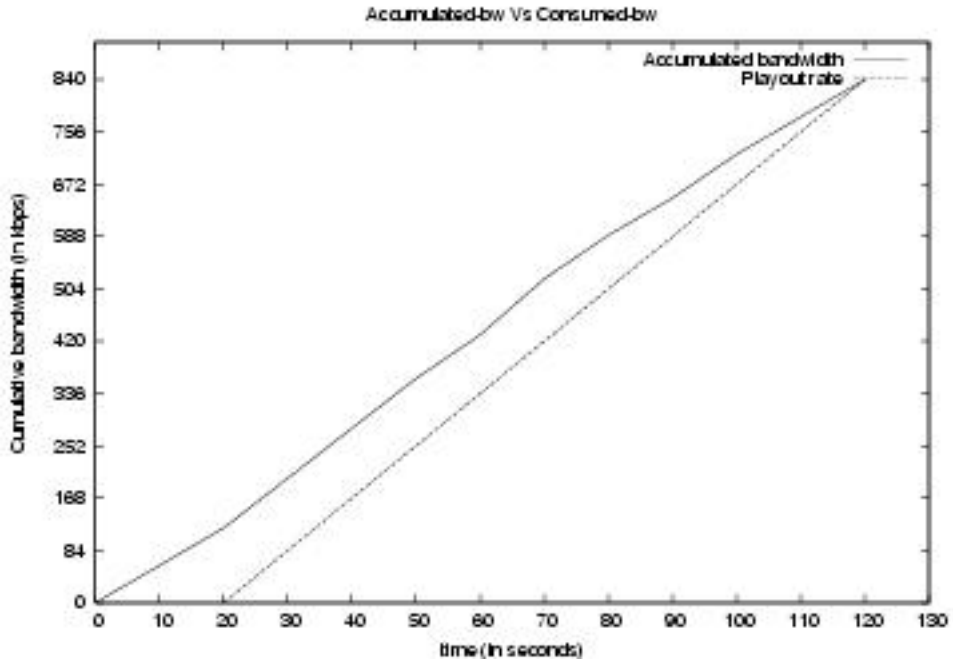

Figure 4.3: Singlehop eg1: Data Accumulation Rate Vs Data consumption Rate

- Number of hops from Source to Client
- Available bandwidth values per feedback interval, for each link on the path from source to client

4.2.2 Multi hop solution strategy

Link-1 is from Source-Relay node.
Link-2 is from Relay node-Client.
In the figure 4.5 two dotted lines represent available bandwidth at link-1 (Source-to-Relay node) and link-2 (Relay node-Client). And the dark line represent end-end transmission rate achieved per feedback interval from Source-Client.

At the intermediate nodes during one feedback interval, amount of data received may not be equal to amount of data transmitted. When the amount of data received is more amount of data transmitted, remaining data is buffered at intermediate nodes. So there is some *deficit* at intermediate node which should be transmitted later on. So if after sometime the downlink from this intermediate node improves, this *deficit* can be cleared. If there is no data already buffered at intermediate nodes then the extra bandwidth at

