Varun Ranjit Singh

Rate-control for Conversational H.264 Video Communication in Heterogeneous Networks

Faculty of Electronics, Communications and Automation

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

Espoo 20.05.2010

Thesis supervisor:

Prof. Jörg Ott

Thesis instructor:

M.Sc.(Tech.) Igor D.D. Curcio

Aalto University
School of Science
and Technology

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF THE
MASTER'S THESIS

Author: Varun Ranjit Singh

Title: Rate-control for Conversational H.264 Video Communication in Heterogeneous Networks

Date: 20.05.2010 Language: English Number of pages: 13+83

Faculty of Electronics, Communications and Automation

Department of Communication and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Prof. Jörg Ott

Instructor: M.Sc.(Tech.) Igor D.D. Curcio

The transmission bit rate available along a communication path in a heterogeneous networks is highly variable. The wireless link quality may vary due to interference and fading phenomena and, peered with radio layer reconfiguration and link layer protection mechanisms, lead to varying error rates, latencies, and, most importantly, changes in the available bit rate. And in both fixed and wireless networks, varying amounts of cross traffic from other nodes (i.e., the total offered load on the individual links of a network path) may lead to fluctuations in queue size (reflected again in a path latency) and to congestion (reflected in packet drops from router queues). Senders have to adapt dynamically to these network conditions and adjust their sending rate and possibly other transmission parameters (such as encoding or redundancy) to match the available bit rate while maximizing the media quality perceived at the receiver.

We investigate congestion indicators and their characteristics in different multimedia environments. Taking these characteristics into account, we propose a rate-adaptation algorithm that works in the following environments: a) Mobile-Mobile, b) Internet-Internet and c) Heterogeneous, Mobile-Internet scenarios. Using metrics such as Peak Signal-to-Noise Ratio (PSNR), loss rate, bandwidth utilization and fairness, we compare the algorithm with other rate-control algorithms for conversational video communication.

Keywords: Rate-control, Conversational Video Communication, H.264, RTP/RTCP, 3G, Internet, Heterogeneous networks, congestion control

"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change."

Charles R. Darwin (1809-1882)

Acknowledgements

My sincere thanks to all the people who helped me with the thesis. First, Prof. Jörg Ott for inviting me to join his research team and providing the opportunity to pursue this research topic, for motivating, his open doors and encouragement towards new ideas. He's been an outstanding guide and a fountain of knowledge. An equal measure of gratitude is due to Igor Curcio of Nokia Research Center for his guidance and pointers during the course of the project, "MObile VIdeo Enhancements" (MoViE), he was instrumental in providing direction and suggesting alternatives.

Second, the Department of Communication and Networking (ComNet) at Aalto University (earlier known as Helsinki University of Technology, Espoo, Finland) for providing the platform to pursue research. Nokia Research Center (Tampere, Finland) for introducing the initial problem statement and TEKES for financially supporting the project (2008-2009). Third, my former colleague Jegadish, for the basic 3G simulator framework. Fourth, work colleagues, without naming them all, for their suggestions and timely feedback. Special thanks to Arja, Sanna and Sarri, Jenni for helping with the various administrative tasks.

Fifth, I would like to thank my flat mates, Javier, Mário, László, for the long hours spent discussing, brainstorming, and for motivating when the chips were down. My friends in Helsinki: Pavan, Cathy, Sathya, Niko, Sergio, Davide, Kuba, Kalle, Tan, Judy, and Amit; who have inspired and supported me through this journey. And this journey wouldn't be complete without thanking my friends at Aalto Entrepreneurship Society (Markus, Tuomo, Kristo, Jori, Andy, Krista, Linda, Lotta and others) for the amazing last year (2009-2011) and for almost waylaying my best laid plans to graduate, for their thought provoking lectures, opportunity to network with other entrepreneurs, and infinite zest.

Sixth, Cathy and Sathya for proof-reading the final drafts of the thesis, it expedited the process.

Seventh, Irena for letting me work through the long hours, for supporting my whims and fancies and for listening in.

Last but not least, I would like to thank my family for the sacrifices they have made and that the success of this thesis wouldn't have been possible with out their love.

