Saba Ahsan

Multipath RTP: Applying Multipath Communication to Real-time Applications

Faculty of Electronics, Communications and Automation

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 16.3.2010

Thesis supervisor:

Prof. Joerg Ott

Thesis instructor:

M.Sc. Varun Singh

Aalto University
School of Science
and Technology

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF THE
MASTER'S THESIS

Author: Saba Ahsan

Title: Multipath RTP: Applying Multipath Communication to Real-time Applications

Date: November 20, 2011 Number of pages:

Faculty of Electronics, Communications and Automation

Department: Department of Electrical and Communications Engineering

Professorship: Networking Technology Code: S-38

Supervisor: Professor Joerg Ott

Instructors: M.Sc. Varun Singh

In the current Internet, most transport protocols select a single path for communication flow between two end hosts, even when multiple paths exist. Such flows are unable to fully utilize the available resources. Multipath capability refers to the simultaneous use of multiple paths through the network, which may significantly improve performance and reliability. This area is of particular interest in real-time communication where it would improve the end-user experience by enhancing the quality of service. Firstly, bandwidth-hungry applications such as video streaming and IP-TV can benefit from the increased, combined throughput available to multihomed clients. Also, as retransmission of lost data is often uncharacteristic of real-time traffic because of time constraints, multipath senders can avoid lossy paths or send redundant data over multiple paths. Furthermore, session-based real-time communication can benefit from the redundancy by implementing failover in case of network failures.

In this thesis, we present Multipath RTP as an extension to RTP with multipath capability. We propose a MPRTP scheduling algorithm for sending RTP packets over multiple paths in the form of a basic MPRTP implementation. Finally, we evaluate its performance in a virtual test environment consisting of a two multihomed clients with three paths available between them.

Keywords: Real-time communication, Multipath transport, RTP

Acknowledgements

Research for this thesis was funded by the Trilogy project [18] and conducted at the Department of Communications and Networking at the Aalto School of Science and Technology. I would like to extend my gratitude to my supervisor Jörg Ott, without whose vision and experience, none of this would have been possible. I would also like to thank my instructor, Varun Singh, for his support and guidance.

My sincere thanks to my colleagues in room D301, who kept my spirits high even when things did not look so optimistic. I would also like to thank my parents for their continued support, and my husband for believing in me.

Otaniemi, November 20, 2011

Saba Ahsan

Table of Contents

Abbreviations

RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
HSN	Highest Sequence Number
MPRTP	Multipath Real-time Transport Protocol
MPRTCP	Multipath Real-time Transport Control Protocol
PSNR	Peak Signal-to-Noise Ratio
RR	Receiver Report
SR	Sender Report
ICE	Interactive Connectivity Establishment
QoS	Quality of Service

List of Figures

List of Tables

Chapter 1

Introduction

The last five decades of human history bore witness to one of the greatest technological development of all times; the evolution of information and communications technology (ICT). The present world is a global village of users separated by thousands of miles brought together by a network of communication standards encompassing a wide range of multimedia. Internet, mobile phones and IP phones are not a luxury being enjoyed by the privileged class but a necessity that is available to the general masses. This trend exists because the advancement in communication is fuelled by the needs of the user. As time goes on, the user expectations are rising and the developers are striving to break new grounds and also to optimize the current technology.

In the field of multimedia communication, higher performance and user availability are of great significance. From a communication network perspective, higher performance may be higher throughput, lower loss rates or greater reliability through redundancy. Protocols capable of utilizing available network resources to enhance any of these qualities can greatly improve the end-user experience and produce new business opportunities. Communicating devices today are equipped with multiple network interfaces of different or same communication standards. This adds reliability and redundancy in communication and also allows access to multiple networks simultaneously, such as WLAN and Ethernet in computers. Many times multiple paths would exist between two endpoints, which may or may not share common bottlenecks. Even though middle boxes such as routers and switches provide load sharing and redundancy by using multiple paths; most protocols do not recognize nor simultaneously use the network connections available to end devices.

Optimization possibilities exist for protocols that can recognize the multiple network connections and use them as one single resource with higher throughput and greater reliability. This behaviour may be referred to as pooling of network connections or simply multipath capability, as the protocol would be using more than one path for communication. It would be particularly beneficial in scenarios where common bottlenecks do not exist. Multipath TCP is one example of many efforts that are already being done to add this feature to existing transport protocols. Our emphasis, however, is on multimedia communication such as live or buffered video streaming, Voice over IP, IPTV etc. We may be able to reap maximum benefits if such enhancements can be added to these bandwidth-hungry, time and reliability constrained traffic flows, by simply utilizing already available resources.

1.1 Problem Statement

Multimedia communication applications on packet networks have been struggling to provide the quality of service that was available to the customer on circuit switched networks. Reliability, higher throughput and lossless channels are the three main characteristics that multimedia communication in general and real time communication in particular require. These characteristics can be enhanced using the multiple interfaces that are already available to end devices such as 3G, Wifi and Ethernet. Current applications are not multipath capable, and hence fail to reap this extra benefit.

Multipath capability can be developed for real time communication by introducing a multipath extension for RTP that can simultaneously use network interfaces. Simultaneously using multiple network interfaces have a two-fold advantage for such applications. Firstly, the added resilience through fallback in case of network failures would help increase availability and call continuity. Secondly, in streaming scenarios, the combined throughput of multiple paths between the endpoints would allow customers to stream higher quality videos. Multipath capability can be introduced to RTP applications through extensions. Since RTP has application-level implementation, deployment of RTP extensions is quick and easy in comparison to other transport protocols that require kernel-level changes.

1.2 Objectives and Scope

Real time Protocol (RTP) is designed for transporting real time data, such as voice and video, over packet networks and is already deployed in the current Internet. An extension of RTP, capable of supporting multiple paths, would enhance quality of service and end user experience. For ease of deployment, it would be necessary that it is backward compatible and can smoothly operate within the existing network and with legacy RTP applications.

The objective of this thesis is to introduce Multipath RTP as an extension for RTP capable of utilizing more than one path between two endpoints, when available. A prototype application would be designed to study the benefits of MPRTP in video streaming networks. Testing would be conducted in a virtual environment consisting of two endpoints with multiple paths available between them. The results of the experiments will be analysed to understand the pros and cons of MPRTP.

The thesis covers MPRTP only for unicast networks. Although there are a number of use cases for MPRTP, the experiments and analysis focus primarily on video streaming applications. Some of the conclusions drawn may be extended for other use cases.

1.3 Structure of the Thesis

In the next chapter, we discuss related research in the field of multipath protocols. We also provide a background of RTP and references about previous work done for real time communication over multiple paths. In chapter 3, we discuss the motivation and objectives of MPRTP. We also discuss its architecture, packet structures and basic operation. Chapter 4 covers our implementation of MPRTP and a scheduler called RTP Adaptation for Multipath Protocols Using Percentage distribution (RAMP-UP). Results of the tests conducted are discussed in chapter 5 that also evaluates the performance of MPRTP and shows comparisons with single path scenarios. Finally, we present our conclusion in chapter 6, focusing on what benefits can be reaped from the use of MPRTP and what challenges lie ahead. A small discussion about future research works in this area is also included.

Chapter 2
Background

The concept of multihoming is being explored in various research circles and a number of protocols are being developed for introducing multipath capability in multihomed clients. The motivation for developing a multipath protocol specifically for multimedia applications is the marked difference in characteristics between real time and non-real time data. As obvious from the name given to it, real time data is time sensitive and is therefore less tolerant to delays than other kinds of data. Fortunately, when it comes to voice and video, the intolerance towards delay is compensated with a more tolerant behaviour towards loss and error. Such characteristics set real time data apart and hence have led to the development of protocols that were specifically designed for carrying real time traffic, such as RTP.

In the first part of this chapter, we will briefly discuss some multipath protocols that have been developed for the internet and some of the work that has been done for transporting real time traffic over multiple paths. In the second part, we discuss RTP protocol, as it forms the basis of our work.

2.1 Multipath Protocols

Considerable amount of research has been done previously to explore the concept of multihoming. The developments in multipath solutions, for the internet, vary and include models based on network, transport or shim layers. This section gives a brief overview of some such protocols.

2.1.1 Multipath Shim Layer Protocols

One possible approach to multihoming is the introduction of a shim layer, without any significant changes in the network and transport layers. Shim6 and Host Identity Protocol are examples of such an approach.

Shim6 [1] is a multihoming shim protocol for IPv6. This protocol enables a host with multiple IPv6 addresses (multiple interfaces) to maintain a state with its peers such that if the primary interface fails, the connection fails over to the other. Shim6 can enable a host to spread the load between different interfaces. It works just above the IP layer and is designed to have minimal impact on the transport and application layers.

The Host Identity Protocol [2] introduces a new namespace for the host machine known as Host Identifier (HI), which is based on public keys. The transport layer protocols are bound to HIs instead of IP addresses. Also, the end-point identifiers (or interface identifiers) are generated from the HI. HIP supports multihoming [3] but there is little about the simultaneous usage of interfaces in the protocol specification. Hence, it basically works for failover scenarios.

2.1.2 Multipath Transport Layer Protocols

Multipath capability within the transport layer can be seen in Stream Control Transmission Protocol (SCTP) and Multipath TCP (MPTCP). SCTP [4] was originally designed for transporting PSTN signalling over IP networks. An SCTP association is capable of failover in case of a network failure, hence providing robustness. However, it is not capable of using the paths simultaneously. SCTP has also been used for transporting RTP traffic. One research effort shows the partial reliability property of SCTP is used for retransmitting I-frames in an MPEG-4 video stream [5]. In another research work, the multihoming property of SCTP is used to provide mobility across heterogeneous networks for real time services [6].

MPTCP [7] is an extension of TCP that provides multipath capability. It pools path resources, providing robustness, higher throughput and congestion control with backward compatibility to TCP. It attains most of the goals we are trying to achieve for multipath real-time transport; however, TCP is not designed for real time data and sometimes fails to account for the requirements of such traffic. Congestion control in

TCP is provided at the cost of data rate, which is undesirable for real time data. Also, TCP error control through retransmissions increases latency, making it unsuitable for real time traffic as well.

2.1.3 Real time data over multiple paths

Multipath diversity for real time traffic and specifically RTP has been explored before. It has been shown that redundant voice traffic is transmitted over multiple paths to minimize delay, losses due to late arrival and increase voice quality [10]. The authors showed that by using multipath communication they can achieve better results than FEC protected single stream. For video transmissions, works include transmission of even and odd frames on different paths to minimize bursty loss [11].

A protocol based on partially reliable SCTP, called the Westwood SCTP-PR, for balanced multihoming of real time traffic has been presented [12]. It uses a bandwidth aware scheduler for balancing the traffic on multiple paths so that out-of-sequence packets and jitter can be minimized. When a new packet is to be sent, the scheduler decides which path would be used to transmit the packet. The scheduler, however, is not independent of the transport layer protocol.

2.2 Real Time Protocol (RTP)

Some key characteristics of voice and video data set them apart from non-real time data. These properties also make some of the popular transport protocols like UDP and TCP less than ideal for transporting such data and have led to the development of the Real time Transport Protocol (RTP) [8]. Firstly, Voice and video is transmitted as encoded samples over packet networks. Hence, the traffic is equally spaced in time. Adjusting the transmission interval to avoid congestion or for the sake of fairness is not as simple and straight forward as in non-real time data. Secondly, Human senses can tolerate a certain degree of losses in voice and video. As long as the samples are kept small, losing a single packet is not noticeable to the human eye or ear. On the other hand retransmission of lost packets would cause a lot of delay and is therefore not advised for real time communication. Instead forward error correction schemes are considered more suitable.

Order is important for voice and video data so there should be a sequencing mechanism to reorder the packets if delivered out of sequence. Furthermore, header overhead must be kept small in real time communication as voice payload is very small and a large header would lower the effective throughput. Video packets are large and require longer times to transmit, which also means that header overhead should be kept small to increase throughput.

RTP is an end-to-end protocol designed for transporting real time data across unicast and multicast networks providing sequencing with minimum overhead. It is independent of the underlying network and transport layers; however, it is designed to work best with IP/UDP stack. RTP uses Real Time Control Protocol (RTCP) for monitoring the quality of data delivery. RTP and RTCP do not guarantee quality of service and do not provide fairness.

RTP is designed to be extensible. RTP packets consist of a fixed RTP header which may be followed by an extension header which may carry any additional information required for extensions. It is widely deployed already in real time applications and has an application level implementation, making it easier to deploy extensions provided they are backward compatible with traditional RTP. All these factors contribute to making RTP suitable as the basis of a multipath protocol for real time communication.

We now briefly present an overview of RTP relevant to MPRTP and video streaming in a server/client model. Detailed information on RTP and its other use cases can be found in the RFC.

2.2.1 RTP Header Format

The fixed RTP header is shown in Figure 2.2-1. An extension header may follow the fixed RTP header. The different fields are explained below.

- *Extension bit* (X) indicates if there is an extension header included in the packet.
- *CSRC Count* (CC) is the number of CSRC identifiers that are included in the header. CSRC is defined later. Only 15 CSRCs can be identified.

- *Sequence number* is a monotonically increasing value assigned to the RTP packets for the purpose of reordering data and also to detect losses.

- *Timestamp* indicates the “sampling instant of the first octet in the RTP data packet”. It is used by the receiver to play back the received voice or video. Furthermore it is used to synchronize audio with video data.

- *Synchronization Source* (SSRC) is a randomly chosen identifier for the source of the stream. SSRC identifiers must be unique within a single RTP session.

- *Contributing Sources* (CSRCs) of the stream that is being carried in the RTP payload. CSRC identifiers indicated in the list are actually the SSRC of the individual sources of the streams that are being mixed.

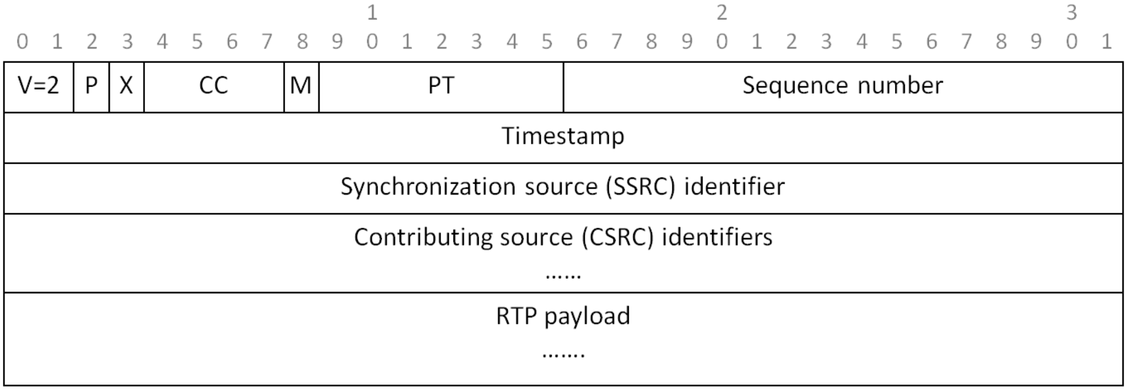


Figure 2.2-1 RTP header format

2.2.2 RTCP Receiver Report

RTP receivers send feedback reports about the quality of the data being received to the senders in the form of RTCP Receiver Reports (RR). If a receiver is also a sender, an additional sender’s block is included in the report, which is discussed in the next section.

The RR reports losses and jitter, and provides information that enables the sender to calculate RTT. The format of the packet is shown in Figure 2.2-2.

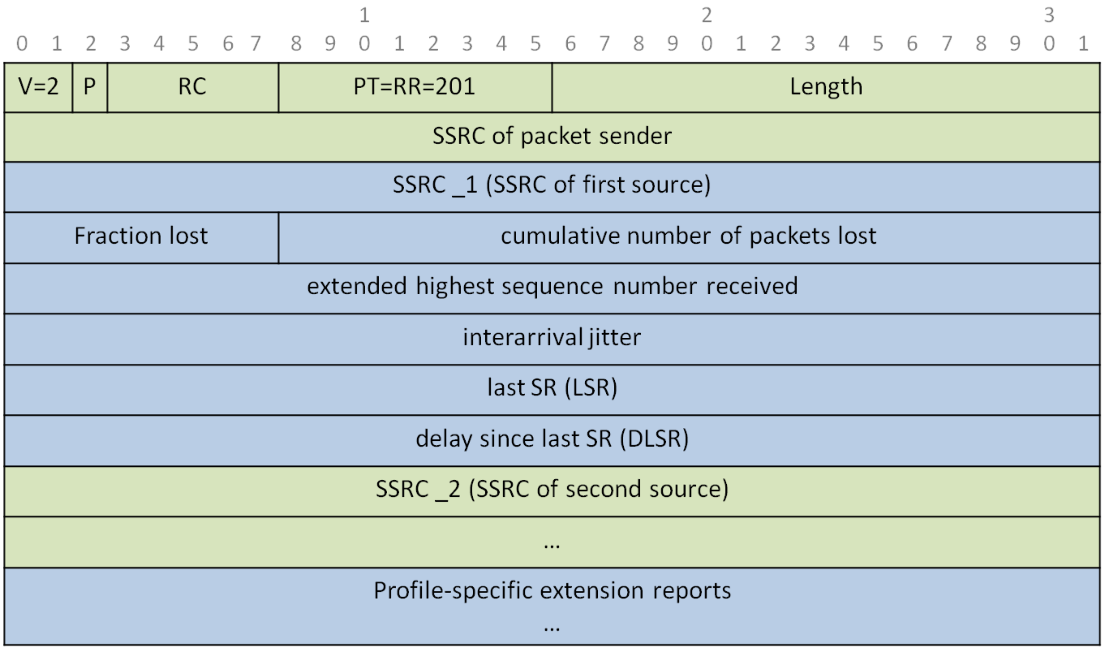

0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1
V=2
P
RC
PT=RR=201
Length
SSRC of packet sender
SSRC _1 (SSRC of first source)
Fraction lost
cumulative number of packets lost
extended highest sequence number received
interarrival jitter
last SR (LSR)
delay since last SR (DLSR)
SSRC _2 (SSRC of second source)
...
Profile-specific extension reports
...