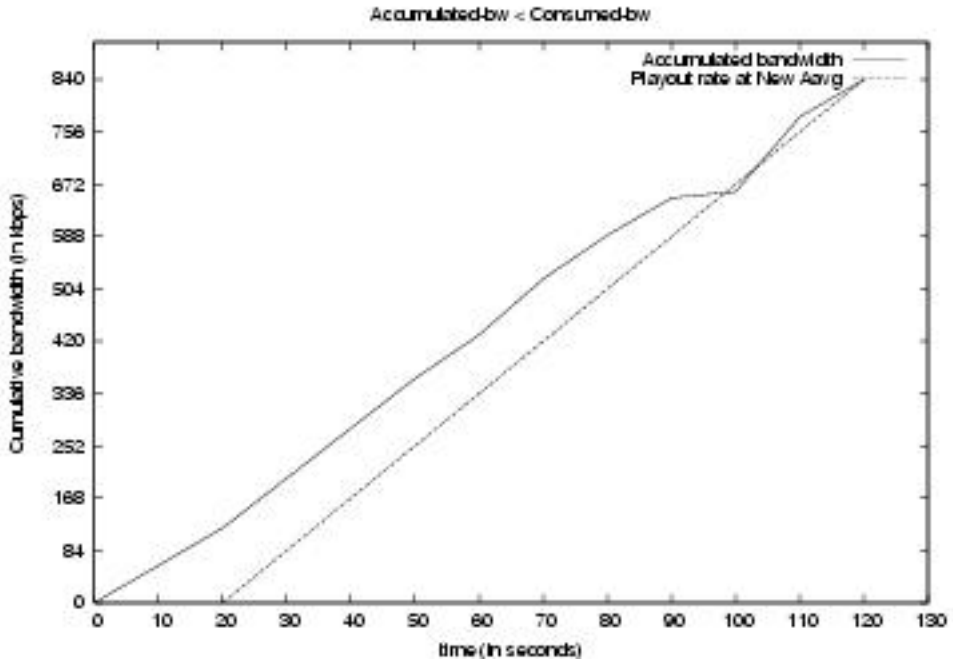

Figure 4.4: Singlehop eg2: Data Accumulation Rate Vs Data consumption Rate

downlink will be under utilized. So different cases are listed here

- Extra bandwidth at link-2 can not be utilized: This happens when there is no data buffered at intermediate node and link-2 bandwidth is more then link-1 bandwidth.

- Extra bandwidth at link-2 utilized to compensate the deficit at intermediate node: This happens when there is already some data put in the buffer of intermediate node and extra bandwidth can be used to send the data that is buffered.

- Less bandwidth at link-2 causes to be buffered at intermediate node.

The effective transmission rate received after k-hops and transmission rate available at link on k+1 hop, is used to find effective transmission rate received at a node after k+1 hops. So this algorithm incremently find the effective rate transmited to each node on the path from source to client.

Finally, end-end effective data received per feedback interval, from source to client is known. So now this is similar to end-end single hop from source to client. We now use the singlehop algorithm discussed in previous section 4.1, to find encoding rate per feedback interval during video playout at client.

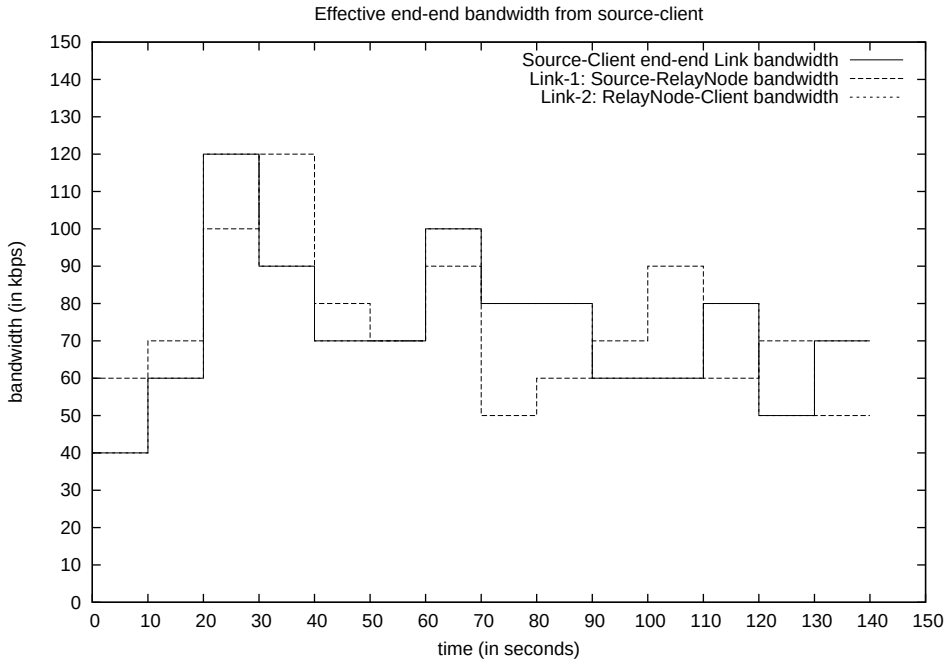

Figure 4.5: Multihop: Effective end-end bandwidth from source-client

4.2.3 Output parameters

This multihop algorithm outputs end-end effective data received per feedback interval, from source to client.

4.2.4 Multihop algorithm

The complete algorithm to find effecticve end-end transmission rate during each feedback interval is presented in algorithm 2.

4.3 Multicast tree scenario

In multicast scenario, clients are at the leaf level of the multicast tree. Source is root of the tree. All clients specify their delay tolerance values. Available link bandwidth values per feedback interval are known from prediction model. These clients share some common links, so providing very high quality data for one client may violate the delay tolerance constraint of other client. So our objective is to find out encoding rates for each client while satisfying the delay tolerance constraints imposed by all clients.