Otaniemi, 25.11.2009

Varun Ranjit Singh

Contents

List of Figures

List of Tables

List of Algorithms

Definition and abbreviations

Definitions

Delay Budget ($delay_{max}$)	is the acceptable delay experienced by a packet from the time it is generated to the time it is decoded. It is also known as application-defined maximum delay, $\mathbf{Time\_render(Pkt_i) - Time\_gen(Pkt_i) \leq Delay\ Budget}$
Discarded Packets	are the amount of packets dropped by the application in an RTCP Interval that did not arrive within the delay budget
Flow	In a packet switched network, flow is a sequence of packets sent from a source to a destination
HSN	is the highest sequence number RTP packet received at the receiver
kilobits	1000 bits
kilobytes	1000 bytes
LPS	is the sequence number of the last packet transmitted by the sender just before receiving the RTCP RR
now	represents the current time, or the status of the associated variable at the current time
NSN	is the next sequence number to be decoded in receiver queue
One-Way Delay	is the difference between the generation time of a packet at the sender and reception time of the packet at the receiver
PDP-context	is a data structure present at both the Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN). The data structure contains the subscriber's IP address, International Mobile Subscriber Identity (IMSI) and Tunneling end point Identity of the SGSN and GGSN
Playout Delay	is the difference between the scheduled playout time of the NSN packet and the time the receiver sends the RTCP Report
RTCP Transmission Interval	is the interval between two successive RTCP packets. The interval calculation is defined in RTP [RFC3550]. Typically, for P2P systems, it is a minimum of 5 ± 2.5 *seconds*. However, for video communication, quicker feedback may be needed as specified in RTP/AVPF [RFC4585].
Round-Trip Time (RTT)	Elapsed time between time of sending of a packet and receiving the corresponding acknowledgment. In case of packets sent by best-effort, it is the time between generating and sending the Sender Report and receiving the Receiver Report. $\mathbf{RTT = Time\_recv(RR_{last}) - DSLR(RR_{last}) - Time\_gen(SR_{last})}$
Time Since Receiving Last Packet	is the difference between the reception time of the HSN packet (last RTP packet) and the time the receiver sends the RTCP Report , $\mathbf{\Delta T_{HSN} = Time_{now} - Time\_recv(Pkt_{HSN})}$

Notation

BR_R	Interval bit rate at receiver
BR_S	Interval bit rate at sender
RR_{est}	Estimated receiver bit rate at the sender
$delay_{max}$	application defined maximum delay or delay budget
$delay_n$	Time taken to render an RTP packet from the time it is generated
$Interval_{RR}$	The RTCP RR reporting interval at the receiver
$Interval_{SR}$	The RTCP SR reporting interval at the sender
seq_no	is the sequence number of an RTP packet, $n \quad \forall \quad n \in${0 ... NSN, HSN, LPS }
$sizeof(n)_{bytes}$	is the packet size of the n^{th} RTP packet in bytes
$Time_t$	current time (now) or a time "t" at which an event takes place, $\forall t \in${0, start session ... NTP TS ... last_SR, last_RR, now}
$TS_R[n]$	Receiving Timestamp for n^{th} RTP packet
$TS_S[n]$	Timestamp for generating n^{th} RTP packet

Abbreviations

3G	3^{rd} Generation
3GPP	3^{rd} Generation Partnership Project
AAAA	Access Anything, Anywhere, at Anytime
ABU	Average Bandwidth Utilization
ACK	ACKnowledgement
ADSL	Asynchronous DSL
ADU	Application Data Unit
AIMD	Additive Increase Multiplicative Decrease
API	Application programming interface
ARQ	Automatic ReQuest
AVC	Audio Video Coding
AVPF	Audio Video Profile with Feedback
BDR	Bytes Discarded Report
BFL	Buffer Fill-Level
BR	Bit Rate
C-NADU	Conversational NADU
CCIR	Consultative Committee for International Radio
CCM	Codec Control Messages
CIF	Common Intermediate Format
CODEC	Compressor-Decompressor or Coder-Decoder
DCCP	Datagram Congestion Control Protocol
DL	Downlink
DLR	Delta Loss Rate
DSL	Digital Subscriber Line
DSLR	Delay Since Last sender Report
DVR	Digital Video Recorder
DRLE	Discard RLE
ECN	Explicit Congestion Notification
EDGE	Enhanced Data Rates for GSM Evolution

e2e	end-to-end
FB	Feedback
FBS	Free Buffer Space
FEC	Forward Error Correction
FPS	Frames per Second
FPQ	First Packet in the Queue
FTP	File Transfer Protocol
GoP	Group of Pictures
GP	Goodput
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HDTV	High Definition Television
HSDPA	High Speed Downlink Packet Access
HSN	Highest Sequence Number
HSPA	High Speed Packet Access
HSUPA	High Speed Uplink Packet Access
HTTP	Hyper-Text Transfer Protocol
IAT	Inter-Arrival Time
IETF	Internet Engineering Task Force
IP	Internet Protocol
IM	Instant Messaging
IMT-2000	International Mobile Telecommunications-2000, also known as, $3G$
IMS	IP Multimedia Subsystem
ISP	Internet Service Provider
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector
KB	Kilo Bytes
LAN	Local Access Network
LPS	Last Packet Sent (Sequence no. of the last packet sent)
LTE	Long-Term Evolution
MAC	Media Access Control
MB	Mega Bytes
MSE	Mean Square Error
MTSI	Media Telephony Service for IMS
MTU	Maximum Transmission Unit
NADU	Next Application Data Unit
NACK	Negative ACKnowledgement
NSN	Next Sequence Number
NTP	Network Time Protocol
NTSC	National Television System Committee
OWD	One-Way Delay
P2P	Peer-to-Peer
PAL	Phase Alternate Line
PD	Playout Delay
PDP	Packet Data Protocol
PiB	Packets in a Buffer
PiT	Packets in Transit
PSNR	Peak Signal-to-Noise Ratio

PSS	Packet-switched Streaming Service
PSTN	Public switched telephone network
PT	Payload Type
PtP	Point-to-Point
QCIF	Quarter Common Intermediate Format
QoS	Quality of Service
QVGA	Quarter Video Graphics Array
RAN	Radio Access Network
RLC	Radio Link Control
RLE	Run Length Encoding
ROHC	RObust Header Compression
RR	(RTCP) Receiver Report
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
RTT	Round-Trip Time
SACK	Selective ACKnowledgement
SDP	Session Description Protocol
SDTV	Standard Definition TV
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SR	(RTCP) Sender Report
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
TMMBN	Temporary Maximum Media Stream Bit Rate Notification
TMMBR	Temporary Maximum Media Stream Bit Rate Request
TS	Timestamp
UDP	User Datagram Protocol
UL	Uplink
UMTS	Universal Monile Telecommunication System
VGA	Video Graphics Array
VLC	Variable Length Coding
VoIP	Voice over IP
VVoIP	Voice and Video over IP
WLAN	Wireless LAN
XR	eXtension Report