Figure 2.2-2 RTCP Receiver Report format

- Packet type (PT) is set to 201 to indicate it is a RR.
- Synchronization Source (SSRC) of the sender of the report.
- SSRC_n is the SSRC of the source whose feedback is included in the report block.
- Fraction lost is the number of packets lost divided by the number of packets expected, since the last report was sent. If duplicates were received, the loss may be negative, in which case the field is set to 0.
- Cumulative number of packets lost is the number of packets lost since the start of the stream. It is calculated by subtracting the number of packets received from the number of packets expected. Late packets and duplicates are counted as packets received and hence this value may be negative.
- Extended highest sequence number (EHSN) consists of two parts. The low 16 bits are the highest sequence number received when the report was sent and the high 16 bits represent the number of sequence number cycles.
- Interarrival jitter is defined as "the mean deviation (smoothed absolute value) of the difference *D* in packet spacing at the receiver compared to the sender for a

pair of packets" [8]. The delay D between two packet i and j can be calculated with the following formula, where R is the arrival time of the packet in RTP timestamp units and S is the RTP timestamp of the packet

$$D(i,j) = (Rj - Sj) - (Ri - Si)$$

The jitter J is then calculated using D for two succeeding packets, i and i-1, on the basis of arrival. These packets may or may not be in sequence.

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16$$

- Delay since last SR (DLSR) is the time difference between the reception time of the last SR received from this sender and sending time of this report.

There may be profile specific extension reports that follow the report blocks in a RR.

2.2.3 RTCP Sender Report

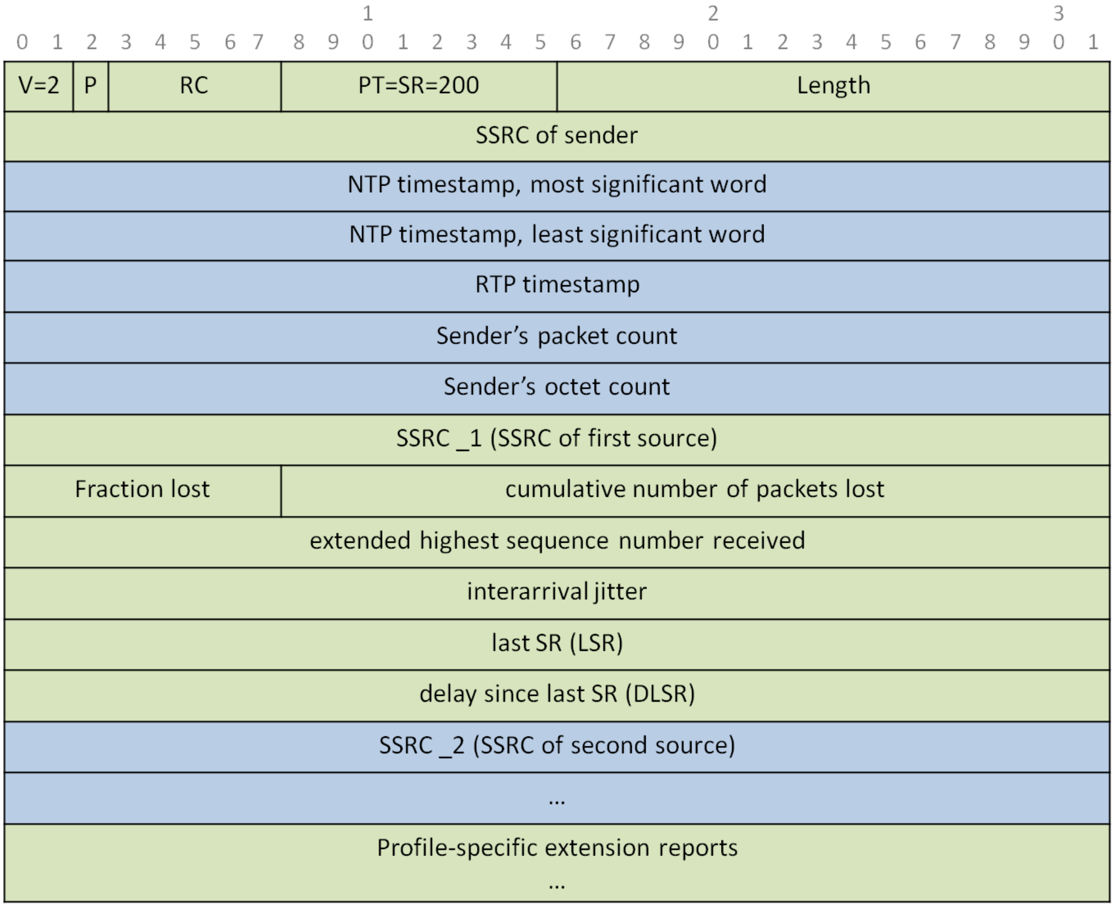
All SRs contain a 20 octet long sender's block which may be followed by receiver blocks if the sender is also a receiver. SR with a sender block is shown in Figure 2.2-3. The fields are explained below.

- NTP timestamp indicates the wall clock time when this report was sent. It is used by senders in conjunction with timestamps in RRs to calculate RTT.

- RTP timestamp is the same as NTP timestamp but in RTP timestamp units. This field is used for media synchronization, but requires that the NTP timestamps of the sender and receiver are synchronized.

- Sender's packet count indicates the number of packets sent by the sender since the start of transmission.

- Sender's payload octet count indicates the number of octets sent by the sender since the start of transmission.



V=2	P	RC	PT=SR=200	Length
SSRC of sender				
NTP timestamp, most significant word				
NTP timestamp, least significant word				
RTP timestamp				
Sender's packet count				
Sender's octet count				
SSRC _1 (SSRC of first source)				
Fraction lost			cumulative number of packets lost	
extended highest sequence number received				
interarrival jitter				
last SR (LSR)				
delay since last SR (DLSR)				
SSRC _2 (SSRC of second source)				
…				
Profile-specific extension reports …				

Figure 2.2-3 RTCP Sender Report format

2.2.4 Frequency of RTCP reports

RTP specification recommends using a minimum RTCP interval of 5 seconds, and a wait time of at least 2.5 seconds before sending the first report. The actual interval is calculated dynamically using "session bandwidth" to ensure scalability with the number of participants. So the more the participants, the less frequently the reports are sent in order to avoid flooding the network. Session bandwidth is the aggregate bandwidth, including IP and UDP header overhead, that is utilized by all the participants of a session, and is provided by the application. The application may determine this value based on the bandwidth reserved by the network for the session, or the type of codec and session. All the participants must use the same value of session bandwidth for RTCP transmission interval calculations. RTP specification recommends that the control traffic is kept at 5% of the session bandwidth. In point-to-point scenarios such as the client/server case, each member gets half of this share.

2.2.5 Jitter Buffer and Playout Delay

In RTP, the receiver must compensate for any variation in the clock rates of the sender and the receiver, network delays and out of order packets. An RTP receiver maintains an RTP jitter buffer to store the data for reordering, and removing duplicate packets as they are received. For each frame a playout time is calculated after compensating for jitter and clock skew. RTP does not provide any algorithms or approach for designing the jitter buffer or for calculating the playout time.

The playout delay is a compromise between latency and quality in an RTP stream. In non-real time video streaming, some level of latency is acceptable, depending on the application and the user preference. When the first RTP packet is received, the receiver has no knowledge of the jitter or clock skew values. The receiver would convert the timestamp of the received packet to a time in terms of the receiver clock. This value is the base time, which will be used for calculating playout time of all subsequent RTP packets. Clock skew is the difference in the clock rate of the receiver and the sender, however, while calculating skew on the receiver's side it naturally includes the effects of network jitter. Various algorithms exist for the calculation of skew. A windowed low point averaging technique [21] is used by Gstreamer[22]; a popular multimedia framework. If Tr_i is the time at which packet i is received at the receiver and Ts_i is the time at the sender then, the drift induced by delay and added noise (jitter) can be calculated as

$$\text{Drift} = (\text{Tr}_\text{i}\text{-Tr}_0) – (\text{Ts}_\text{i}\text{-Ts}_0)$$

The receiver maintains a window of delay values observed and calculates skew based on the minimum value in the window, *DriftWmin*.

$$\text{Skew} = (\text{DriftWmin} + (124 \text{ x Skew})) / 125$$

Gstreamer uses a 2 second window or 512 data points, whichever is larger. It uses a weighting factor of 125 for calculating the average. Using the minimum delay values would ensure that the skew calculations are not affected by a temporary queuing delay experienced by a few packets.

2.3 Summary

Various research groups have realized the importance of multipath protocols and different protocols have been proposed. These protocols range from shim layer solutions such as HIP and Shim6; to transport layer protocols such as MPTCP. Some research has been focused on real time data over multiple paths, however, there is little progress in the field of a multipath protocol for real time data. We feel that such a protocol holds significance, given the marked difference in properties of real time traffic.

RTP is a protocol designed for transporting real time traffic over unicast and multicast networks, which currently deployed in the Internet. RTP typically runs over UDP/IP but is independent of the lower layers. RTP supports extensibility and can be utilized as the basis for developing multipath capabilities for real time traffic. Proper extensions need to be developed for this purpose, which are discussed in the next chapter.

Chapter 3
Multipath RTP

In this chapter, we discuss Multipath RTP as an extension of RTP protocol. MPRTP adds multipath capability to RTP by allowing senders to split a single stream of data over multiple paths. It would schedule the packets on the different paths based on the network characteristics gathered using Multipath RTCP (MPRTCP) reports. Like RTP/RTCP, MPRTP/MPRTCP is independent of the transport layer; however, it is designed to work well with UDP/IP.

3.1 Motivation

Introducing multipath capability for multihomed clients may lead to increased throughput and higher resilience, directly contributing to higher quality of service in multimedia applications. Hence multimedia communication can greatly benefit from a multipath protocol designed to enhance real time transport. MPRTP is one such solution.

MPRTP is designed to allow simultaneously utilizing multiple paths between end points without any dependency on the lower layers. From an implementation perspective, it works on the application layer. Hence, it requires no change in kernel level implementations or network level infrastructure. This makes MPRTP a flexible solution for multipath scenarios of real time communication.

The primary use case of MPRTP is for streaming high-bitrate multimedia content, such as in the case of IPTV. In such a case, increased throughput can be provided and bottlenecks can be avoided if either or both of the endpoints are multihomed. Furthermore, MPRTP can be used for load balancing of the multiple paths. Another use case of MPRTP is in Voice over IP (VoIP) applications, where it helps increase resilience. The capability to recognize and use multiple paths enables MPRTP capable hosts to seamlessly switch from one path to another in case of outage or network problems. Also, if the paths are lossy, multiple paths can be used for sending redundant data.

3.2 Goals and Requirements

MPRTP applications are required to meet the following characteristics.

- Resilience can be achieved if the protocol is capable of failover. In case one of the paths goes down; it should gracefully shift to the other without disrupting communication. Packet losses can be minimized by sending redundant packets on other paths.

- The protocol should be able to achieve a higher throughput than the individual throughput of any of the available paths.

- The protocol should be able to work in today's Internet environment; it should be able to traverse middle boxes such as NATs and firewalls. This can be achieved if the subflows in the multipath protocol appear as individual RTP flows to the middle boxes.

- Also, the new protocol should be backward compatible with RTP applications. Multipath capability should be available as an option, and an MPRTP application should be able to communicate with legacy RTP applications.

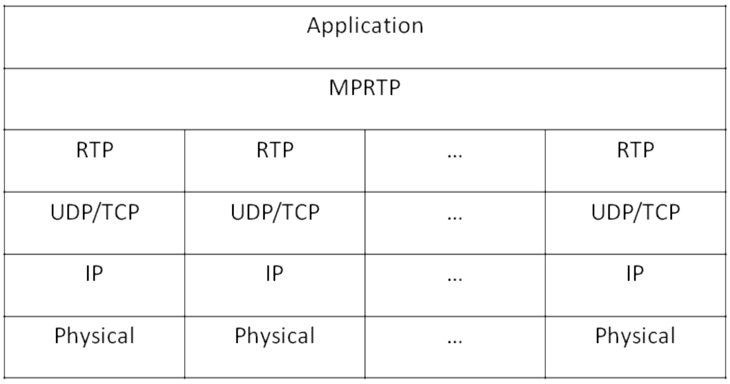
Unlike non-real time traffic, throughput is not the only parameter we are interested in when it comes to voice or video streaming. In addition to the above, MPRTP implementations would give better performance if the following rules are also met in the design.

- Data travelling on different paths will face different network delays. A good multipath protocol for real time traffic should be able to minimize the jitter experienced by the receiver.

- A late packet can be equivalent to a lost packet in real time scenarios; hence not only do we need to maximize throughput, but we also need to make sure that when using paths with different characteristics, the fraction of out-of-sequence packets is minimal.

3.3 Architecture

An MPRTP application uses multiple paths to send and/or receive data and hence must perform a dual function; transport data across each path individually as well as maintain and synchronize the multiple paths. A layer view of MPRTP is shown in Figure 3.3-1. Like RTP, MPRTP is more suited to work with UDP but should be able to operate other transport protocols as well. It may create conflicts when used with multipath transport protocols.



Application			
MPRTP			
RTP	RTP	...	RTP
UDP/TCP	UDP/TCP	...	UDP/TCP
IP	IP	...	IP
Physical	Physical	...	Physical

Figure 3.3-1 MPRTP Layer Model

The MPRTP layer is responsible for managing the available paths. It is aware of new paths becoming available or old ones failing, and a maintains path characteristics gathered through MPRTCP reports. It also initiates and records the results for connectivity checks to see the health of a particular path between peers. MPRTP senders split a single RTP stream over the multiple paths available, a process referred to as packet scheduling. The packets on any particular path constitute a subflow. MPRTP provides unique identifiers and packet sequencing for each subflow. Each subflow is represented in Figure 3.3-1 as an individual RTP flow and this is how it will appear to the network as well. The MPRTP layer acts common to all the subflows and ensures that the MPRTP session appears as a single RTP session to legacy applications. In receivers, the MPRTP layer recombines and reorders the packets coming from the multiple subflows to form a single stream for the application again. An example case with three subflows is illustrated in Figure 3.3-2.

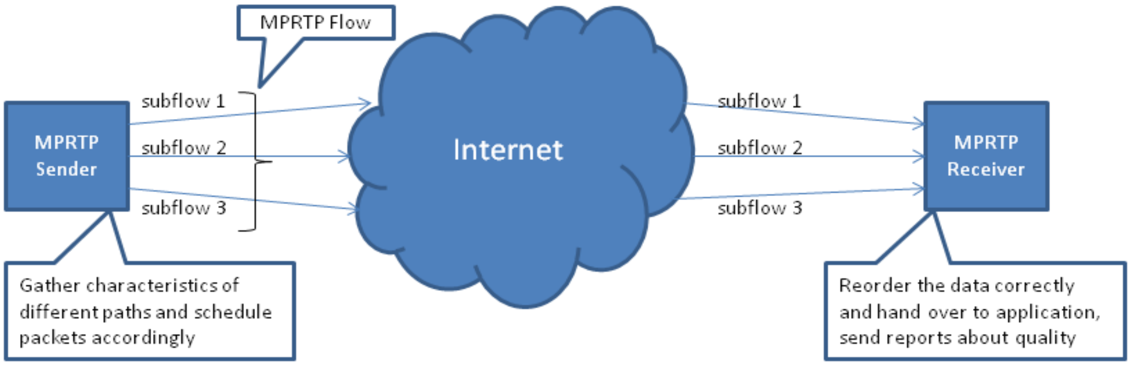

MPRTP Flow
subflow 1
subflow 2
subflow 3
MPRTP Sender
Internet
subflow 1
subflow 2
subflow 3
MPRTP Receiver
Gather characteristics of different paths and schedule packets accordingly
Reorder the data correctly and hand over to application, send reports about quality

Figure 3.3-2 Example of an MPRTP scenario in the Internet

3.4 Signalling

RTP flows do not use in-band signaling and are hence often used in conjunction with other signaling protocols such as HTTP[15], RTSP[13] and SIP[14]. MPRTP also requires an out-of-band mechanism for session setup. MPRTP may utilize Interactive Connectivity Establishment (ICE)[20] for interface discovery and connectivity checks, which can be done prior to the session setup or later on. Interface advertisements and MPRTP path information is exchanged using in-band signaling through RTP and RTCP header extension [21] for MPRTP. The MPRTP sender may choose to increase or decrease the number of paths in use or change the packet scheduling mechanisms based on this information.