Algorithm 2 Algorithm to compute_data_received_after_multihop

```
link_info[total_no_of_nodes][total_no_of_feedback_intervals] is available from bandwidth
prediction module
compute path_from source for each node
total_deficit=0
link_1 = link from source to relay_node
link_2 = link from relay_node to client
for every feedback interval do
  Interval_deficit = max(0,link_1_Tx_rate - link_2_Tx_rate)
  Interval_extra = max(0,link_2_Tx_rate - link_1_Tx_rate)
  total_deficit = total_deficit + Interval_deficit
  if total_deficit==0 then
    Interval_Tx_rate = link_1_Tx_rate
  else
    if interval_extra¡=total_deficit then
      Interval_Tx_rate=link_2_Tx_rate
    else
      Interval_Tx_rate= link_1_Tx_rate + total_deficit
    end if
  end if
end for
```

4.3.1 Input parameters

Inputs for multicast tree topology algorithm are

- Delay tolerance values for each client
- Available link bandwidth values per feedback interval
- Adjacency information of nodes in the tree.

4.3.2 Multicast tree topology solution strategy

In multicast topology clients share common links, so providing very high quality data for one client over the common link may increase the delay for the other client. So we must ensure all client tolerance constraint are satisfied. Algorithm for multicast scenario is presented in 4.3.4. Here we explain our soultion strategy with an example.

Figure 4.6 shows a multicast tree with 3 nodes. Different delay tolerance values of clients C1,C2,C3 are 20,40,30 resp.

- From the given node adjacency information we find path of each client from the source. Table4.1 shows these are computed paths for each node.
- Find all delay tolerance values of clients in the topology. Refer 4.1
- Isolate the client and find best possible encoding rate.
  Here for each client we assume, this is the only client in the network and now this case is similar to multihop scenario. We find the possible encoding rate for this client in multihop scenario. This value is an upper bound on the encoding rate this client can get, since currently we have ignored constraints due to other clients in network. We do this for all different delay tolerance values. For the example topology, these computed values for each node are shown in table 4.1
  e.g. With delay tolerance of 20 seconds client-4 can get data encoded at 70 kbps and if this client waits for 30 seconds he can achieve 75 kbps encoding rate.
- Now client with minimum delay tolerance value is chosen first.
  And for each of his anscestor from the source, constraint is put on the maximum encoding rate of anscestor. Even with larger delay tolerance value encoding rate supported by this anscestor can not exceed the rate this anscestor supports now.

- e.g. Refer to figure 4.6 and table 4.1. Here client-2 has delay tolerance = 20 sec. And node R1 can support maximum encoding rate of 84 kbps with delay tolerance=20. Now if data with encoding rate more than 84 kbps is sent to R1, it will violate delay tolerance constraint for Client-1 (C1). Hence we put constraint on the values for node R1 in table 4.1, for delay tolerance=30 sec, now node R1 can support only 84 kbps and it can not support earlier value of 90 kbps.

- Also for client each of its parent must support the encoding rate otherwise client's encoding rate is restricted to parent's encoding rate.
  e.g. with delay tolerance=40, client-3 can get 90 kbps but its parent R1 can only support 84 kbps, So client-3's encoding rate is also restricted to 84 kbps.

- This is repeated till the maximum delay tolerance value. Finally encoding rates of each of the client is output as per corresponding delay tolerance value.

- C1 will get data encoded at 75 kbps, C2=84 kbps and C3=75kbps.

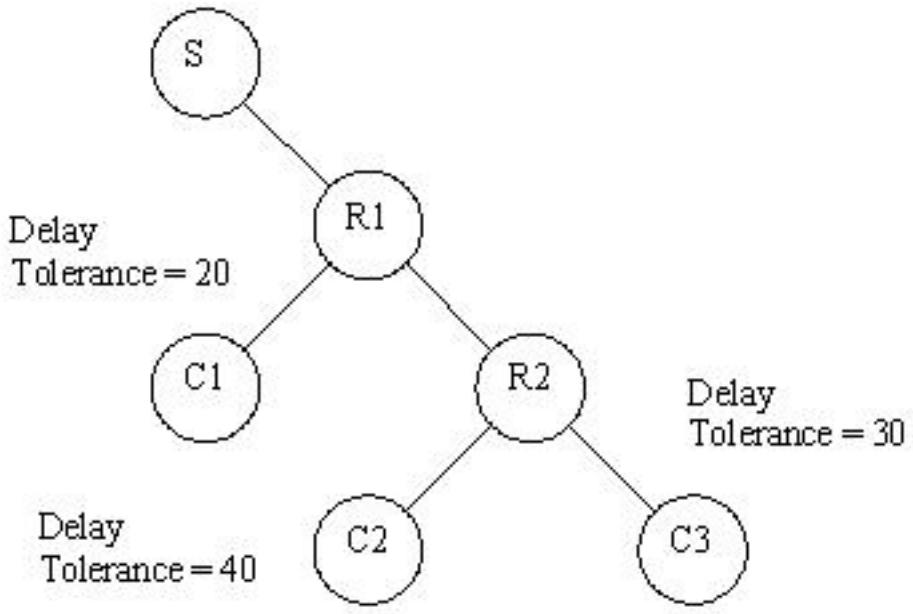

Figure 4.6: Example Multicast topology with 3 clients

Figure 4.6 shows small multicast topology, with 3 clients.