1 Introduction

Over the past 15 years, the Internet has been changing the way we consume and share information. Traditional media[1] have been supplemented by the Internet to provide quick access to information. Due to the steady increase in network capacity over the last decade, many multimedia services have emerged. comScore [1] recently reported that in the month of November 2009, about 31 billion videos were watched online by more than 170 million U.S. Internet viewers [2]. Therefore, video streaming websites like YouTube [3], Hulu [4], etc, have enhanced the way multimedia content is generated and consumed. The increased connectivity has led to a paradigm shift in consumption of multimedia services from a simple pre-scheduled, broadcast-type model to Access Anything, Anywhere at Anytime (AAAA). Moreover, due to the emergence of mobile devices, both the user and session mobility have become key factors for media consumption. For example, Digital Video Recorder (DVR) products like Slingbox [5], TiVo [6] and Hava Player [7] not only allow remote access for recording televised content, but also allow streaming the content to the user's mobile device(s).

With the emergence of Voice over Internet Protocol (VoIP) applications on the Internet, users are able to remain in contact with people around the globe at reduced voice call costs [8]. Furthermore, VoIP reduces operating cost and increases flexibility by avoiding the need for separate infrastructure for voice and data networks. In recent years, VoIP calls not only connect Internet users but also connect Internet to Public Switching Telephone Networks (PSTN) [8]. Application software like SIP-based Phones [9], Skype [10] and Gizmo Project [11] already supports Video and Voice over IP (VVoIP) communication, but at the moment calls are limited to peers on the Internet. Similar to video streaming, trends indicate that in the near future, VVoIP calls will also originate on the Internet and terminate on mobile devices or vice versa.

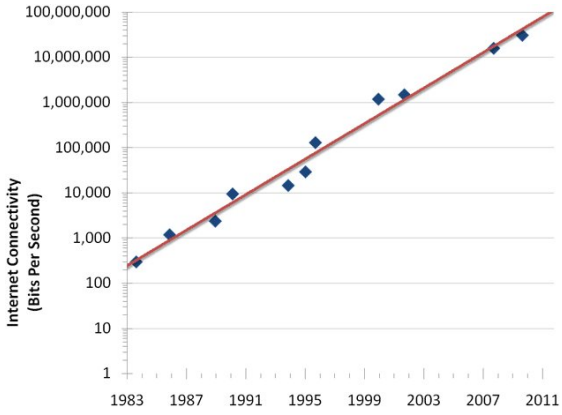


Figure 1: Channel Capacity evolution in bits/second since 1993. The straight line shows predicted while the dots depict real data rates.

Note: Y-axis is a logarithmic scale.

Source: Wikipedia, "Nielsen's Law of Internet Bandwidth"

Nielsen's Law of Internet bandwidth [12] states that a high-end user's connection speed grows at 50% per year or doubles every 21 months [13]. Figure 1 shows the evolution of channel capacity over the years. Similarly, Moore's Law predicts that computing power

[1] E.g. Newspaper, Television, Radio, and Magazines etc.

grows 60% every year or doubles every 18 months [14]. Therefore, the rate of growth of computing capacity exceeds the rate of growth of network bandwidth. Correlating the two laws, Nielsen also observed that for a good user experience, bandwidth would be the bottleneck and not processing power [12]. However, applications that were considered challenging a few years ago due to limited network capacity are nowadays possible.

The rest of the chapter is structured as follows: Sub-sections 1.1-1.3 describe the advancements in connectivity, the different types of multimedia formats, and challenges in delivering multimedia content over a network. Sub-sections 1.4-1.6 describe the problem that this thesis tackles, the contribution of the author, and the scope and goals of the thesis. Lastly, Sub-section 1.7 lays out the structure of the thesis.

1.1 Connectivity

Initially, we discuss about mobile users. The Universal Mobile Telecommunications System (UMTS) or the third generation (3G) networks typically provide a user with 256 kbps to 2 Mbps data connection. The ITU-T IMT-2000 specifies that a minimum bit rate of 2 Mbps is provided for stationary or walking users, and 348 Kbit/s for a user in a moving vehicle [15]. However, using High Speed Packet Access (HSPA), 3G networks can provide up to 14 Mbps in the downlink and 5.8 Mbps in the uplink. 3G specifications are continuously evolving and data rates are expected to exceed 100 Mbps with Long Term Evolution (LTE).

Alternatively, fixed line subscribers such as home and corporate users access the Internet via broadband technologies such as Asynchronous Digital Subscriber Line (ADSL), Cable Internet access, optic fibers or even satellite links. ADSL is widely popular in residential and home-office environments. It provides mean data rates of about 6.144 Mbps on the downstream and 0.64 Mbps on the upstream link [16].