It is noteworthy that starting with a single path ensures backward compatibility with RTP. MPRTP is implemented as an RTP extension and the additional information is carried in RTP header extensions. In a server/client scenario, if a client is not capable of MPRTP, it would simply ignore the MPRTP extensions in the packets received from the server. The server would recognize MPRTP silence from the client as multipath incapability and would continue using a single path for the entire session.

3.5 MPRTP Call Flow

MPRTP needs to gather information about the available paths before it can use them effectively. If ICE interface discovery and connectivity checks are completed prior to a call setup and MPRTP capability is communicated via the RTSP/SIP setup messages, it is possible for an MPRTP session to begin transmitting on all available paths as soon as the session is established. If this is not the case, and all interfaces are not known, connectivity checks are not completed or MPRTP capability of the other end is not known, a session should begin with a single stream over a single path. In accordance with the objective of this work, we discuss the call flow for a server/client scenario for video streaming as shown in Figure 3.5-1. A1 and A2 are the interfaces of the server and B1 and B2 are interfaces of the client. A1-B1 and A2-B2 are the two available paths. The steps are explained below

i. The session is established using RTSP or SIP over path A1-B1.

ii. Upon successful session setup, the server begins to transmit the video stream to the client via path A1-B1. It includes the MPRTP extension header in the stream to show MPRTP capability to the client.

iii. The client discovers a second interface B2. Interface discovery may be done using ICE. The client sends an interface advertisement to the server including address and ports for both B1 and B2.

iv. The server responds with its own interface advertisement for A1 and A2.

v. Connectivity checks between A2 and B2 are performed using either MPRTP or another protocol such as ICE.

vi. If the connectivity checks are successful, the server splits the stream to transmit on both paths A1-B1 and A2-B2.

vii. The client reorders and recombines the data received on the two paths before handing it over to the application.

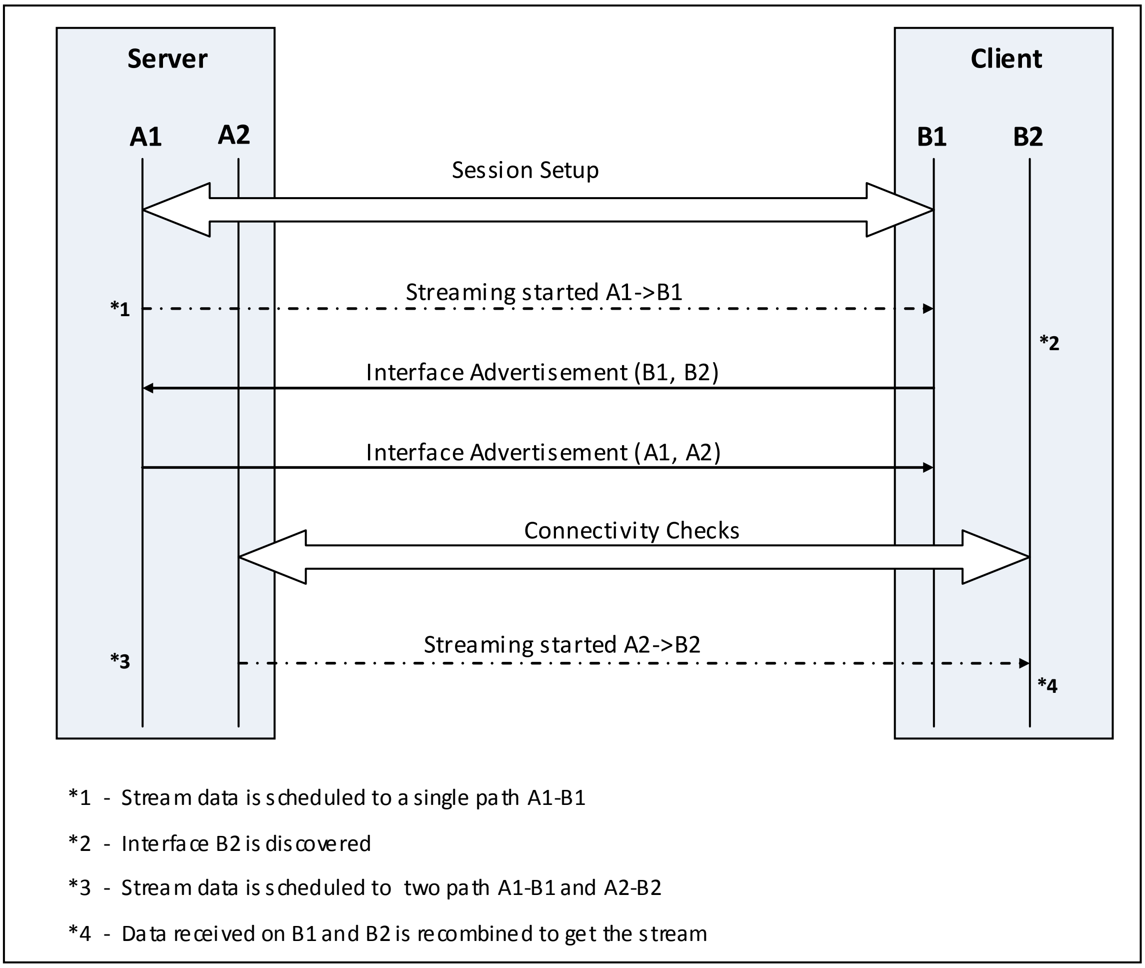

Server
Client
A1
A2
B1
B2
Session Setup
*1
Streaming started A1->B1
*2
Interface Advertisement (B1, B2)
Interface Advertisement (A1, A2)
Connectivity Checks
*3
Streaming started A2->B2
*4
*1 - Stream data is scheduled to a single path A1-B1
*2 - Interface B2 is discovered
*3 - Stream data is scheduled to two path A1-B1 and A2-B2
*4 - Data received on B1 and B2 is recombined to get the stream

Figure 3.5-1 Example of MPRTP streaming in a server/client scenario

3.6 MPRTP Message Formats

MPRTP uses RTP/RTCP header extensions for sending MPRTP-specific information to peers. As already discussed, these extensions are simply ignored by peers that do not have MPRTP capabilities. In this section, we describe the basic header extensions used by MPRTP during a session. Details of connectivity checks are not included in the scope of the thesis.

3.6.1 MPRTP Subflow Header

The MPRTP RTP extension header can be a subflow header or a connectivity check. If required, other message types can also be included in the protocol specification. A subflow header is shown highlighted in Figure 3.6-1. The fields are explained below.

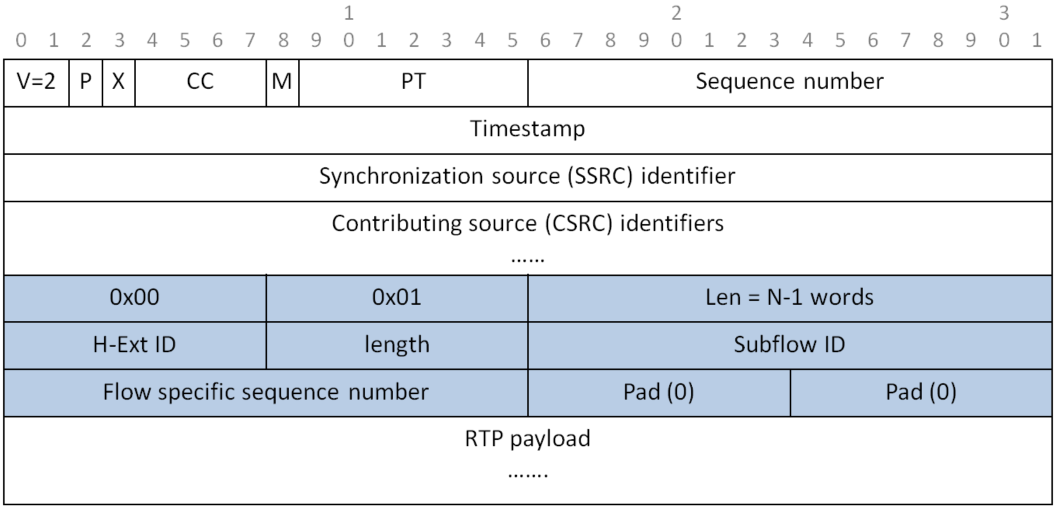


Figure 3.6-1 MPRTP subflow header

- H-Ext ID : The field indicates the type of MPRTP message and has the value 0x00 in the case of subflow header. If this field has the value 0x01, it means the message is a connectivity check.

- Length : Indicates the number of bytes in the extension header excluding H-Ext ID and the length field itself. It has value 0x06 in case of subflow headers.

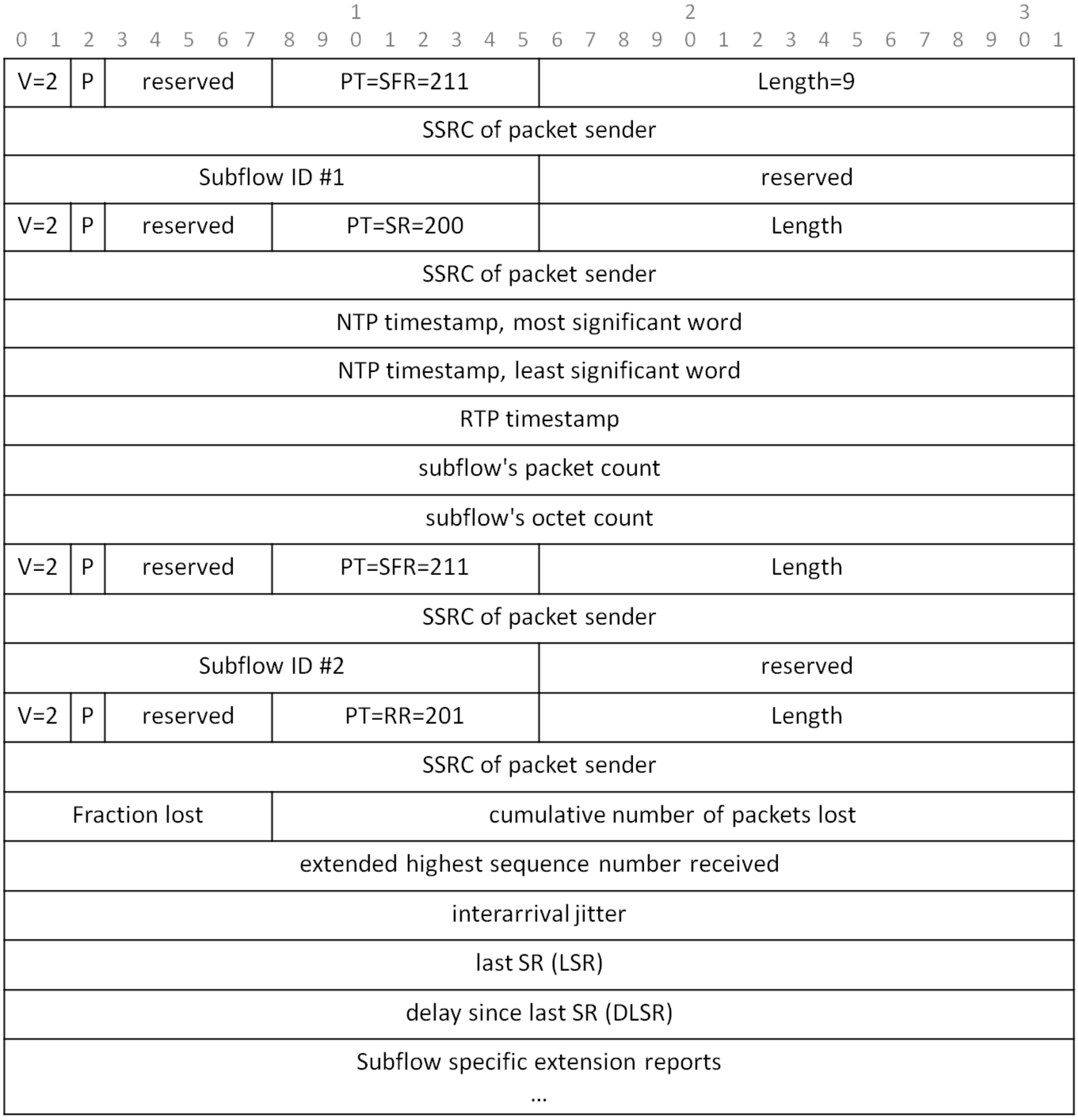
- Subflow ID : Each subflow has a unique identifier which is carried in this field.

- Flow specific sequence number : A strictly monotonically increasing sequence number assigned to each packet in the subflow.

3.6.2 MPRTCP Sender and Receiver Reports

The reports sent on a particular path will only contain subflow-specific information and hence are referred to as Subflow-specific SR (SSR), Subflow-specific RR (SRR). This ensures that the sender and receiver have information about the health and performance of each path and not just an overall value. A different approach would be to concatenate reports of various paths into a single packet and sending on all or any one single path. Concatenating reports of various subflows into a single packet would lead to larger packets, which if lost, would result in greater information loss in comparison to smaller packets of flow-specific reports. Furthermore, for RTT measurements, we have to send reports on all available paths. Since a subflow's report is only relevant as long as the path is active, it is acceptable to only send it along the same path rather than on any other path.

Subflow-specific extension reports, if any, are appended to the SRR. The fields within SRR and SSR are the same as RTCP RR and SR respectively, making them backward compatible as well as easy to implement. The message format of a MPRTCP report is shown in Figure 3.6-2.



0 1	2	3 4 5 6 7	8 9 0 1 2 3 4 5	6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
V=2	P	reserved	PT=SFR=211	Length=9
SSRC of packet sender				
Subflow ID #1				reserved
V=2	P	reserved	PT=SR=200	Length
SSRC of packet sender				
NTP timestamp, most significant word				
NTP timestamp, least significant word				
RTP timestamp				
subflow's packet count				
subflow's octet count				
V=2	P	reserved	PT=SFR=211	Length
SSRC of packet sender				
Subflow ID #2				reserved
V=2	P	reserved	PT=RR=201	Length
SSRC of packet sender				
Fraction lost			cumulative number of packets lost	
extended highest sequence number received				
interarrival jitter				
last SR (LSR)				
delay since last SR (DLSR)				
Subflow specific extension reports ...				

Figure 3.6-2 MPRTCP Sender and Receiver Reports

3.7 Frequency of MPRTCP reports

Scheduling depends on the information gathered by the sender about the path characteristics. This information is obtained using MPRTCP RRs that are received for each path. The frequency of these reports is a key factor and may directly contribute to the accuracy of the information gathered and subsequently, the effectiveness of the scheduling.

RTCP reports are only used for feedback and reporting purposes, and do not affect the quality of the stream. Hence, keeping a minimum interval of 5 seconds is understandable. In case of MPRTCP, however, we need to consider the following aspects before defining a minimum transmission interval for reports.

- In the beginning, when the sender does not know anything about the path characteristics, frequent reports would enable the sender to adjust the flow on each path more quickly and avoid sending packets on lossy or congested paths. Once the characteristics of the paths are known, less frequent reports would be acceptable.

- Each time new RRs are received, the sender must recalculate path properties and if required adjust the percentage of traffic on each path. Also, both sender and receiver would require cycles for creating and sending the reports. The higher the frequency of the reports, the more computations needed.

- If the frequency of the reports is so high that the number of packets received between two RRs is just 2 or 3, fields such as fraction lost may provide misleading information to the sender about path quality. Also, a short-lived deterioration in the network may unnecessarily cause the sender to switch traffic from the effected path.

3.8 Jitter Buffer and Skew Calculations

The role of a jitter buffer becomes exceedingly important in MPRTP. Packets travelling on different paths will experience different network delays and hence the extent of out-of-order delivery and jitter would be increased in MPRTP in comparison to the single path flows of RTP. This also increases the significance of playout delay calculations when adaptive playout is in use.