Node	Path	$\delta = 20$	$\delta = 30$	$\delta = 40$
1	S-R1	84	90 (84)	95 (84)
2	S-R1-C1(20)	75*	75	75
3	S-R1-R2	80	85 (84)	90 (84)
4	S-R1-R2-R3-C2(40)	80	85	90 (84)
5	S-R1-R2-C3(30)	70	75*	75

Table 4.1: Encoding rates with different delay tolerance values

4.3.3 Output parameters

Multicast algorithm provides encoding rates for each client while satisfying the delay tolerance constraints imposed by all clients. In the example topology C1 will get data encoded at 75 kbps, C2=84 kbps and C3=75kbps.

4.3.4 Multicast algorithm

The complete algorithm to find encoding rates for each client while satisfying the delay tolerance constraints imposed by all clients, is presented in algorithm 3.

4.4 Prediction-window based overall algorithm

Given the delay-tolerance value and link bandwidth prediction for entire duration of video file, earlier algorithms outputs the encoding rate at each client per feedback interval.

In this section we assume link bandwidth prediction values are available only for a time window, we call this prediction-window.

- At any time t, we know link bandwidth predction values from time=t to time=(t + windowlength).

- After every feedback interval, prediction window slides by feedback interval length.

Algorithm 3 Multicast algorithm gives the optimal encoding rate at each client can get per feedback interval

```
for every feedback interval (for each value of current_time) do
  for every diff value of startup latency (loop for diff δ) do
    for every_client with current_startup_latency do
      if client_startup_latency == current_startup_latency then
        recent_parent = source_node
        for each of the parent from source_path (for (no_hops -1)) do
          current_parent = node_id
          if current_parent_bw < constraining_node_bw_entry then
            constraining_node_bw_entry=current_parent_bw
            //repeating this bandwidth ahead till last startup_latencies
          else
            if recent_parent != source then
              if recent_parent_bw < current_parent_bw then
                current_parent_bw = recent_parent_bw
              end if
              repeat (current_parent_bw) ahead
            end if
          end if
          recent_parent = current_parent
          recent_parent_bw = current_parent_bw
        end for
      end if
    end for
  end for
end for
```

- For each prediction window, we update *current-delay-tolerance* values and *cuurent-playout-duration*. And use algorithms discussed in earlier sections to find encoding rates.

4.4.1 Prediction-window example

Here we present prediction-window based video transmission example. Input parameters are
prediction-window-length = 50 sec
total-duration-of-video-file = 160 sec
max-delay-tolerance = 40 sec
feedback-interval = 10 sec

We compute following parameters initially

- last-prediction-time = (max-delay-tolerance + duration-of-video-file) - prediction-window-length So in this example last-prediction will be at time 150 sec, at time 150 sec we will have predictions for next 50 seconds. So we have link bandwidth values till time 200. And playout ends at time 200 sec.

- current-delay-tol = delay-tol * (prediction-window-length / (max-delay-tolerance + duration-of-video-file))
  In this example initially current-delay-tol will be set to 10 sec.

- current-playout-time = duration-of-video-file * (prediction-window-length / (max-delay-tolerance + duration-of-video-file))
  In this example initially current-playout-time will be set to 40 sec.

We can now find encoding rates at each client with current values of delay-tolerance and current-playout-duration. At the next feedback interval we will compute time-offset and update input values. Time-offset computation is explained in next section.