1.2 Multimedia contents

Video quality often depends on the resolution of the display, Table 1 lists the various video formats, their dimensions and applications; it also lists the uncompressed volume of video data generated in 1 *second* (sampled at 15 and 30 frames per second (fps)). The first 3 rows of the table describe resolutions commonly used for mobile video streaming and communication.

For example, QVGA generates 2.3 MB of uncompressed YUV 4:2:2 data, i.e., $18.4 Mbps$ for 30 fps video or $9.2 Mbps$ for 15 fps video. As noted in the previous section, peak rates of an ADSL or 3G connection will not be sufficient to transmit even the 15 fps raw video feed. Therefore, the raw YUV data is compressed using modern codecs for transmitting in a network. For achieving high compression, most modern codecs employ motion compensation between video frames to reduce temporal redundancy, followed by spatial transformation or compression to reduce spatial redundancy within a frame. Additionally, the encoder may alter the frame rate, or the quantization factor (used in frame compression) to reduce the size of the frame but at the cost of deteriorating video quality. Variable Length Coding (VLC), a type of entropy coding, helps in further compressing the bitstream.

Table 1: Video Formats, their Dimensions, and typical applications

Format	Dimensions (w x h)	Volume of uncompressed video data generated (YUV422 [17]) every second and sampled at 15 FPS	Volume of uncompressed video data generated (YUV422 [17]) every second and sampled at 30 FPS	Applications
Sub-QCIF	128 x 96	0.19 MB	0.37 MB	Handheld mobile video &
QCIF	176 x 144	0.37 MB	0.76 MB	video-conferencing using
QVGA	320 x 240	1.15 MB	2.3 MB	phone/wireless networks
CIF	352 x 288	1.52 MB	3.04 MB	Videotape recorder quality
CCIR 601	720 x 480	5.2 MB	10.40 MB	SDTV/NTSC
4CIF	704 x 576	6.09 MB	12.17 MB	SDTV/PAL
HDTV 720	1280 x 720	13.83 MB	27.65 MB	DVD sources
HDTV 1440	1440 x 960	23.5 MB	47.00 MB	Consumer HDTV
HDTV 1080	1920 x 1080	31.35 MB	62.70 MB	Studio HDTV

Source: Wiley, "Video Compression and Communications: From Basics to H.261, H.263, H.264, MPEG4 for DVB and HSDPA-Style Adaptive Turbo-Transceivers"

1.3 Network challenges

In wired Internet access, the video telephony application shares bandwidth with other applications such as BitTorrent [18], Web browsers [19], Instant Messaging (IM) applications [20], etc. These applications compete for capacity with one-another and can therefore cause congestion or delays in packet transmission. Similarly, sharing the Internet access with other users in the local area via hubs, switches, Ethernet, Wireless LAN (WLAN), can cause congestion - users compete for the same capacity. Therefore, a video telephony application competes for bandwidth with applications on the same device or in the same network.

Channel capacity in mobile networks fluctuates due to fading, interference, mobility, handovers, cell-load, etc. and can be the cause of bit-error losses. However, similar to wired networks, mobile networks also experience queuing losses[2] in the router. Since high packet losses are detrimental to video quality perception and expensive to repair, they need to be avoided as much as possible.

To overcome these network challenges, Internet protocols usually have a feedback loop. For instance, the Transmission Control Protocol (TCP) has a closed feedback loop, wherein the sender receives an Automatic ReQuest (ARQ) from the receiver. This feedback can be in the form of an ACKnowledgment (ACK) for successfully receiving a packet or a Negative ACKnowledgement (NACK) for a lost packet, or Cumulative or Selective ACKnowledgement (SACK) to aggregate the information for a sequence of packets. The aforementioned ARQ mechanisms also serve as input to various TCP-based congestion control algorithms, such as Additive Increase Multiplicative Decrease (AIMD), TCP Vegas, TCP Reno, etc [21].

[2] queues may overflow

1.4 Problem Statement

Video communication is a delay-sensitive real-time application and it requires the end-points to adapt quickly to the changes in the network capacity (or conditions). The end-points rely on congestion indicators to make adjustments to the audio/video encoding rate so that they do not exceed the end-to-end channel capacity.

Traditional congestion indicators such as packet losses are not applicable because 1) air interface losses and congestion losses may be hard to differentiate and, more importantly, 2) increased queuing delays in the network may cause the receiver to discard packets even before congestion losses appear. Furthermore, 3) each multimedia flow competes with other flows for bandwidth on a shared link. Therefore, a sender has to anticipate upcoming congestion from various cues—including but not limited to the per-packet delay used in many delay-based congestion control algorithms—to prevent network queues from building up in the first place. This requires extreme sensitivity to the reported transmission characteristics.

Multimedia applications thus need to adapt to the bandwidth constraints by adjusting their encoding and/or transmission rate. However, congestion control in heterogeneous networks (containing both wireless and wired paths) for conversational video applications is challenging because the application-defined maximum delay and the minimal network-incurred latency leave only very little room for a congestion control algorithm to operate. Furthermore, differences in the physical nature and the network architecture of 3G mobile services and ADSL-type wired Internet access makes congestion control in heterogeneous environments challenging.

1.5 Contribution of the Thesis

We investigate congestion indicators and their characteristics in different multimedia environments. Taking these characteristics into account we propose a rate-adaptation algorithm that works in the following environments a) Mobile-Mobile, b) Internet-Internet and c) Heterogeneous, Mobile-Internet scenario. Using metrics such as PSNR, loss rate, bandwidth utilization and fairness, we compare the algorithm with other rate-control algorithms for video communication.