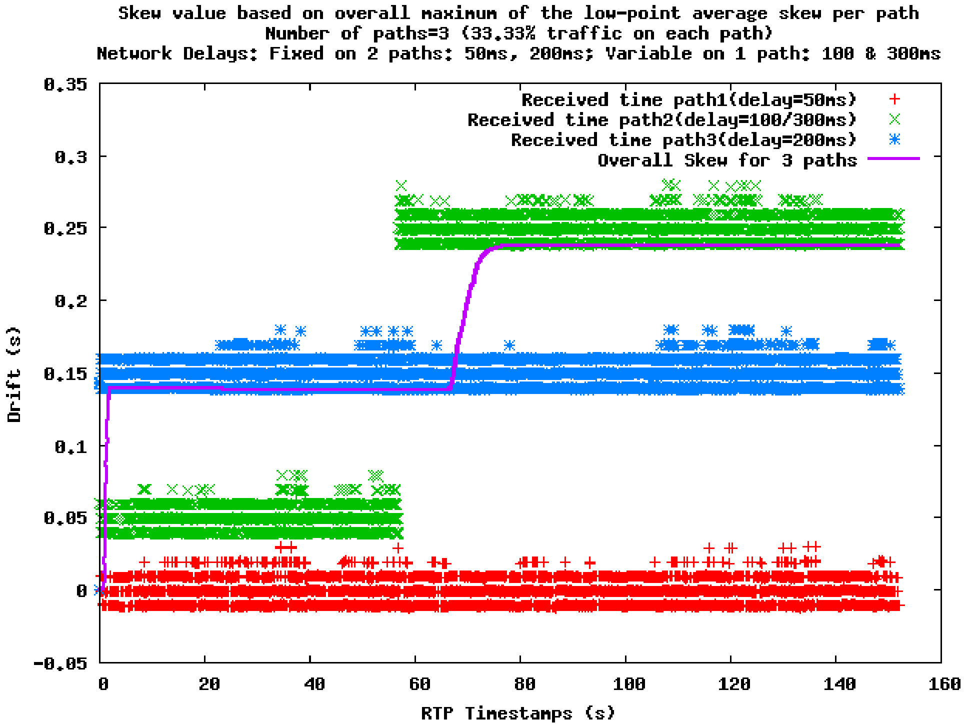


Figure 3.8-1Skew calculations based on overall maximum of per path skew values

While calculating overall skew observed by the packets flowing over multiple paths, skew on each path must be considered. A suitable approach may be to use a windowed low-point averaging for calculating the skew value for each path separately at first and then using these values to calculate the overall skew. Our tests revealed that using a windowed high-point average for calculating the overall skew from per-path skew values yields desired results when multiple paths with different network delays are used. The overall skew should be used for playout delay calculations of all the packets irrespective of the path taken to ensure that packets travelling on a slower path are not discarded.

Figure 3.8-1 shows the results of such a scheme when three paths are used with delay values of 50ms and 200ms on path1 and path3 respectively. The delay on path3 is 100ms initially, but changes to 300ms at about 55 seconds. We see that the skew value remains at 150ms for the next 10 seconds. This is because the algorithm uses a windowed low-point average for each path, and there will still be values of the previous shorter delay in the window. This approach ensures that if only a few packets are delayed, the skew value is not affected by it. The overall skew value starts increasing slowly once the older delay values are cleared. It reaches 250ms after approximately another 10 seconds have passed. The gradual increase is because we use averaging instead of absolute values in order to avoid sudden changes in the skew.

3.9 Summary

MPRTP is a protocol designed to introduce multipath capability to real time communication with the aim to improve quality of service. It aims on achieving higher throughput by resource pooling if multiple paths are available between two endpoints. It should also provide a higher level of resilience and lower packet losses in comparison to RTP under similar conditions. MPRTP is designed to be network compatible as well as backward compatible with RTP applications.

An MPRTP sender is capable of splitting a single RTP stream over the available paths, while an MPRTP receiver reorders and recombines it before handing it over to the application. Packets travelling over different paths are more prone to out-of-order delivery and higher values of jitter than those travelling over a single path, and hence MPRTP receivers may require higher buffering delays for smooth playout than RTP receivers. Possible use cases of MPRTP include high bitrate streaming scenarios and voice/video calls. MPRTP can provide higher throughput for the former case and redundancy through fallback for the latter.

Chapter 4
Implementation: A Sample Scheduler

Observing the actual behavior of RTP in the presence of multiple paths is of great significance for refining the MPRTP protocol towards maturity. Since, one of the important consequences of multiple paths is the additional throughput, video streaming applications demanding higher bandwidth were chosen as the starting point for the experimentation.

We developed a MPRTP sender based on our sample scheduler called RAMP-UP (RTP Adaptation for Multiple Paths Using Percentage distribution). RAMP-UP is designed specifically for video streaming from a server to a client. It focuses on scheduling of packets over the multiple available paths in order to achieve higher efficiency. We also implemented a simple MPRTP receiver with buffering considerations for MPRTP. It does not, as yet, cover aspects of interface discovery nor connectivity checks.

4.1 Design Decisions

Regardless of how efficient the scheduling of an MPRTP sender is, if a single lossless path is available with sufficient capacity, using a single RTP flow on this path would outperform using multiple subflows on multiple paths. The reason for this is that the packets on a single path suffer similar network effects and would lead to a higher level of order and lower values of jitter. On the other hand, sending packets over different paths with varying network characteristics, would, in most cases, lead to out-of-order delivery and high jitter values. Also, extra computations and memory would be required for managing the paths, scheduling at the sender and reordering at the receiver. The header overhead for MPRTP would also be introduced. Nevertheless, the use of multiple paths would still trump a single path if, for instance, none of the paths available provide sufficient throughput. Furthermore, the reliability and redundancy factor is also added when more than one path is used.

Our design assumes that a single path cannot meet the bandwidth or reliability requirements for the traffic, and the use of multiple paths is necessary. This assumption is of little consequence, since MPRTP can be controlled to some extent by the application, and it may be possible for the application to assign a preferred path in the scenario where such a path exists. For now, RAMP-UP does not discover paths nor does it perform connectivity checks or interface advertisements. The maximum number of paths to be used, along with the IP and port combinations must be fed manually to both sender and receiver. This state can be achieved as a result of ICE exchange in a real scenario.

RAMP-UP focuses on avoiding congested or lossy paths, however it does not aim to load balance the paths. As described in Chapter 2, RTP is not fair because of the periodic nature of real time data. Although in the presence of multiple paths, an element of fairness may be added to the protocol by shifting load to other paths, but we leave this problem to be looked into as part of future work.

There is no distinction between congestion losses and other types of losses in the design. The assumption is that a path exhibiting packet loss is experiencing congestion. This in turn implies that the scheduler may act unpredictably over lossy paths, where the losses are error-induced.

Finally, since MPRTP leads to out of order delivery when paths have different latencies, there should be a method of buffering to compensate for this. The buffering can be done either at the sender or the receiver end. When done at the sender end, the sender can send future packets on slower paths. However, this approach would require foresight and would only be successful if the sender has sufficient information about the paths, and the path characteristics are stable. Sender-side buffering may reduce out of order delivery and jitter, but would not be able to eliminate the need for receiver-side buffering completely, given the unpredictable nature of packet networks. In our solution, we used buffering at the receiver with a fixed playout delay as well as an adaptable skew factor. This design is beneficial because the receiver can more quickly adapt to changing path characteristics as it gets the information first hand, and unlike the sender does not have to wait for the RTCP RR. We describe the buffering in more detail in the next section.

Development was done using C/C++ application-level programming and uses threads for sending and receiving data on each path. A parent thread is responsible for scheduling packets in the sender and recombining the sub-streams in the receiver. The sender implements only MPRTP and does not have an in-built encoder or decoder.

4.2 MPRTP Receiver

An MPRTP receiver is responsible for recombining the data received on the various paths to produce a single stream of packets for the application. In our implementation, we use a shared jitter buffer that not only removes jitter but also serves to recombine the sub streams. A configurable playout delay is included to ensure smooth playback. As discussed already in the previous section, MPRTP packets may suffer from higher jitter and require greater reordering at the receiver, and hence would probably require larger playout delays in comparison to RTP receivers.

Figure 4.2-1 shows a graphic representation of the jitter buffer when two paths are in use. Each packet is inserted into the jitter buffer as soon as it arrives, unless the playout time for the received packet has already expired, in which case it is considered late and is discarded. All received packets of a frame are handed over to the application at playout time and not before.

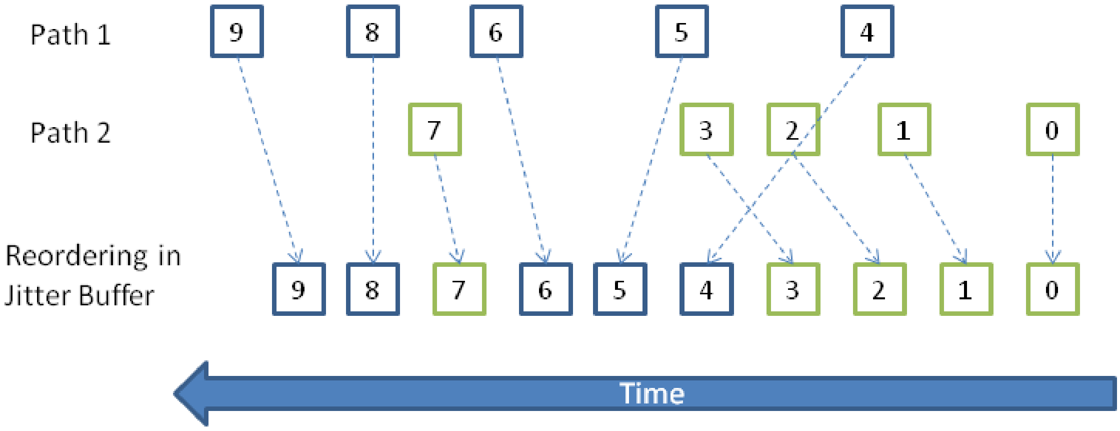


Figure 4.2-1 MPRTP receiver jitter buffer for reordering

4.3 MPRTP Sender

The MPRTP sender is designed to distribute the data on the paths available between the receiver and itself. It uses the MPRTCP reports to determine the characteristics of each path and accordingly assigns a percentage of total traffic to be sent on that path. Using a percentage distribution technique has two advantages over a per-packet decision approach. Firstly, it eliminates flapping, which means the traffic is shifted continuously from one path to the other. This happens if the sending bitrate is higher than the available capacity and increasing load on any of the paths forces it to go into congestion, causing the sender to constantly switch routes to avoid losses. Secondly, if the bitrate of the stream is increased or decreased, the change would be equally distributed on all paths.

RAMP-UP sender assigns equal percentages to all available paths at startup; for two paths it would assign 50% to each. Every time a new packet arrives, it is assigned to a particular path depending on the percentage distribution. All packets from a single frame are sent on the same path. Since the sender is not capable of reducing the video bitrate, if the available bandwidth is not sufficient for the stream, losses cannot be avoided.

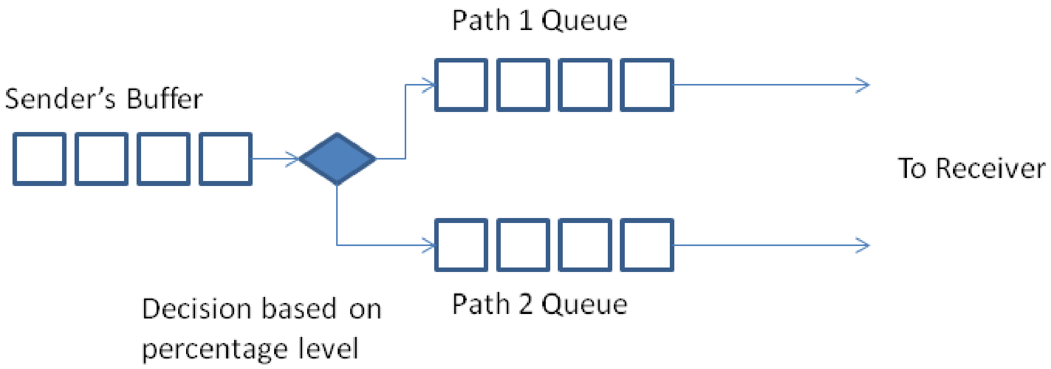


Figure 4.3-1 RAMP-UP Sender's queue

The traffic percentage controls the number of bytes sent on the path and not the number of packets. We use a probabilistic approach based on random number generation, which adds more freedom in allocating percentage shares. This approach also results in even offered load for different sized packets. The effective percentage of each path is adapted according to the amount of traffic that has already been sent on that path. The path to be used is decided whenever a new packet arrives. Given the following parameters for a path *Z*

p1 = ratio allotted to the path

b = Bytes already sent on the path

t = total bytes received (including the packet that is yet to be transmitted)

p2 = b/t

Then the effective percentage *p* of path *Z* is *p1* minus *p2*; and the path will be selected for transmission if the random number *r* is less than or equal to *p*, where *r* lies between 0 and 1.

4.4 Scheduling Algorithm

The RAMP-UP scheduling algorithm is based on the assumptions discussed in section 4.1 and is designed for video streaming applications where more than one path exists between the server and the client. It is currently only capable of handling unicast applications, though the design may be extended to multicast. The algorithm uses a non-aggressive approach, which implies that we do not put more traffic on a path unless necessary. This in turn implies that the full capacity of certain paths may never be known. Hence, the algorithm only distributes load and does not regulate it to provide congestion control.

4.4.1 Bitrate Measurements

The scheduler uses the RRs to calculate path bitrates which it uses for assigning percentages. When the sender receives the *i-th* RR on a path *j*, it calculates the instantaneous bitrate $B_{i,j}$ for that path, using the following formula

$$\mathrm{B_{i,j}} = \frac{(\mathrm{HSN_{i-1}} - \mathrm{HSN_i}) \times (1 - \mathrm{L_i}) \times \mathrm{S_i}}{\mathrm{t_i} - \mathrm{t_{i-1}}}$$

where HSN is the highest sequence number and L is the fractional loss observed in the RR. The time of reception of RR is denoted by t and S is the average size of the packet during the interval t_{i-1} to t_i. Figure 4.4-1 illustrates the concept graphically.

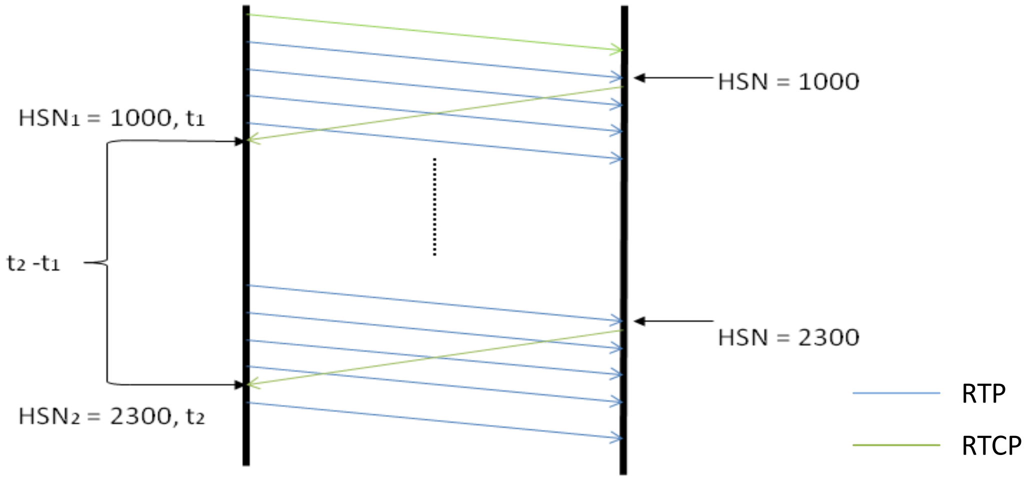


Figure 4.4-1 Measurements for calculating bitrates

Based on the instantaneous bitrate the scheduler calculates two values that it uses for actually calculating the percentage distribution. These are listed below.

- The Tested bitrate $TB_{i,j}$ is the highest bitrate observed on a path, after taking into account the losses. It is calculated using the following formula, where $0 \leq \alpha \leq 1$.

$$\mathrm{TB_j} = \begin{cases} \mathrm{TB_j} & \text{if } \mathrm{Li} = 0, \mathrm{B_{i,j}} < \mathrm{TB_j} \\ \alpha\mathrm{TB_j} + (1-\alpha)\mathrm{B_{i,j}} & \text{else} \end{cases}$$

- Congested bitrate $CB_{i,j}$ is calculated when there are losses on the path, and low bitrate values are being observed. This value is used in conjunction with the congestion indicator *CI* and the congestion time *Ctime*. *CI* is an integral value indicating the likelihood that a path is in congestion, and a path is considered congested if *CI* is equal to *CIMAX*. *CI* is incremented by 1 when losses are observed. *Ctime* is the absolute time when *CI* was last modified or when losses were detected, whichever is more recent.