4.4.2 time-offset computation

In each feedback interval, the amount of data we send to client is interval-Tx-rate. This data will be consumed in playout-interval-duration. Consider an example, say feedback

interval is 10 sec. The encoding rate of the data to be sent is 90 kbps, this is the rate at which video will be played at client. And transmission rate in the interval is 60 kbps, this is the rate at which data is transfered. So the data that we transfer in 10 seconds will be (60*10)kbps and this data is consumed in time (60*10)/90 =6.66 seconds.
Now each node has current delay tolerance value, using this we can compute expected-playout-time-per-interval as

- playout-interval-duration = (interval-Tx-rate * feedback-interval) / interval-encoding-rate-value
- expected-playout-time-per-interval = current-L / window-length * feedback-interval

Now we compute time-offset and update the delay-tolerance and playout-duration values as

- offset = playout-interval-duration - expected-playout-time
  when offset is positive we are sending data at a faster rate
  when offset is negative we are sending data slower
- update current-delay-tol = current-delay-tol + offset
- update current-playout-time = current-playout-time - offset

Now update the current-time
current-time = current-time + feedback-interval

Repeat the algorithm for prediction-window till we reach last-prediction-time.

4.4.3 offset comutation example

Here are two offset comutation example: Example 1 : with Interval-Tx-rate=30

- interval-encoding-rate-value = 60
- feedback-interval = 10
- playout-interval-duration = 10 * (30/60) = 5 sec
  Data sent in 10 sec interval will be consumed in 5 second.

- expected-playout-time-per-interval =10 sec
- offset = playout-interval-duration - expected-playout-time
  offset will be (5-10) = -5 seconds, indicating we are sending data at slower rate.

Example 2 : with Interval-Tx-rate=90

- interval-encoding-rate-value = 60
- feedback-interval = 10
- playout-interval-duration = 10 * (90/60) = 15 sec
  Data sent in 10 sec interval will be playout out for 15 second.
- expected-playout-time-per-interval =10 sec
- offset = playout-interval-duration - expected-playout-time
  offset will be (15-10) = 5 seconds, indicating we are have already sent data for next 5 sec.

Complete prediction-window based algorithm is presented in algorithm 4.

Algorithm 4 Algorithm for Optimal encoding rate module

```
Compute maxmimum-delay-tolerance and total-no-of-feedback-intervals
for each node multicast tree in BFS order do
   compute attribute no-of-hops-from-source
   Store Path-from-source
end for
Find max-no-hops-in-tree
Read different-delay-tolerance-values from clients requirements
for each node in the multicast tree do
   delay-tolerance-value = min (delay-tolerance-values of its children)
end for
Run Multihop Algorithm
Compute delay-tolerance-per-window and playout-duration-per-window
current-delay-tol = delay-tol * (prediction-window-length / (max-delay-tolerance +
duration-of-video-file) )
current-playout-time = duration-of-video-file * (prediction-window-length / (max-
delay-tolerance + duration-of-video-file) )
while current-time < last-prediction-time do
   for every feedback interval upto prediction-window-length do
      for each node do
         for each different-delay-tolerance-value do
            run effective-single-hop algorithm at each node
         end for
      end for
   end for
   Run multicast tree algorithm
   Output rate at each client after multicast algorithm
   playout-interval-duration = (interval-Tx-rate * feedback-interval) / interval-
   encoding-rate-value
   expected-playout-time = current-L / window-length * feedback-interval
   compute offset = playout-interval-duration - expected-playout-time
   update current-delay-tol = current-delay-tol + offset
   update current-playout-time = current-playout-time - offset
   current-time = current-time + feedback-interval
end while
```

Chapter 5

Simulation Experiments and Results

In this chapter we present the simulations and results. The objective of these experiments is to compare our TPAMM scheme with multilayering mechanism. Important claims are

- *Smooth video playout at client:* Compared to multilayering scheme, TPAMM scheme should provide video to client at smooth rate. i.e. sudden variation in playout rate at client minimized.

- *Maximize the minimum video quality during playout:* Due to link bandwidth variation the video quality changes at client over time. And for some time we may not get good quality video. Compared to multilayering scheme TPAMM scheme tries to maximize the minimum quality of video playout.

5.1 Simulations steps

We perform the simulations to validate above claims. Simulations consists of following steps

- *Traffic profile generation:* All the links in the network show variations in available bandwidth. So this module randomly generates various traffic profiles in the given range of bandwidth values. Different links in the topology are assigned different traffic profiles. Traffic profile is active for the duration of entire duration of playout and startup latency.

- *Optimal Encoding rate computation module* This module outputs the encoding rate at each client in network. We have explained this module is detail in chapter 4

- Post processing to compute comparision parameters: Encoding rates output by optimal encoding rate computation module, are processed to find comparision parameters such as standard deviation of encoding rates, minimum encoding rate during playout.

Simlations are run with duration of video file in the range 100-2000 sec. Feedback intervals in the range 10-50 sec. Delay tolerance values are some fraction of duration of video file. Next section we compare the two schemes with respect to varios parameters.