1.6 Scope and Goals

The Real-time Transport Protocol (RTP) is used to deliver real-time content and its associated RTP Control Protocol (RTCP) forms the control channel between the sender and receiver. Moreover, RTCP carries important information related to media playout and link conditions. RTCP reports are sent periodically but not often, the time interval is in the order of seconds. Using the existing RTCP extensions standardized in the IETF and 3GPP as a starting point, we propose a new rate adaptation algorithm for video communication.

We make the following assumptions: The video is captured in YUV and compressed using an H.264 [22] codec. Furthermore, the video is compressed for point-to-point (PtP) video communication and we simulate video flow only in the upstream direction and presume that the video on the downstream will be affected in the same way.

Initially, we apply this rate control algorithm to 3G mobile networks and as a next step expand its applicability to fixed or wired Internet. Finally, the rate control algorithm is applied to video communication in heterogeneous networks and compared against existing congestion control algorithms. Meanwhile, we also study the different congestion indicators and compare them with one another for applicability in different environments.

1.7 Structure

This document is organized as follows: Section 2 compares the characteristics of different multimedia environments. Section 3 describes the different congestion indicators and also discusses the available metrics for comparing the different rate control mechanisms. Furthermore, Section 4 introduces basic RTP and RTCP mechanisms and proposes RTCP extensions useful for rate control in video communication. Section 5 describes the simulation environment. Section 6 proposes algorithms for congestion control. Section 7 compares the different rate adaptation techniques (defined in Sec. 4.5) in 3G (Subsection 7.1), Internet (Subsection 7.2), and Heterogeneous environments (Subsection 7.3). Section 8 summarizes the usefulness of different congestion indicators in different environments. It concludes with some observations and possible future directions.

2 Multimedia Communication Environments

This chapter introduces the basic video communication ecosystem and discusses in detail the different components that form the system. Video communication is a duplex channel and Figure 2 describes the upstream channel of video communication. The video camera and the desktop screen represent the video acquisition and rendering part of the system. The video is captured and rendered as raw data, i.e., represented in RGB or YUV form. The encoder and decoder blocks compress and decompress, respectively, the raw video data frames. Compression makes it easier to store or transmit the video data.

The packetizer encodes the compressed data into data packets, fragments large packets into Maximum Transmission Unit (MTU) size packets, and encapsulates the packets in application-specific headers to differentiate between packets. On the other hand, the depacketizer combines fragmented packets, removes the application-specific headers and reconstructs the compressed frame.

The sender buffer schedules the packet for transmission based on the available channel capacity. While in the receiver buffer, the received packets are re-ordered, lost packets requested, and late packets discarded from the queue. The network connects the two peers and the packets traverse the network through several routers - where they may be queued - before the packets reach the receiver.

Conversational video communication differs from both video streaming and broadcast in the following ways:

- Multimedia content flows in both directions, from caller to callee and vice versa, unlike streaming, wherein media content flows from a server to a client (unidirectional).
- Initial buffering time in the case of video streaming can be an order of magnitude longer than in the case of video communication.

2.1 Encoding and Decoding

Video is displayed on the screen as a series of pictures, known as frames. The number of pictures displayed on the screen per second is known as frame rate (measured in frames per second). There are three types of pictures (or frames) used to encode video. Namely, Intra-coded pictures (I-frame), Predicted pictures (P-frames), and Bi-predictive pictures (B-frames). An I-frame is like a static image file with spatial compression [23] and thus independent of the other frames. The frame is also called a key-frame because it is seen as an anchor point for the succeeding frames. The P-frames and B-frames are temporally encoded frames, i.e., they depend on the past or future frames to be decoded. So, P-frames and B-frames store only part of the image information and therefore occupy less space than an I-frame.

P-frames ('Predicted picture') are also known as delta-frames because the P-frame encodes only the changes in the image from the previous frame. For example, in a scene where an object moves across a static background, only the object's movements need to be encoded. A B-frame ('Bi-predictive picture') is even smaller because it encodes the differences between the current frame and the preceding and succeeding frames, thus having higher compression. Figure 3a shows a typical Group of Pictures (GOP). P-frames are predicted