Figure 4.4-2 RAMP-UP scheduler: Flowchart for calculating bitratesillustrates the principle graphically.

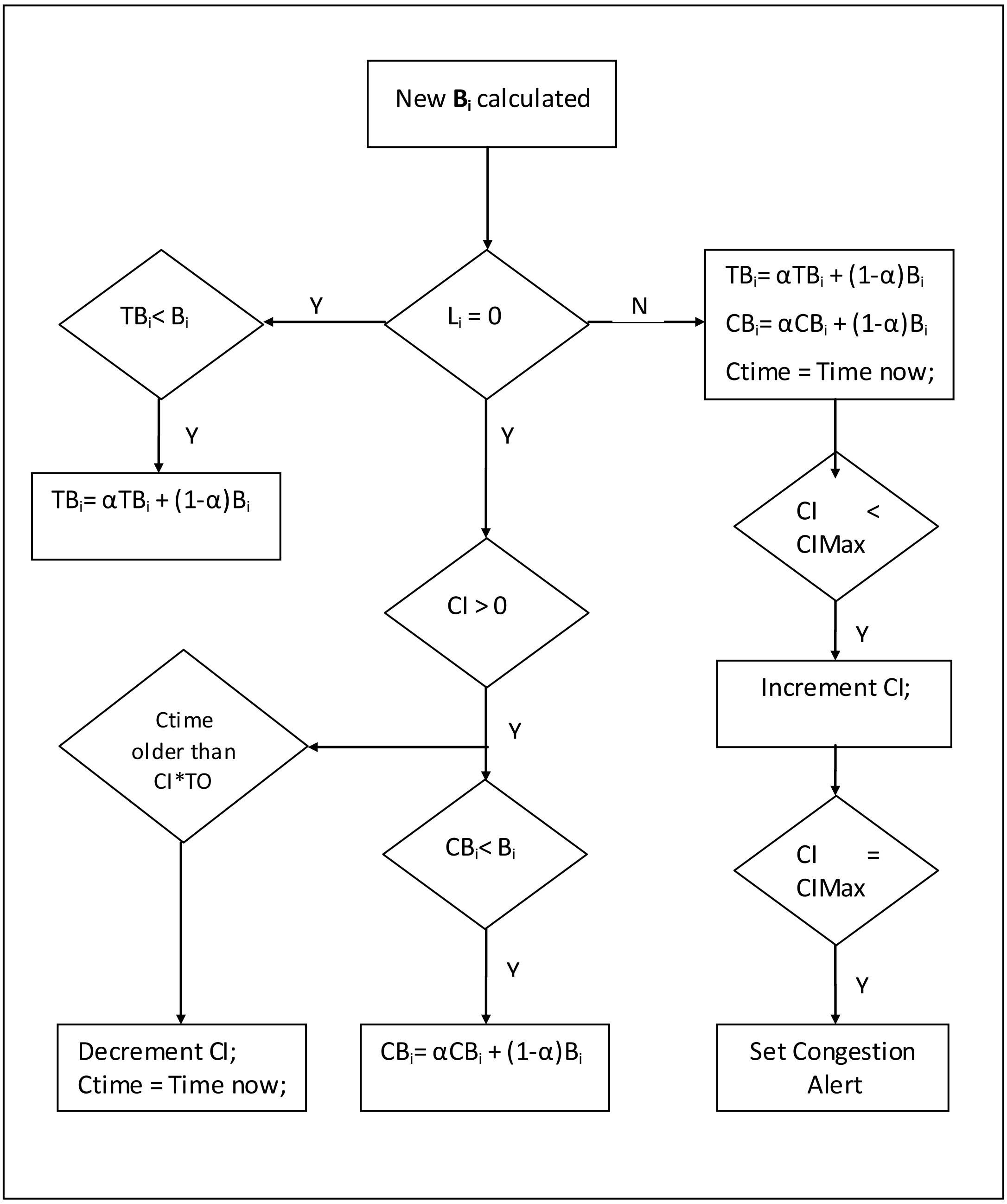


Figure 4.4-2 RAMP-UP scheduler: Flowchart for calculating bitrates

4.4.2 Calculating Percentage Distribution

The scheduler will reassign percentages to each path based on the new set of bitrates. The formula used by the scheduler for this calculation depends on the number of routes that are congested or lossy. We consider a path with *CI* = *CIMAX* congested, however, if the *CI* has a value greater than 0, but has not yet reached *CIMAX*, we declare it lossy but not yet congested. This is to allow room for temporary losses to clear on their own. If the losses are being observed continuously, and *CI* reaches *CIMAX* only then would the scheduler drastically decrease traffic on the route.

If none of the paths are congested then the assigned percentage is the ratio of the path's *TB* to the total *TB* of all the paths combined. When all paths are congested the assigned percentage is the ratio of a path's *CB* to the total *CB* of all the paths combined. If *c* is the number of paths that are currently congested, *l* are the number of lossy paths and *n* is the total number of paths available then the percentage p_j that will be assigned to path *j* is calculated as follows.

$$\text{if } c = 0\,, \qquad p_j = \frac{TB_j}{\sum_{i=0}^{n} TB_i}$$

$$\text{if } \; c = n, \qquad p_j = \frac{CB_j}{\sum_{i=0}^{n} CB_i}$$

If some paths are congested, while others are not i.e $c < n$, we use a stepwise approach.

STEP I : Assign percentages to the congested routes

For all j such that $CI_j = CI_{max}$ and $0 \leq \beta \leq 1$.

$$p_j = \mathrm{Min}\left\{\frac{TB_j}{\sum_{i=0}^{n} TB_i}, \frac{\beta CB_j}{SB_i}\right\}$$

The assigned percentage is the minimum of two terms. The first term is simply the ratio of the path's *TB* to the total *TB*. The second term is based on the congested bitrate of the path, which is the bitrate the path exhibits during the congestion. The denominator is the sending bitrate SB_i, which is the current average bitrate of the stream updated after every second. Assigning ratio CB_j/SB_i on the congested path j would ensure that the bitrate on the path j is equal to CB_j. However, this holds true only if SB_i remains constant. For variable bitrate, we needed to lower the assigned ratio further to keep the traffic within bounds of the congested bitrate. Also, since we have uncongested paths, we prefer to keep the ratio of traffic on the congested paths low. Finally, if the ratio of the path's *TB* to the total available *TB* is lower than the second term, then we have enough bitrate available on other paths and we don’t need to put extra traffic on the congested path.

STEP II: If there is at least one path that is neither lossy nor congested then assign percentages to the lossy routes

For all j such that $0 < CI_j < CI_{max}$ and $0 \leq \gamma \leq 1$.

$$p_j = \mathrm{Min}\left\{\frac{TB_j}{\sum_{i=0}^{n} TB_i}, \frac{\gamma CB_j}{SB_i}\right\}$$

This formula is similar to the one used in step I, with the exception of the variable γ instead of β. The value of γ can be higher or equal to β, so that lesser traffic is routed towards congested routes in comparison to lossy ones.

STEP III : Assign percentages to the remaining routes

For all j such that p_j has still not been assigned, these would be the lossy paths if there is no path that is neither lossy nor congested.

$$p_j = \frac{TB_j}{\sum_{\text{unassigned}} TB} \times (1 - AP)$$

where *AP* is the assigned percentage i.e. the sum of the percentages that has been assigned in Step 1 and 2. In this step we assign the percentage remaining after step I and II on the remaining paths. Hence more traffic is shifted to the paths with no losses, but even the paths with losses still get assigned some traffic.

Figure 4.4-3 illustrates the complete principle of percentage distribution with the help of a flowchart.

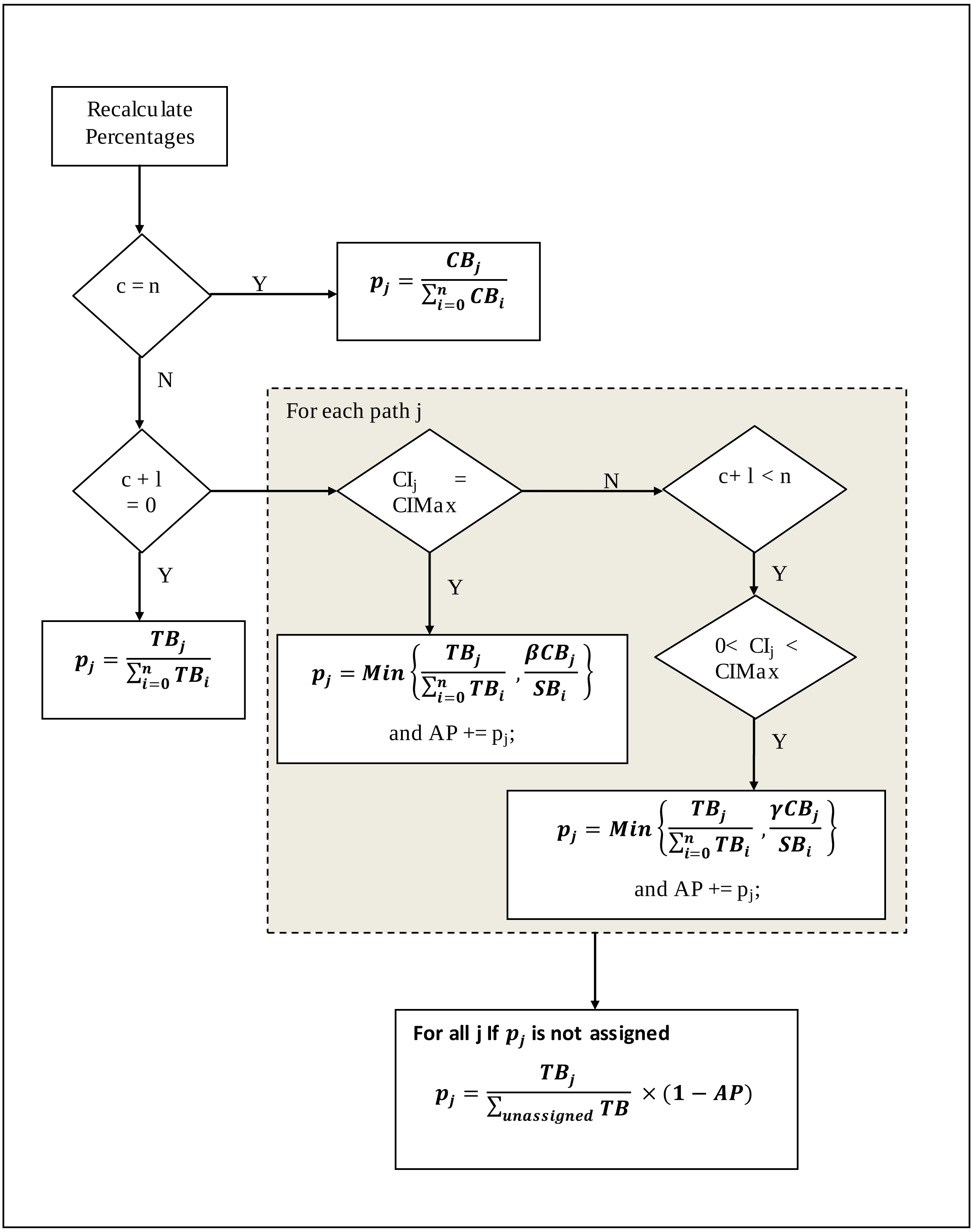

Recalculate Percentages
c = n
Y
$p_j = \frac{CB_j}{\sum_{i=0}^{n} CB_i}$
N
For each path j
c + 1 = 0
CI_j = CIMax
N
c+ 1 < n
Y
Y
Y
$p_j = \frac{TB_j}{\sum_{i=0}^{n} TB_i}$
$p_j = Min\left\{\frac{TB_j}{\sum_{i=0}^{n} TB_i}, \frac{\beta CB_j}{SB_i}\right\}$
and AP += p_j;
0< CI_j < CIMax
Y
$p_j = Min\left\{\frac{TB_j}{\sum_{i=0}^{n} TB_i}, \frac{\gamma CB_j}{SB_i}\right\}$
and AP += p_j;
For all j If p_j is not assigned
$p_j = \frac{TB_j}{\sum_{unassigned} TB} \times (1 - AP)$

Figure 4.4-3 RAMP-UP scheduler: Flowchart for calculating percentage distribution

4.4.3 Frequency of Redistribution

Calculating percentage distribution and redistributing traffic, takes up processing and time. Hence, it is important to find an optimum interval for this action to take place. A few factors that are of significance in this matter are listed below

- In the beginning, the scheduler does not have any information about the path properties and distributes the traffic equally on all paths which may not always be an ideal distribution. It would be better to redistribute the traffic in a better manner, as soon as some information is received about the paths.

- When there are losses on a path, changing the distribution quickly would prevent high loss rates. However, the losses could be a temporary condition, and reacting too quickly might result in an unnecessary shift of traffic.

- When the paths are stable and enough information is known about them, the traffic distribution does not need to be revised too often.

In RAMP-UP, we recalculate percentages on the expiry of the reschedule interval, *r_int*, defined in seconds. It is calculated by the following formula

$$r\_int = r\_rec \times (rand + 0.5)$$

where R_INT_MIN ≤ r_rec ≤ R_INT_MAX for normal operation and is called the reschedule recovery. The randomization is to prevent synchronized rescheduling of multiple senders with common paths. r_rec is set to R_INT_MIN at startup. It is incremented with each recalculation until it reaches R_INT_MAX. Deviation from normal operation is when there is a congestion alert and the r_rec is set to zero so that the scheduler can redistribute traffic without any further delay. R_INT_MAX and R_INT_MIN can be set based on the characteristics of the available networks. For instance, if the application uses paths that have rapidly changing characteristic such as 3G or GPRS, then the reassignment should be scheduled quickly and R_INT_MAX should be low. However, networks that are usually stable in terms of bitrate can use higher values of R_INT_MAX. R_INT_MIN should be large enough that the rescheduling occurs after at least one set of MPRTCP RRs have been received and the information regarding path properties has been updated.

4.5 Summary

RAMP-UP is a basic first attempt to building a scheduler for a MPRTP sender. The purpose of designing such an application was to be able to experiment with MPRTP in a multipath environment. The results of such experiments may prove to be beneficial in the development and subsequent deployment of the protocol.

The RAMP-UP sender uses RTCP RRs to estimate the bitrate on each of the available paths and assigns traffic percentage to the paths based on these values. The receiver buffers the incoming packets, reorders and recombines the sub streams on the different paths and hands it to the application.

Our implementation of MPRTP sender and receiver is developed for video streaming scenarios; however, it may be extended at a later point to include other use cases of MPRTP. For the time being, the available paths are fed manually in the form of IPv4 addresses and ports. Path discovery through ICE or other means may also be incorporated at a later stage.

Chapter 5
Testing and Results

In this chapter, we evaluate the performance of MPRTP through RAMP-UP. Our evaluation includes experiments designed specifically to test our own algorithm design, as well as those that effectively simulate real life cases in which MPRTP may prove to be helpful for multihomed devices such as smart phones and tablets with fixed and wireless connections. In order to effectively evaluate performance with respect to particular path characteristics, we are sometimes forced to make simplistic assumptions which may not be present in real networks.

5.1 Evaluation Environment

For the evaluation, we set up a virtual environment consisting of a sender and a receiver, having three interfaces each. Three paths are available between the sender and the receiver via virtual routers as shown in Figure 5.1-1. All virtual machines run on the same physical machine.

Network characteristics were emulated on the paths using NetEm [27]. Various tests were conducted to observe the performance of MPRTP in general and RAMP-UP under different network conditions. Path latency, bandwidth and losses were emulated during the testing.

The RAMP-UP sender reads RTP packets saved in an rtpdump file created using rtp tools [19]; it adds the MPRTP header to the packets and sends them across the network. The MPRTP receiver reorders the received packets and writes them to an rtpdump file which is later used for analysis. Furthermore, other statistical data such as time of arrival of packets, the path taken, observed bitrates and losses are measured and recorded at the receiver as well. We chose to use H.264 constant bitrate (CBR) videos for the testing.