5.2 Results

We first study the performance of TPAMM scheme for various delay tolerance values and then we compare TPAMM scheme with multilayering scheme. Simulations are run with 160 sec video file duration, 10 sec of feedback interval and 50-120 kbps link bandwidth range.

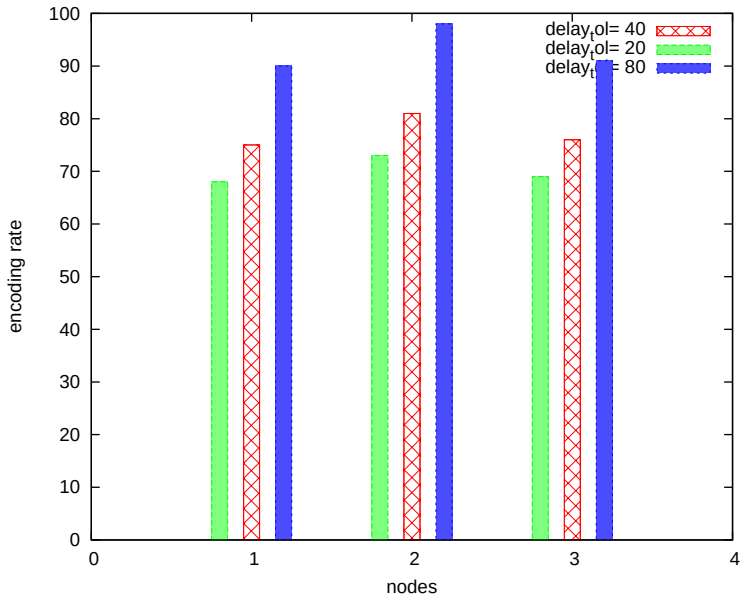

Figure 5.1: Effect of delay tolerance on encoding rate

5.2.1 Effect of client delay tolerance on video quality

Topology is same for all runs in this case. We vary traffic profile with different random seed. Delay tolerance values are 20,40,80. Parameter being observed is average encoding

rate at each client.

In figure 5.1 we see that at every node, encoding rate improves with increase in delay tolerance. This is because if we wait for more time, more data will be transfered and hence average encoding rate improves.

5.2.2 Effect of prediction window size on video quality

In this experiment we observe the variation in video quality at client for different prediction window sizes. Traffic profile is same for all runs, but prediction window size is changed. And link bandwidth values are available only within current prediction window. Delay tolerance value is 40 sec.

Parameter being observed is standard deviation of the average encoding rate during playout. We chose to observe standard deviation because,it is a number that indicates how much on an average each of the values deviates from the mean.

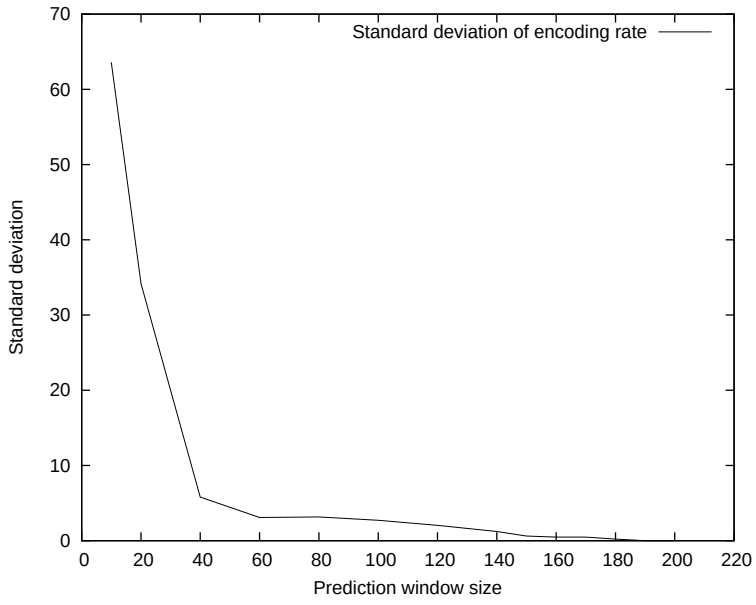

Figure 5.2: Prediction window size Vs Standard deviation of Encoding rate

Prediction window size is increased from 10 sec to 200 sec. From the figure 5.2 we can see that as the prediction window size increases, standard deviation of the encoding rate becomes smaller. So if we have more predictions available playout at client will be smoother.

Interesting thing to note in this graph is, with small increase in the prediction window

size in the initial part, there is significant drop in variations. So even for small prediction window size we acheive smoother playout at client. When the prediction window size is equal to one feedback interval, this is same as using multilayering scheme.