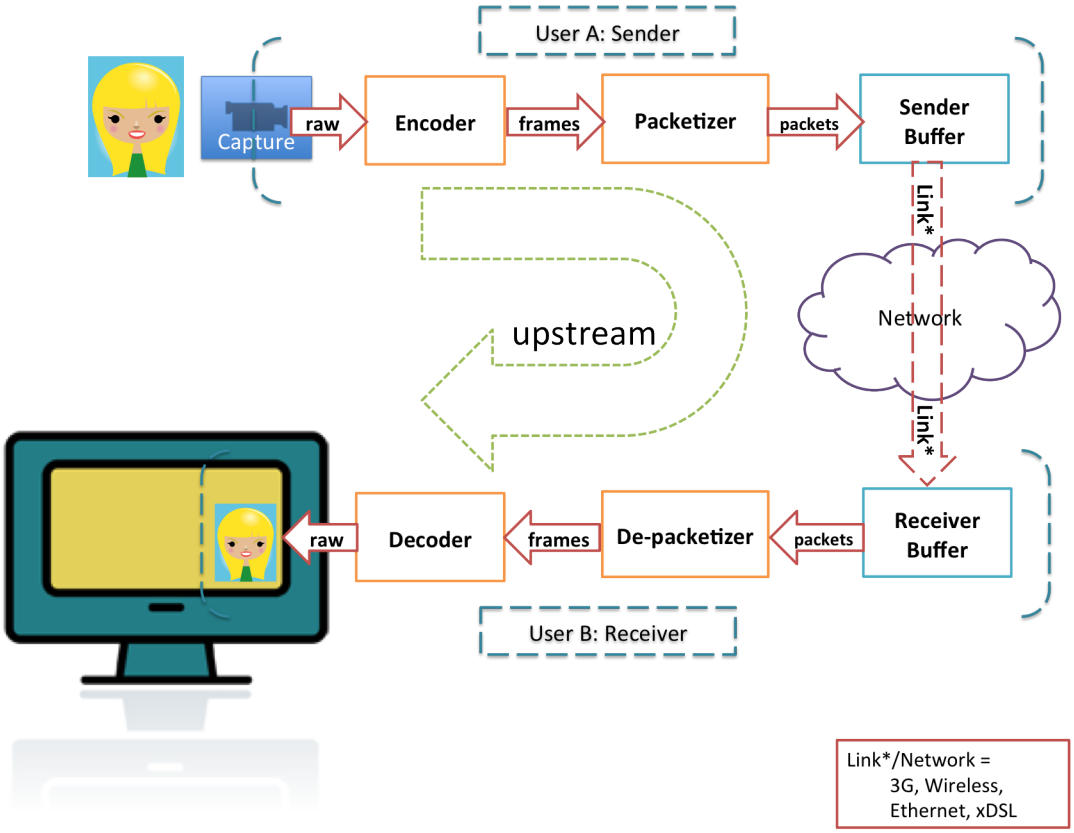


Figure 2: Video communication ecosystem

using older I- or P-frames while B-frames are predicted using both future and past I- or P-frames.

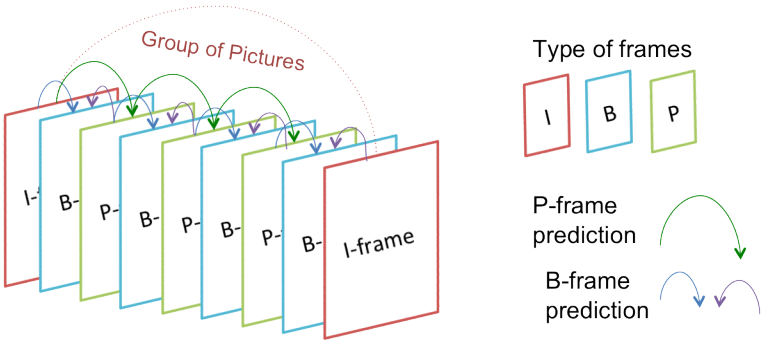
In video communication, the real-time constraints limit the size of buffers, typically shorter than 1 second. Due to their dependency on future and past packets for decoding, B-frames are not generated in video communication (as shown in Figure 3b). Consequently, this increases the speed of encoding and decoding the video.

A loss of a video frame will propagate artifacts that can be corrected by retransmission of the lost packet. However, retransmission is advisable only if $\frac{3}{2}RTT \leq delay_{max}$[3]. Alternatively, sending more I-frames in a GOP or introducing redundancy increases reliance but consumes additional channel capacity. Figure 3 compares a GOP arrangement for a streaming scenario (a) with a communication scenario (b) [23].

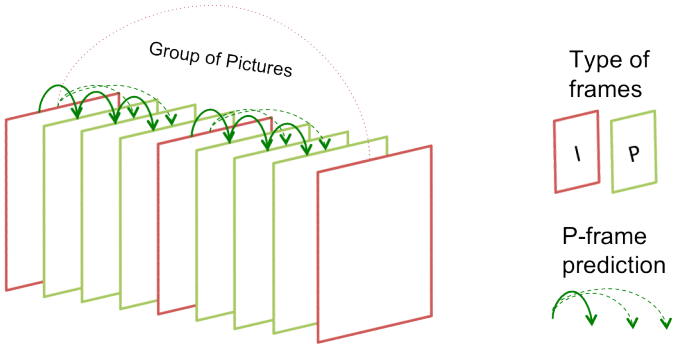
2.2 Network Traffic

Network traffic can be classified as *elastic* or *inelastic*. **Elastic traffic** is not very sensitive to dramatic changes in delay and/or throughput and still delivers reasonable Quality of Service (QoS) to the receiving application. It is the traditional type of Internet traffic; some examples of elastic applications are File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), Telnet, Simple Network Management Protocol (SNMP). Elastic traffic

[3] $delay_{max}$ is the application defined maximum delay allowed (real-time constraint).



(a) GOP for Streaming



(b) GOP for Communication

Figure 3: shows the typical Group of Pictures (GOP) arrangement for (a) Video Streaming (b) Video Communication

is typically delivered using the Transmission Control Protocol (TCP). In the case of TCP, the application can use as much of the available end-to-end capacity between the sender and receiver.