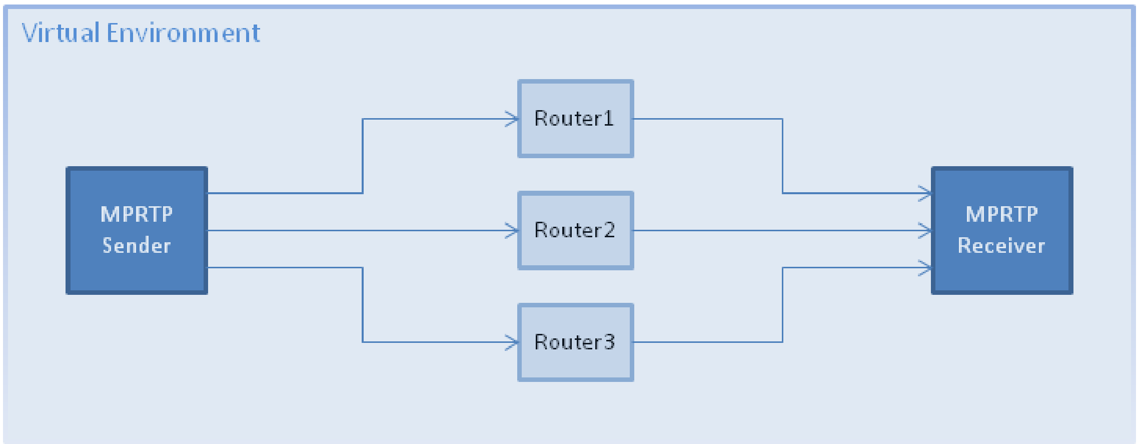


Figure 5.1-1 Virtual environment for testing

5.2 Test parameters

The "Foreman" video sequence [28] is used for the testing. It is pre-encoded using Nokia's H.264 encoder [21] at an average media rate of 1 Mbps, 30 FPS and GOP=16 and the video sequence is 265 seconds long. The instantaneous bitrate is shown in Figure 5.2-1.

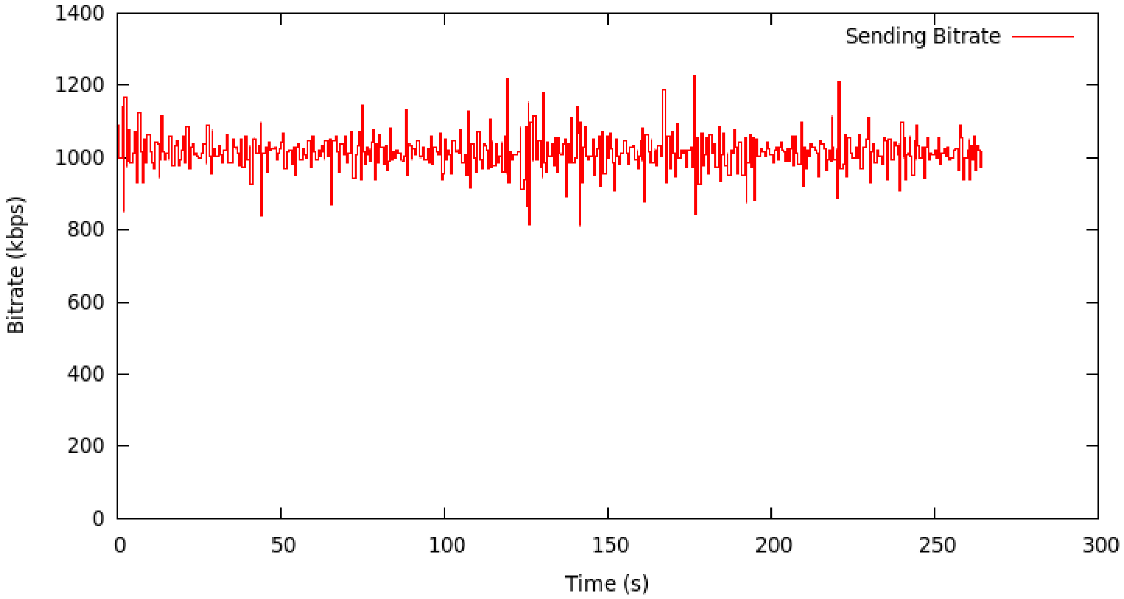


Figure 5.2-1 Instantaneous bitrate of the test video stream

The values assigned to the different parameters of the scheduler during the testing are shown in Table 5.2-1. The values were chosen after careful consideration to the evaluation environment and were tweaked to match our needs after some basic experimentation. Experiments to study the effects of each variable under different test scenarios was not done, instead the focus was kept on evaluating the algorithm as a whole under different path characteristics.

Parameter	Value
α	0.99
β	0.5
γ	0.75
CIMAX	3
R_INT_MIN	3 seconds
R_INT_MAX	10 seconds

Table 5.2-1 Test parameter values

In our testing, we use a minimum interval of 500ms for sending MPRTCP reports instead of 5 seconds. Since, we are using high bitrate streams; the ratio of MPRTCP reports is still within 5% of the session bandwidth. A more adaptive approach for calculating this interval may also be used.

5.3 Test Results

In this section we evaluate the performance of our algorithm through metrics such as Peak Signal-to-Noise Ratio (PSNR) and percentage loss rate. We also observe the traffic distribution assigned by the scheduler over the course of the time. In order to maintain objectivity for the reader and to avoid discussing cases that yield similar conclusions, we present here the results of only some of our experiments.

5.3.1 Paths with similar properties

In the first scenario, we used 2Mbps paths with 50ms path delay values. We repeated the experiments with different loss rates on the paths. Table 5.3-1 shows the results for lossless paths. It shows that the PSNR value remains the same regardless of the number of paths used. Table 5.3-2 and Table 5.3-3 show results when loss rates were introduced on the paths. It can be seen that similar PSNR values were observed for single and multiple paths.

Table 5.3-1 PSNR comparison; when paths have equal capacity & delay and no losses

Scenario	PSNR		Percentage
All paths have 0% losses	Average	StdDev	packet loss
Single Path	48.4274	0.0000	0.0000
2 paths using RAMP-UP	48.4274	0.0000	0.0000
3paths using RAMP-UP	48.4274	0.0000	0.0000
2paths using static distribution	48.4274	0.0000	0.0000
3paths using static distribution	48.4274	0.0000	0.0000

Table 5.3-2 PSNR comparison; when paths have equal capacity & delay and 0.5% losses

Scenario	PSNR		Percentage
All paths have 0.5% losses	Average	StdDev	packet loss
Single Path	41.8868	0.5059	0.4873
2 paths using RAMP-UP	40.3142	0.5763	0.5051
3paths using RAMP-UP	40.4063	0.8492	0.4944
2paths using static distribution	40.9122	0.1908	0.4916
3paths using static distribution	40.4834	0.7529	0.4852

Table 5.3-3 PSNR comparison; when paths have equal capacity & delay and 1% losses

Scenario	PSNR		Percentage
All paths have 1% losses	Average	StdDev	packet loss
Single Path	36.1726	0.7050	1.0059
2 paths using RAMP-UP	36.5637	1.0059	0.9391
3paths using RAMP-UP	36.2120	0.5717	0.9952
2paths using static distribution	36.4890	0.8504	1.0165
3paths using static distribution	36.2855	0.4991	1.0797

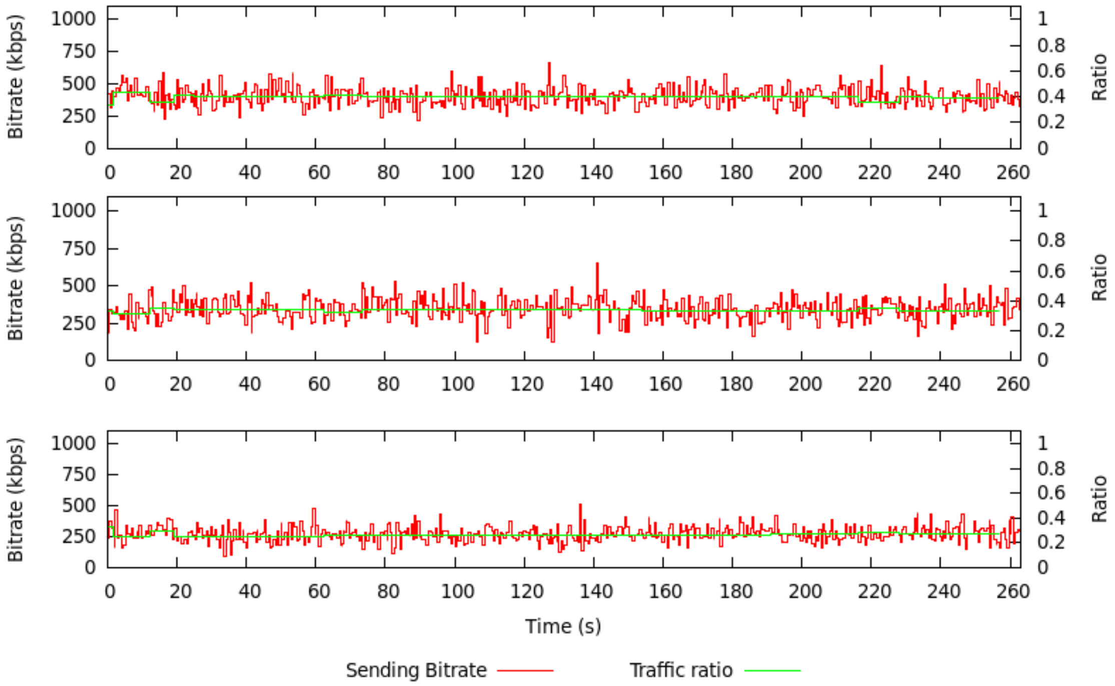


Figure 5.3-1 RAMP-UP scheduler's percentage distribution over lossy paths

Since RAMP-UP, by design, interprets losses as indication for congestion, we did not expect it to act ideally during these experiments. As expected, the behavior of the scheduler is not uniform when the paths have losses. We explain in the previous chapter that the scheduler declares a path congested if the losses are observed close together in time. When the loss rate is low, the losses were spread apart in time, and hence the scheduler does not drastically lower the traffic share of any path. Figure 5.3-1 shows the results of an experiment where losses were 0.5% and the scheduler did not declare any path congested. The scheduler maintains an almost uniform distribution of traffic over all paths throughout the course of the experiment. The sending bitrate is the instantaneous bitrate of the sent stream as measured by the sender.

5.3.2 Paths with different latencies

In our second set of experiments, we use paths that have the same bandwidth but different latencies. The difference in latencies would result in out-of-order packets, which are put in order using the jitter buffer.

Table 5.3-4 shows results of when we use a playout delay of 1 second at the receiver end. As long as the playout delay is greater than the difference between the latencies, the received packets can be reordered on reception. In practice, some processing time is needed at the receiver end, which must also be compensated in the playout delay along with the path latencies.

Table 5.3-4 PSNR comparisons; when paths have different latencies

Scenario	PSNR	
All paths have 1Mbps capacity, Latency for path1 50ms, path2 100ms, path3 200ms	Average	StdDev
Single path (all paths)	48.4274	0.0000
2 paths using RAMP-UP	48.4274	0.0000
3paths using RAMP-UP	48.4274	0.0000
2paths using static distribution	48.4274	0.0000
3paths using static distribution	48.4274	0.0000

The initial bitrate measurement on a slower path would give a slightly lower bitrate due to the delay in receiving the packets, hence lowering the traffic share to a small extent. However, this effect would diminish in later measurements. Figure 5.3-2 shows the traffic distribution for three paths.

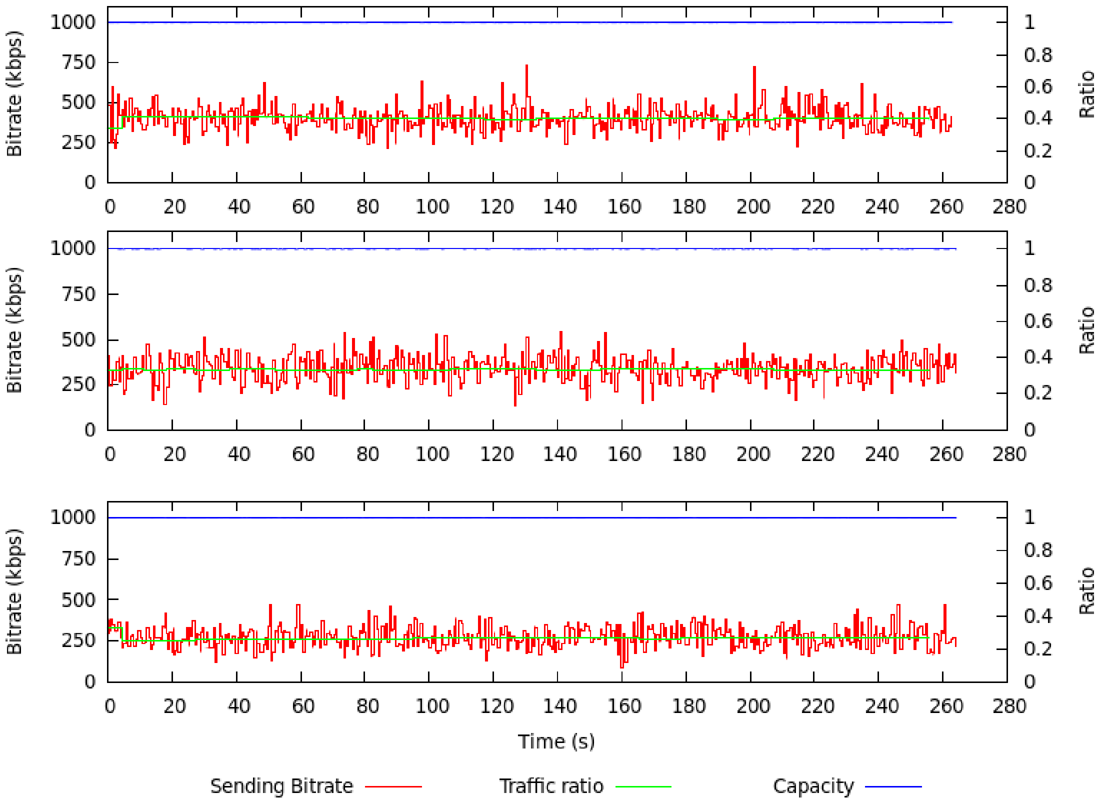


Figure 5.3-2 RAMP-UP behaviour when paths have different latencies

5.3.3 Paths with different loss rates

RAMP-UP is not designed to cope with error-induced losses and sees all lossy paths as congested. It is still important to see its behavior in such an environment. We used three paths in this experiment, all having 2Mbps bandwidth and 50ms latency. The losses were set to 0 on path1, 0.5% on path2 and 1% on path3. Again, RAMP-UP does not declare congestion on any path and hence the traffic share on each path does not change much as can be seen in Figure 5.3-3. The PSNR values and percentages losses are shown in Table 5.3-5.

Table 5.3-5 PSNR comparisons; when paths have different loss rates

Scenario	PSNR		
All paths have 2Mbps capacity and 50ms latency, Loss rate for path1 0%, path2 0.05%, path3 1%	Average	StdDev	Percentage packet loss
Single Path (path1)	48.4274	0.0000	0.0000
Single Path (path2)	41.8868	0.5059	0.4873
Single Path (path3)	36.1726	0.7050	1.0059

2 paths using RAMP-UP (path1, path2)	43.3902	1.9475	0.2422
3paths using RAMP-UP	40.4923	0.4918	0.4786

The results for RAMP-UP with multiple paths are much better than using single path RTP over path3 and are comparable with single path RTP over path2. The reason is obviously that the amount of traffic on the lossy paths is automatically decreased when multiple paths are used and hence overall percentage loss is also decreased. In a scenario such as this, where a lossless path is available; the best approach would be to use the lossless path if there is enough bandwidth. A legacy RTP application would have no way of knowing which path is lossless, its choice of path would be random and it might be just as likely of picking a lossy path as picking the lossless one.

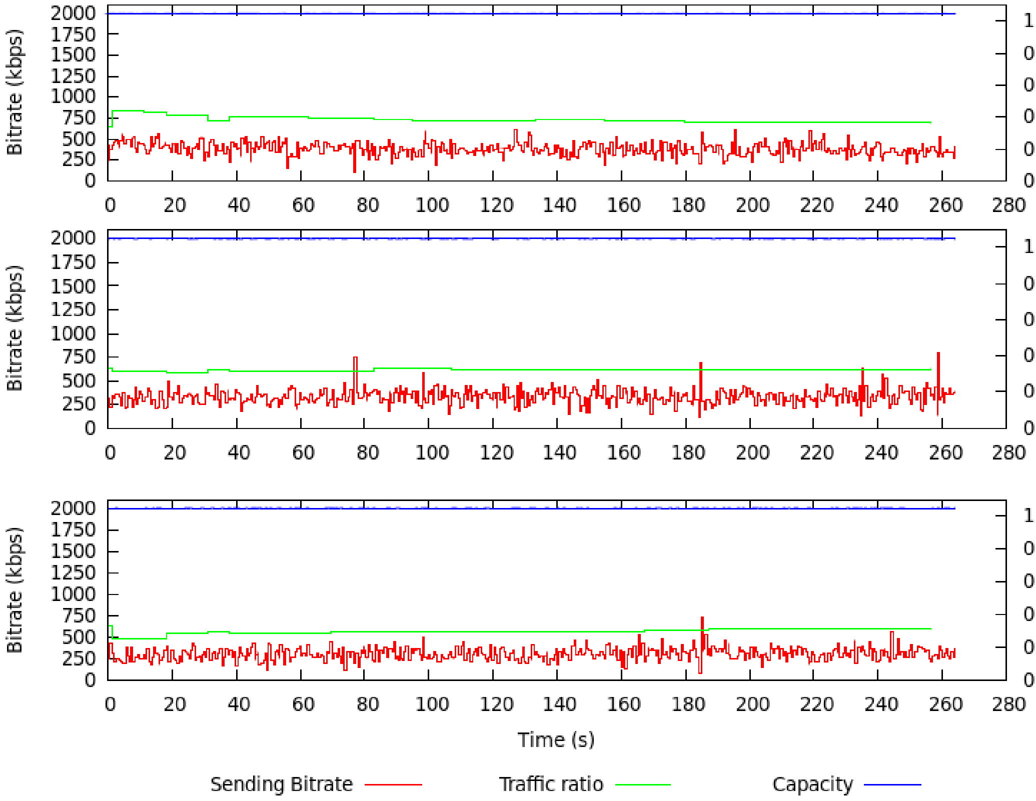


Figure 5.3-3 RAMP-UP behaviour when paths have different loss rates

5.3.4 Paths with different bandwidths

RAMP-UP is specifically designed to withstand capacity changes in the available links by shifting traffic off congested paths. In this experiment, we have two available paths; path1 with fixed 1Mbps link capacity and path2 with a varying capacity. Both paths have fixed network delay of 50ms and no losses. Neither of the two paths have enough capacity to carry the stream independently. The PSNR value is shown in Table 5.3-6 and it can be seen that percentage loss is kept below 0.8%.