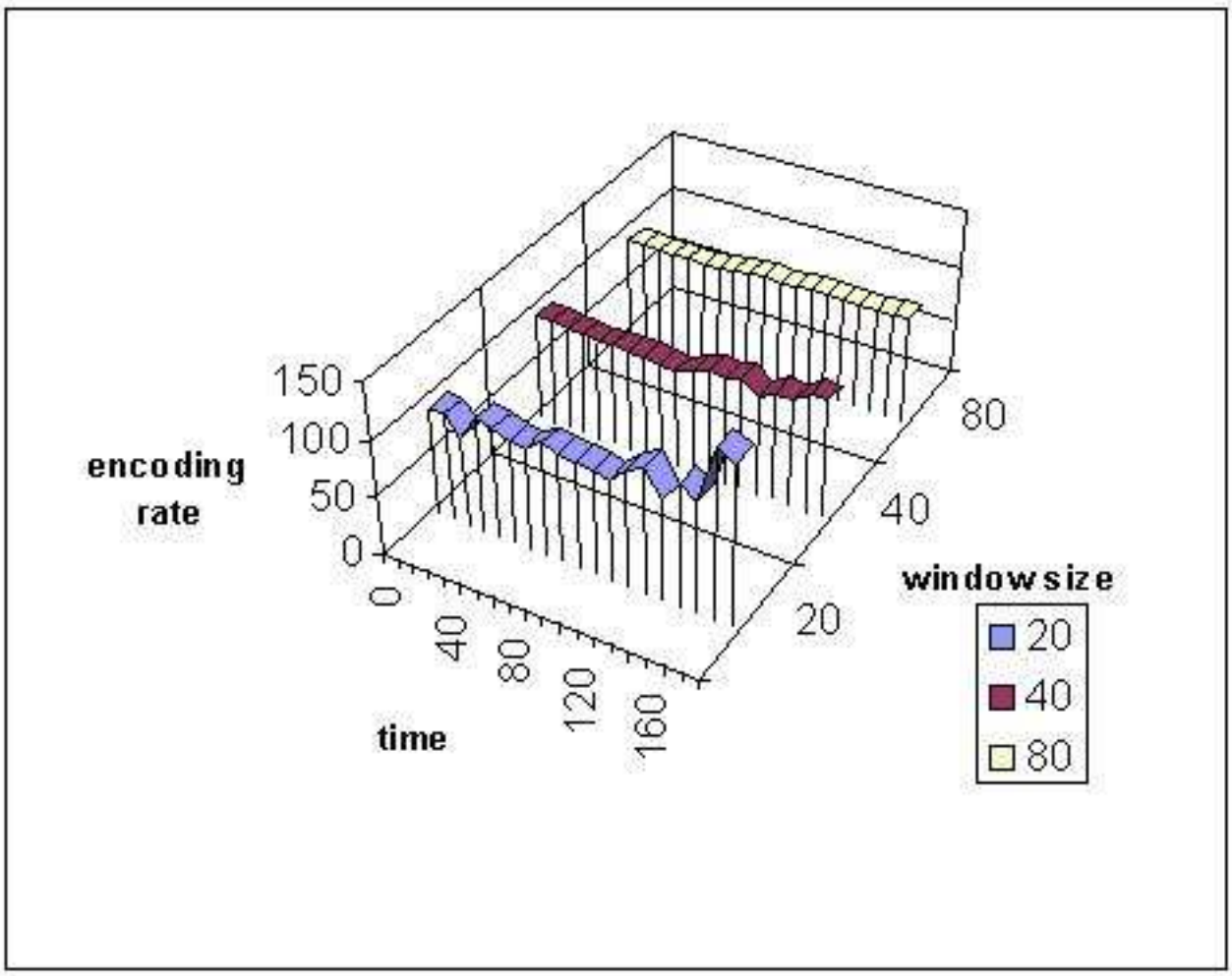

Figure 5.3: Effect prediction window size on video quality

In figure 5.3 we plot encoding rate of video along Z-axis, time along X-axis and prediction window size along Y-axis. We have shown graph for three prediction window sizes 20,40,80. Here we can onserve with increase in the prediction window size, encoding rate during playout becomes smoother i.e. there are less variation.