Typically, **inelastic traffic** comes from real-time sources such as stock-ticker quotes, live TV broadcast of events, Video and/or Tele-conferencing, online gaming, etc. However, inelastic traffic does not easily adapt to variations in network conditions due to real-time constraints, i.e., delayed data may be considered old and not useful. Inelastic traffic has traditionally been delivered using the User Datagram Protocol (UDP). As UDP itself does not provide error-resilience or congestion-control like TCP does, the UDP-based application can use as much of the available end-to-end capacity up to the application's data generation rate. Moreover, data generation rate may be independent of available capacity and may require dynamic compression or preferential treatment. This comes at the cost of congesting the network and being unfair to others.

For example, the Real-time Transport Protocol [24] encapsulates inelastic multimedia traffic for real-time services.

2.3 Characteristics of Multimedia Environment

Multimedia environments can be classified in many ways. We define some of the key characteristics of a Link as:

1. Channel Capacity
2. Network Latency
3. Losses

Fig. 4 visualizes the different types of the Link characteristics and associated factors.

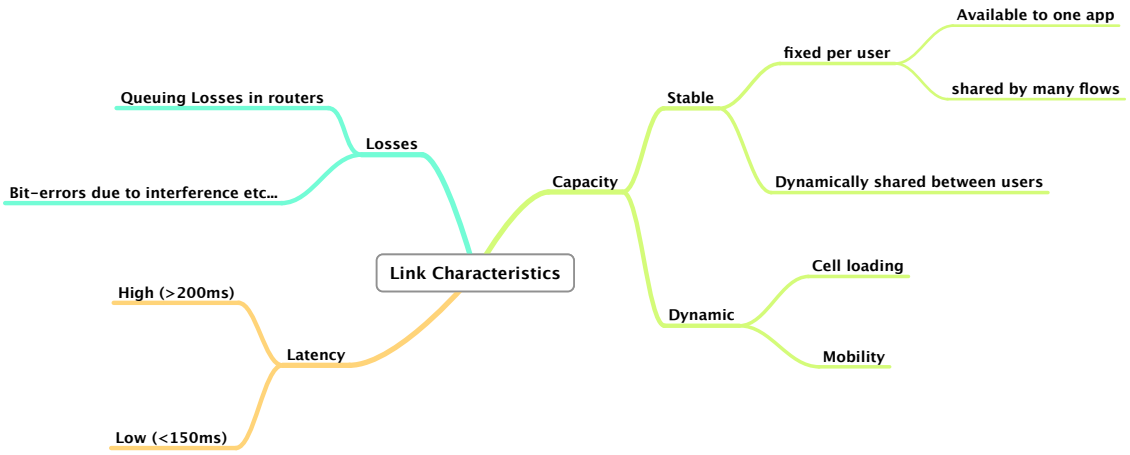


Figure 4: Different Link Characteristics in Multimedia Environments

Channel Capacity may be time-varying or static depending on the channel conditions. A wired link's capacity is static but the capacity available for each flow is variable, depending on the number of parallel flows and their usage. The end-to-end capacity on a network is constrained by a bottleneck link - a link with the least capacity for a specific flow. This constraint may be due to a) limited bandwidth in the intermediate router, or b) flows competing for the same capacity.

Mobile and wireless networks are considered **dynamic** due to the fluctuating channel conditions over the air interface. These fluctuations may be caused by user mobility or changing physical conditions and cannot be predicted beforehand. Thus, mobile and wireless networks have a varying channel capacity. Distinguishing between shared and dedicated links is a challenge.

Network Latency is the amount of time a packet takes from source to destination (one-way). Latency can also be measured as a round-trip: the time it takes from source to destination (upstream path) and destination to source (downstream path). However, due to a variation in queuing, serialization, processing and propagation delays in a network, delay/latency experienced in different directions of the same path may be asymmetric. Furthermore, one-way delay of two successive packets may vary due to the aforementioned variation.

$$\begin{aligned} One-Way\ Delay\ (OWD) = Propagation\ Delay + Queuing\ Delay \\ + Serialization\ Delay + Processing\ Delay \end{aligned}$$

In the context of video communication, we define **high latency networks** as networks with latency closer to the upper bound of real-time requirements, allowing only very little

room for queuing and processing delays in the network. On the other hand, **low latency networks** provide more room for queuing in the network. However, distinguishing the cause of low latency and high latency is a challenge because RTT values contain both queuing and serialization delay. Moreover, network latency changes as a function of load.

Losses in a packet network are due to queue losses and/or bit-errors. **Queuing or congestion losses** are packets dropped at an intermediate router due to over-saturated links. Additionally, for inelastic traffic, packets may also be discarded at the receiver due to late arrival[4]. On the other hand, **bit-error losses** occur due to packet corruption over the air interface in a wireless medium[5] or in a wired medium. Moreover, losses may also cause delays if layer-2 mechanisms attempt to do retransmissions. Distinguishing between bit-error losses and congestion losses is extremely challenging.