Table 5.3-6 PSNR and loss rate when paths have different bandwidths

Scenario	PSNR		Percentage packet loss
Changing capacity on 1 path	Average	StdDev	
2 paths using RAMP-UP	42.9309	2.2293	0.7722

Figure 5.3-4 shows the traffic distribution. It takes RAMP-UP approximately 3 seconds to detect the losses and lower the percentage if the link goes into congestion. The algorithm would probe the link to see if the congestion has cleared, and balance the load, however, this probing only takes place at longer intervals to avoid traffic load oscillations between the paths.

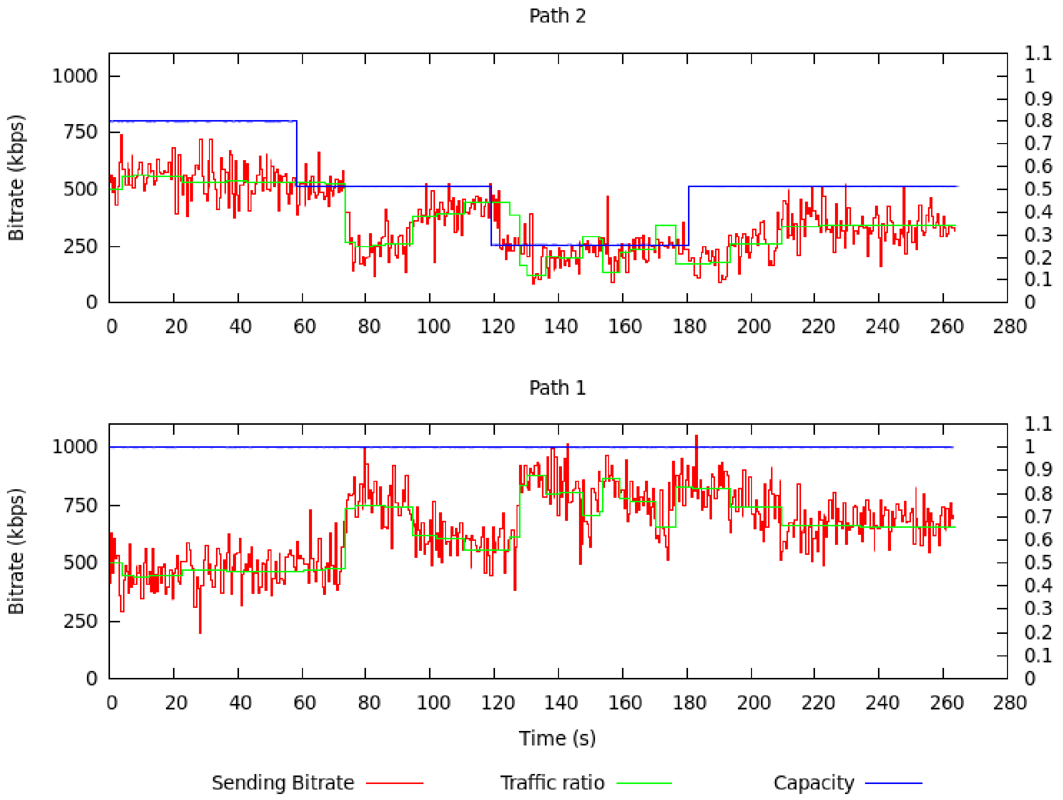


Figure 5.3-4 RAMP-UP behaviour when paths have different bandwidths

5.3.5 Competing RAMP-UP senders

In this experiment, we wanted to observe how RAMP-UP scheduler would behave while competing with another RAMP-UP sender for the same resources. There are three available paths, path1 with 800kbps capacity, path2 and path3 with 1Mbps capacity. The delay on all paths is set to 50ms and there are no losses. Two senders A and B are simultaneously streaming video to two different receivers. Sender A uses path1 and path2, while sender B uses path1 and path3. Hence path1 is being shared by the two senders. The results are given in Table 5.3-7. Both senders act fairly towards each other resulting in a comparable PSNR and percentage loss for each flow. Figure 5.3-5 shows the sending rate of the two senders.

Table 5.3-7 PSNR values for competing RAMP-UP senders

Scenario	PSNR		Percentage
Competing RAMP-Ups	Average	StdDev	packet loss
Sender A	44.41	0.03	0.13
Sender B	44.50	0.23	0.11

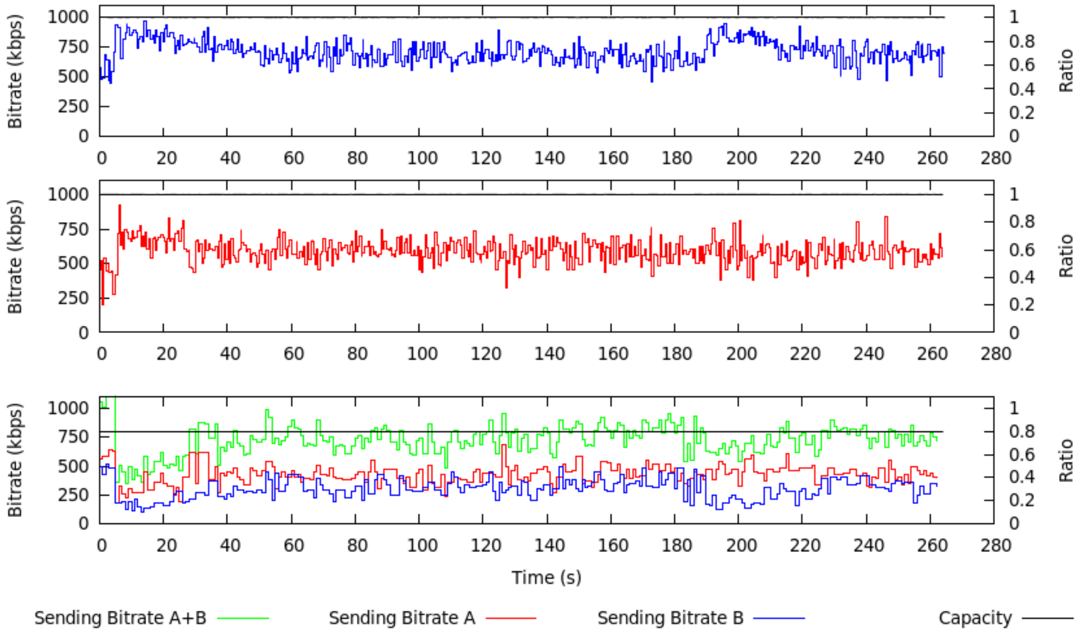


Figure 5.3-5 RAMP-UP senders when competing for common resources

5.3.6 Fixed and wireless paths

Internet users often have a fixed Internet connection along with one or more wireless connections. Hence, to create a more practical scenario, we tested MPRTP over one fixed Internet path (Path 2) and a wireless 3G path (Path 1). We used a 1Mbps fixed capacity for the Internet path, and simulated 3G using the 300s RLC trace provided in

[24] with 0.5-1.0% bit error losses. The link capacity was changed at 10s slow intervals and 1s quick intervals for performance comparison, while the delay was kept constant.

Table 5.3-8 PSNR values for fixed and wireless paths

Scenario	PSNR		Percentage packet loss
Internet and 3G path	Average	StdDev	
3G link capacity changes at 1s	46.7173	0.2084	0.3296
3G link capacity changes at 10s	42.4825	0.5506	0.8531

Figure 5.3-6 shows the results of the slow capacity changes. The algorithm assigns more traffic to the Internet link whenever it experiences congestion on the 3G link. The intervals 110-140 sec and 180-210 seconds demonstrate the scheduler probing the 3G link for more capacity in order to balance the load. The lossy nature of the 3G link also inhibits the scheduler from putting too much load on it.

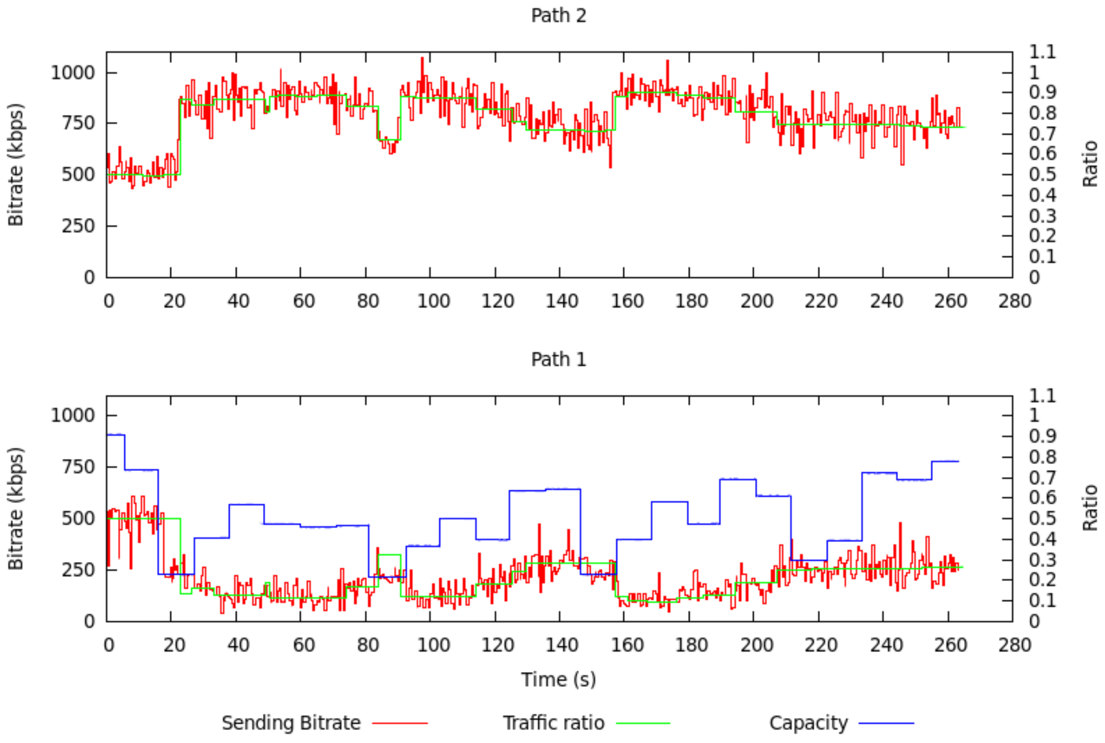


Figure 5.3-6 Path1 is 3G, path2 is Internet. 3G link capacity changes at 10s

Figure 5.3-7 shows the results of the quick capacity changes. In this case the 3G link has a higher bitrate on average resulting in a more uniform load distribution. For about 170 seconds, we see that the assigned ratios are not affected by the changes in capacity of the 3G path. This is because the scheduler doesn't know the actual bitrate of the path. Despite of the changes in the available bitrate, since the capacity of path 1 is sufficient to carry the traffic that is assigned to it, the scheduler keeps sending the same ratio of traffic on it.

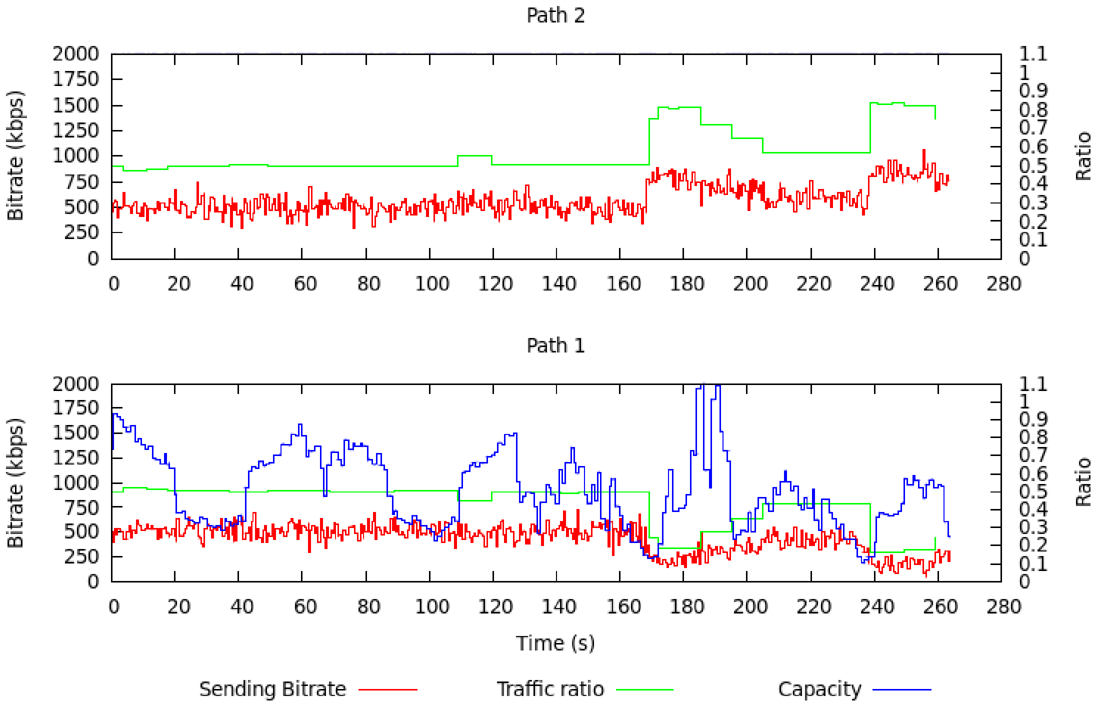


Figure 5.3-7 Path1 is 3G, path2 is Internet. 3G link capacity changes at 1s

5.3.7 Multiple 3G paths

In our final set of experiments, we evaluate the performance of our algorithm when using multiple 3G paths in an outdoor environment. The 3G path is simulated as in the previous case, with slow and quick capacity changes, 0-1% losses and similar delay values. The combined capacity of the two paths is always kept enough to carry the stream, which is approximately 1Mbps. Table 5.3-9 shows the PSNR results.

Table 5.3-9 PSNR values for two 3G paths

Scenario	PSNR		Percentage packet loss
Using two 3G paths	Average	StdDev	
Link capacity changes at 1s	46.1704	0.1751	0.9505
Link capacity changes at 10s	39.2680	1.8932	1.4074

Figure 5.3-8 and 5.3-9 show the performance for slow and quick capacity changes, respectively. The load shifting occurs only when capacity of any path becomes drops so low that it is unable to carry the percentage of traffic assigned to it. This behavior becomes more apparent after 180s, before which the load is almost uniformly distributed between the two paths.