5.2.3 Maximize the minimum video quality

In our scheme, even when we have good bandwidth we may transmit at moderate encoding rate so as to compensate for the low bandwidth period later on. Hence even if when have

low bandwidth we can still sustain moderate quality. In this experiment we plot minimum

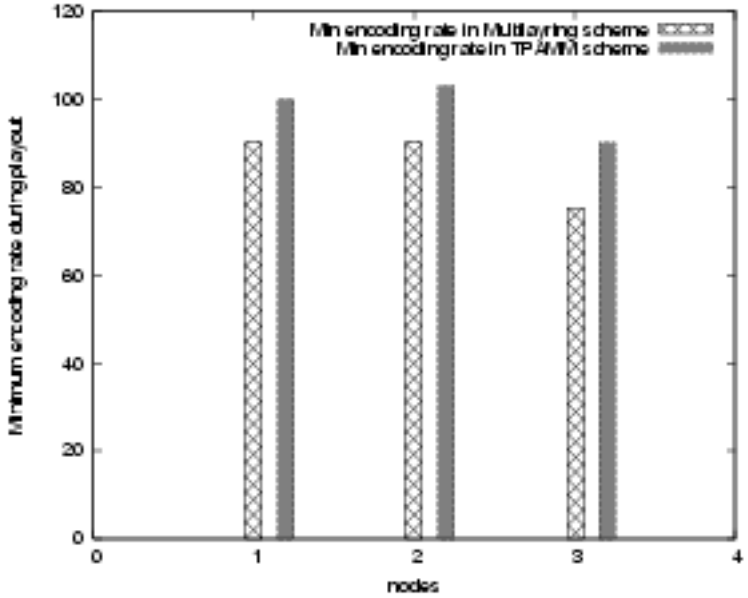

Figure 5.4: Minimum Encoding rate during playout

encoding rate during playout at each client in both multilayering and our approach. From figure 5.4 we can see that at every client minimum playout rate provided by our method is more than multilayering method.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

From our experiment results in previous chapter we have seen that compared to Multi-layering scheme, TPAMM scheme minimizes the variations in video quality. Using TPAMM scheme, we also observed as the prediction window size increases variations in the encoding rate at client are minimized. From the experiments we have seen that using client delay tolerance we can provide better encoding rates to client. Also TPAMM scheme maximizes the minimum video quality during playout."Delay tolerance can be used to minimize uncertainty caused by varying available bandwidth".

6.2 Future work

Future directions for our project are

- *Priority assignment to client:* Clients can be either paid or free users. Our scheme should be refined to give better encoding rates to client with higher priority.
- *Asynchronous video transmission:* Currently we assume all clients requests at available at start. When some client joins while the transmission is in progress, he should able to get the data that is already stored at intermediate nodes. For this we need to combine our scheme with caching based streamming mechanisms.

Bibliography

[1] Taehyun Kim and Mostafa H. Ammar. A comparison of layering and stream replication video multicast schemes. In *In NOSSDAV'01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 63–72, New York, USA, 2001. ACM Press.

[2] Celio Albuquerque Brett J. Vickers and Tatsuya Suda. Source-adaptive multilayered multicast algorithms for real-time video distribution. In *IEEE/ACM Transactions on Networking*, pages 720–733, 2000.

[3] Turuk S. Chattopadhyay R. Kumar, Rao and G. K. Rao. A protocol to support qos for multimedia traffic over internet with transcoding. 2002.

[4] X. Wang and H. Schulzrinne. Comparison of adaptive internet multimedia applications. In *IEICE Trans. COMMUN.*, 1999.

[5] Celio Albuquerque Brett J. Vickers and Tatsuya Suda. Adaptive multicast of multi-layered video: Rate-based and credit-based approaches. In *In INFOCOM*, pages 1073–1083, 1998.

[6] S. Krithivasan and Sridhar Iyer. Mechanisms for effective dissemination of multimedia. 2004.

[7] S. Krithivasan and Sridhar Iyer. To beam or to stream:satellite-based vs. streaming-based infrastructure for distance education. In *In In World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 293–299, 2004.

[8] Bo Li Jiangchuan Liu and Ya-Qin Zhang. Adaptive video multicast over the internet. In *IEEE MultiMedia*, pages 10(1):22–33, 2003.

Acknowledgements

I take this opportunity to express my sincere gratitude for **Prof. Sridhar Iyer** for his constant support and encouragement. His excellent guidance has been instrumental in making this project work a success.

I would like to thank **Punit Rathod**, **Saraswathi Madam** and **Annanda** for their constant help throughout the project.

I would also like to thank my **family** and **friends** especially my entire **MTech Batch**, who have been a source of encouragement and inspiration throughout the duration of the project.

Last but not the least, I would like to thank the entire KReSIT family for making my stay at IIT Bombay a memorable one.

Laxmikant Patil

I. I. T. Bombay

July 17^{th}, 2006