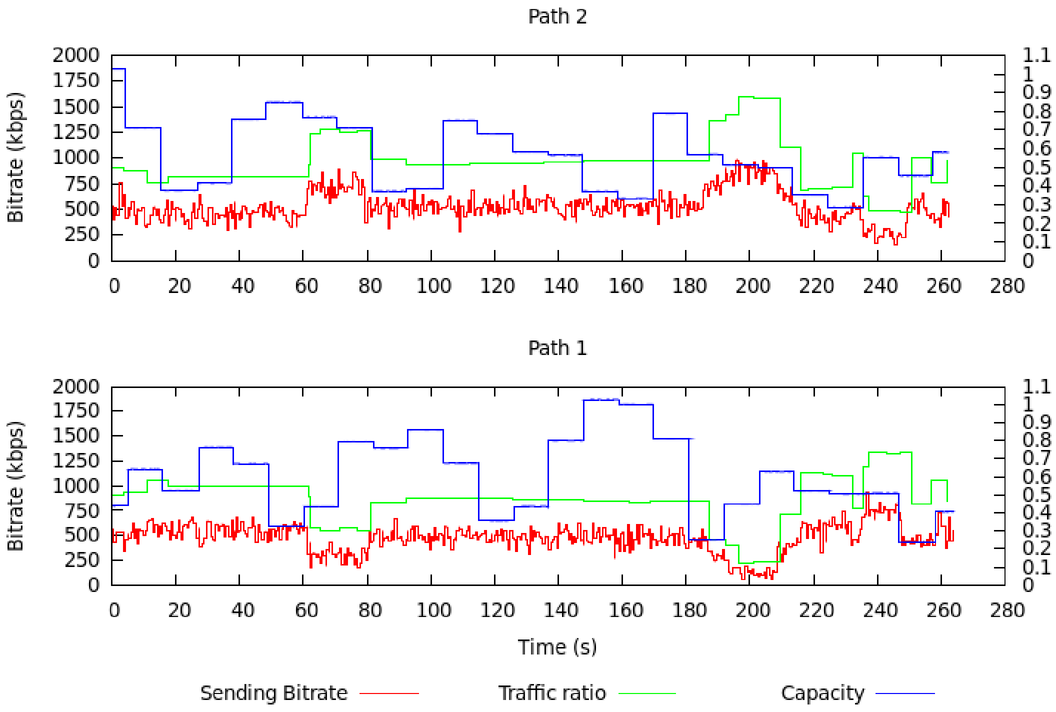


Figure 5.3-8 MPRTP over two 3G paths with link capacity changes at 10s

Figure 5.3-9 shows another aspect of the scheduling algorithm. At about 185 seconds, path 1 begins to experience losses due to a drop in the available capacity and the algorithm starts using the CB value for calculating percentage distribution. At about 210 seconds, path 2 goes into congestion as well. The percentages are continuously being updated after this point till about 250 seconds. The ratio assigned at 250 seconds is almost equal for both paths and similar to what it was before either of the paths had gone into congestion. This is because the algorithm has not seen losses on either path and has cleared the CI for both paths. The ratios are once again being calculated on the basis of TB. The ratio for path 2 is 0.54, which is slightly higher than that of path 1. The reason for this difference becomes clear if we see the sending bitrate of the paths between 190 and 210 seconds. The sending bitrate assigned to path 2 during this interval is higher than the bitrate observed on path 1 throughout the length of the experiments. This results in a higher value of TB for path 2 in comparison to path 1.

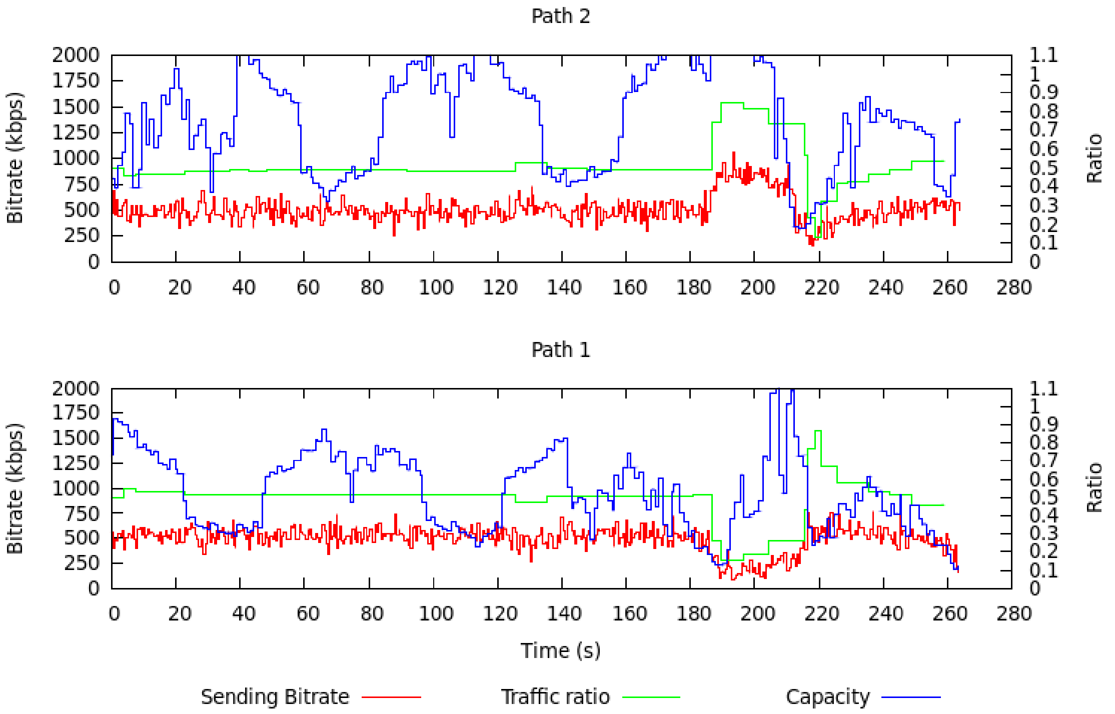


Figure 5.3-9 MPRTP over two 3G paths with link capacity changes at 1s

5.3.8 Backward compatibility

As a sanity test to ensure that our MPRTP application could perform with legacy RTP applications, we did a series of simple tests with gstreamer. We first used an MPRTP receiver to receive a stream from a gstreamer RTP sender and stored it to a file. The stream was successfully played back after being received. We then used a MPRTP sender that sent a video stream over a single path to a gstreamer RTP receiver. In this case as well, the stream was successfully played back. Finally, to test with multiple paths, we programmed a MPRTP stream multiplexer. The function of the multiplexer was to simply receive the incoming stream from multiple paths and transmit it over one single path. The final test setup is shown in Figure 5.3-10.

In this scenario, we observed that when the out-of-order delivery and jitter values were within gstreamer's buffering range, no losses were observed. However, when the path latencies and/or bandwidths mismatched significantly, we observed that three gstreamer properties needed to be adjusted to avoid losses. Firstly, for gstreamer to detect a stream and initialize the jitter buffer, it requires in order packets in the beginning, and hence using multiple paths can lead to initial frame losses due to misorder. Secondly, a packet should not be so late that gstreamer thinks that the stream was reset, and new sequence numbers have started. Thirdly, the playout delay should be sufficient to avoid losses due to late arrival. All these parameters are functions of the receiver side buffer.

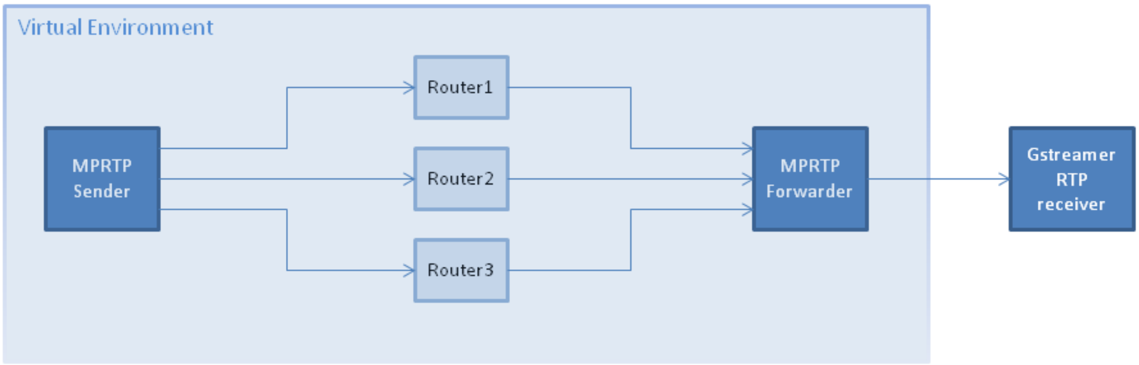


Figure 5.3-10 MPRTP setup for backward compatibility test

5.4 Summary

We conducted a series of experiments to evaluate the performance of our own MPRTP algorithm, RAMP-UP, and also to better understand the usability and feasibility of MPRTP in server to client video streaming scenarios.

Our first set of experiments focused on studying behavior over multiple paths in reference with specific path characteristics. We observed that when sufficient capacity is available, performance of MPRTP streaming over multiple paths is comparable to that of single path streaming. The algorithm is not able to completely avoid lossy paths due to design constraints, however by spreading the stream over multiple paths, the percentage loss is decreased. Furthermore, by using playout buffering we can effectively eliminate packet drops due to out-of-order delivery when the paths taken have different network delays.

The second set of experiments consisted of scenarios when a single path does not have enough capacity/bitrate to carry the stream, and the load must be split over multiple paths. The algorithm was able to avoid congestion by shifting load to other paths. When competing for resources, it delivers some level of fairness by spreading the load over multiple paths. Furthermore, our simulations show that the algorithm gives reasonable performance when using a combination of wired and wireless paths. In 3G scenarios where path bitrates change over time, the algorithm is able to ramp up or down the amount of traffic on the path when the bitrate increases or decreases, respectively.

Finally, we performed some tests where we used RAMP-UP with gstreamer RTP pipelines and found that the two could work together smoothly, provided we adjust the playout parameters of gstreamer to compensate for packet misorder and jitter.

Chapter 6
Conclusion

We have presented in this work an extension for RTP that is capable of simultaneously utilizing multiple paths between endpoints. The extension, MPRTP, is backward-compatible with RTP applications. It is implemented on application-level and does not require any kernel-level modifications for deployment. It runs smoothly over UDP, however, it can run over other transport protocols as well. We also presented our design and implementation of a sample scheduler called RAMP-UP and a MPRTP receiver for video streaming scenarios.

6.1 Multipath vs. Single path

Our experiments with RAMP-UP confirmed that MPRTP is capable of combining the capacity of multiple paths and increase the available bandwidth, allowing senders to stream high quality videos. Furthermore, splitting the stream across multiple paths spreads the load, and prevents overloading any one path, and reduces overall percentage losses when some of the paths are lossy. The effects of jitter and out of order delivery can be minimized for the stream by using sufficient receiver side buffering, and introducing playout delay. We observed that RAMP-UP performance over multiple paths was comparable to similar single path cases. Hence, it is capable of delivering similar quality while exploiting additional resources.

For cases where paths had diverse bandwidth values, our application is able to spread the load across the paths so that none of the paths experiences congestion; given the combined bandwidth of the paths is sufficient to carry the stream. It is also able to shift load to other paths if it experiences congestion on a path due to competition with another application or any other network conditions.

6.2 Implementation Challenges and Backward Compatibility

As discussed before, the MPRTP implementation does not require kernel-level modifications. This ensures quick and easy deployment. MPRTP can also be introduced to existing RTP applications, however since it operates between RTP and the transport layer, the modifications require adding an interface that communicates with both these layers. MPRTP senders that rely on receiver side buffering may induce higher levels of out-of-order delivery and delayed packets that legacy RTP receivers may not be tolerant to. Therefore, when upgrading legacy applications to MPRTP, we would need to modify the buffering parameters as well.

Our experiments with gstreamer confirmed that MPRTP is backward compatible with legacy RTP applications. The additional MPRTP header is ignored as an unrecognized extension.

6.3 Future Work

The work in the thesis is conclusive in providing experimental evidence of the advantages of MPRTP. However, the focus of our work has been video streaming in unicast scenarios and covers only one possible use case of MPRTP. Our experiments show positive results but we need a more detailed evaluation using complex network conditions, wider range of bitrates and diverse video encodings. Our algorithm only focuses on bandwidth aggregation and congestion avoidance, whereas redundancy, fallback and avoiding lossy paths can greatly add to the advantages of MPRTP in media streaming. These aspects also need to be investigated.

Further research is required to carry MPRTP towards maturity as a protocol. The possibility of integrating our research on MPRTP with rate adaptation (e.g. [25]) needs to be explored as part of future work. Also, research done in the area of layering and redundancy for RTP [26] should also be reviewed in the context of availability of secondary paths in case of MPRTP.

References

[1] E. Nordmark and M. Bagnulo , *Shim6: Level 3 Multihoming Shim Protocol for IPv6*, Request For Comments 5533, Network Working Group: Retrieved June 2011 http://tools.ietf.org/html/rfc5533

[2] R. Moskowitz, P. Nikander, P. Jokela and T. Henderson, *Host Identity Protocol,* Request For Comments 5201, Network Working Group: Retrieved July 2010 http://www.ietf.org/rfc/rfc5201.txt

[3] P. Nikander, T. Henderson, C. Vogt and J. Arkko, *End-Host mobility and multihoming with the Host Identity Protocol,* Request For Comments 5206, Network Working Group: Retrieved July 2010 http://tools.ietf.org/search/rfc5206

[4] R. Stewart, Ed., *Stream Control Transmission Protocol,* Request For Comments 4960, Network Working Group: Retrieved July 2010 http://www.rfc-editor.org/rfc/rfc4960.txt

[5] M. Molteni and M. Villari, *Using SCTP with Partial Reliability for MPEG-4 Multimedia Streaming,* Proc. of BSDCon Europe, 2002

[6] Hyelim Park, Myungchul Kim et. al., *A mobility management scheme using SCTP-SIP for realtime services across heterogeneous networks*, ACM Symposium on Applied Computing, 2009

[7] R. Stewart, Ed., *Architectural Guidelines for Multipath TCP Development* Internet Draft,Engineering Task Force: Retrieved July 2010 http://tools.ietf.org/html/draft-ietfmptcp-architecture-01

[8] H.Schulzrinne, S.Casner, R.Frederick and V. Jacobson, *RTP Profile for Audio and Video Conferences with Minimal Control,* Request For Comments 3551, Network Working Group: Retrieved July 2011 < http://www.ietf.org/rfc/rfc3551.txt>

[9] H.Schulzrinne and S.Casner, *RTP: A transport protocol for realtime applications,* Request For Comments 3550, Network Working Group: Retrieved July 2010 http://www.ietf.org/rfc/rfc3550.txt

[10] Yi J. Liang, Eckehard G. Steinbach, and Bernd Girod, *Multi-stream voice over IP using packet path diversity*,in IEEE Fourth Workshop on Multimedia Signal Processing

[11] J. G. Apostolopoulos, *Reliable video communication over lossy packet networks using multiple state encoding and path diversity*, in Proceedings Visual Communication and Image Processing, Jan. 2001.

[12] M. Fiore and C. Casetti, *An Adaptive Transport Protocol for Balanced Multihoming of Real-Time Traffic*, IEEE Globecom 2005

[13] H. Schulzrinne, A. Rao and R. Lanphier, *Real Time Streaming Protocol (RTSP)*, Request For Comments 2326, Network Working Group: Retrieved June 2011 http://www.ietf.org/rfc/rfc2326.txt

[14] M. Handley, H. Schulzrinne et. al. , *SIP: Session Initiation Protocol*, Request For Comments 3261, Network Working Group: Retrieved June 2011 http://www.ietf.org/rfc/rfc3261.txt

[15] R. Fielding et. al. , *Hypertext Transfer Protocol -- HTTP/1.1*, Request For Comments 2616, Network Working Group: Retrieved June 2011 http://www.ietf.org/rfc/rfc2616.txt

[16] Colin Perkins (2003), *RTP audio and video for the Internet,* Pearson Education

[17] James Harry Greene, Jeane Fleming (2002), *Voice and video over IP,* McGraw-Hill Professional Publishing

[18] Trilogy: Architecting the future Internet, Retrieved June 2011 http://trilogy-project.org/home.html

[19] RTP Tools (Version 1.18), Retrieved June 2011 http://www.cs.columbia.edu/irt/software/rtptools/

[20] J. Rosenberg, *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*, Request For Comments 5245, Internet Engineering Task Force (IETF): Retrieved June 2011 http://tools.ietf.org/html/rfc5245

[21] Dominique Fober, Yann Orlarey, Stephane Letz, *Real time clock skew estimation over network delays,* Grame, June 2005

[22] GStreamer Good Plugins 0.10 Plugins Reference Manual: GTK-Doc V1.14: Retrieved June 2010 http://www.gstreamer.net/data/doc/gstreamer/head/gst-plugins-good-plugins/html/gstplugins-good-plugins-gstrtpjitterbuffer.html

[23] Nokia, “Homepage of H.264 codec.”, http://research.nokia.com/page/4988

[24] 3GPP R1-081955, “LTE Link Level Throughput Data for SA4 Evaluation Framework.” May 2008.

[25] V. Singh, J. Ott, and I. Curcio, *Rate adaptation for conversational 3G video*, INFOCOM Workshop on Mobile Video Delivery, Rio de Janeiro, Brazil, 2009.

[26] RTP: Redundancy and Layering, Retrieved June 2011, http://www.cs.columbia.edu/~hgs/rtp/redundancy.html

[27] Netem, Retrieved November 2011, http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[28] Xiph.org Test Media, Retrieved July 2011, http://media.xiph.org/video/derf/