2.4 Types of Multimedia Environments

This section discusses in detail the three types of multimedia environments. Namely, the wired Internet environment, the mobile environment and the heterogeneous or mixed environment containing both mobile and wired Internet users. Furthermore, we define each multimedia environment in terms of the above link characteristics:

$$Multimedia\ Environment\ (M_{rr}) = f_{rr}(x \times C, \quad y \times D, \quad z \times (B + Q));$$

$$\forall \quad environments \quad \begin{cases} C = Channel\ Capacity, \\ D = Delay, \\ B = bit - error\ losses, \\ Q = Queuing\ losses \\ x, y, z\ are\ variables \end{cases}$$

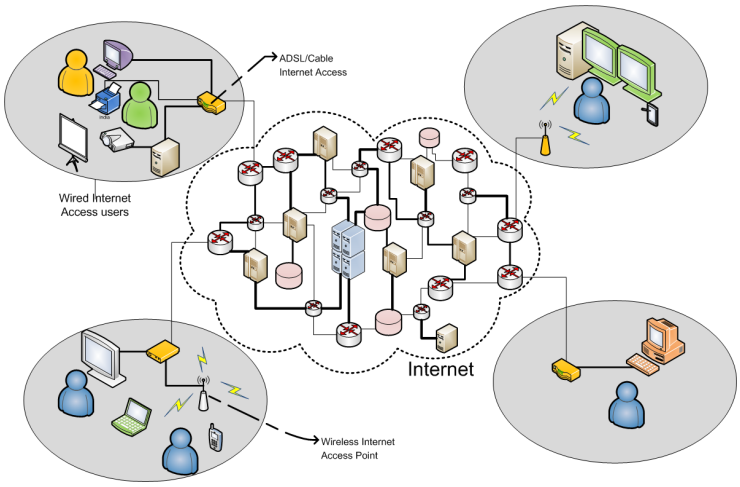
2.4.1 Wired Internet environment

The Internet has been called a network of networks, a global connection of computing systems, routers and switches. Moreover, due to variable load in the network, successive packets may experience different latencies. Therefore, the serialization delay each packet experiences is different. To summarize, packets in the Internet sent to the same destination from a source can traverse different paths and the one-way delay (latency) experienced by each could be different.

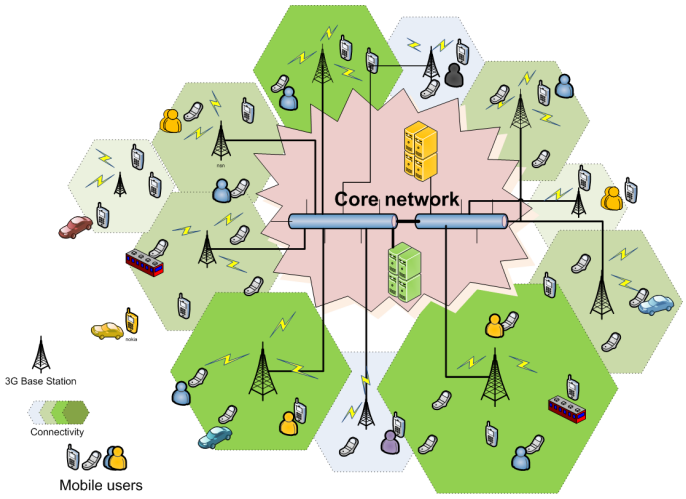
Internet access is not only used by one application on a host but also shared by multiple applications on multiple hosts. Therefore, each service sharing the common information pipe should maximize the bandwidth utilization, but not at the cost of degrading the QoS for the other services (also known as fairness). Thus, the variable link latency and the varying link utilization makes congestion control in the wired environment challenging. These scenarios are illustrated in Figure 5a.

[4] packets exceed the application-specified maximum delay (delay budget)

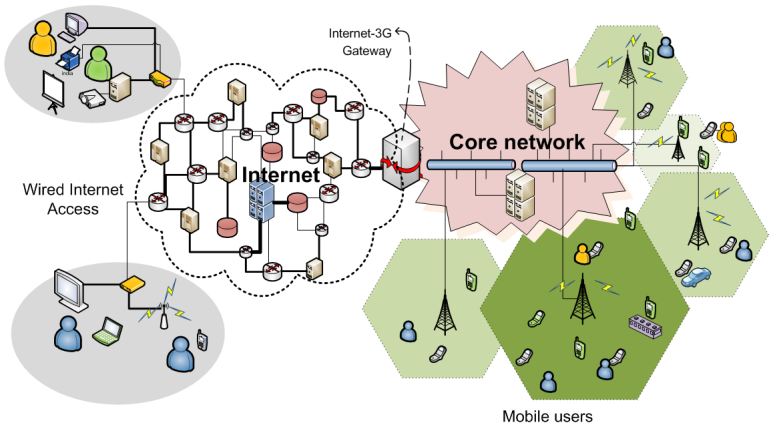
[5] Signal degradation due to fading and interference are the main causes of corruption.



(a) Wired



(b) Mobile



(c) Heterogeneous

Figure 5: shows the (a) fixed-line or wired Internet environment, (b) mobile or 3G environment and (c) mixed or heterogeneous environment

To summarize, the wired multimedia environment can be defined as,

$$M_{WIE} = f_{WIE}(constant \times C, \quad variable \times D, \quad variable \times Q + 10^{-4} \times B)$$
$$\forall \begin{cases} Wired\ environment\ there\ are \\ some\ bit\ error\ losses \\ majority\ of\ the\ losses\ due\ to\ queuing. \end{cases}$$

2.4.2 Mobile environment

In a 3G mobile network, the Radio Access Network (RAN) carries traffic from many different applications and the Radio Link Control (RLC) controls the link layer mechanisms depending on the service. The RLC operates on a layer above the Media Access Control (MAC) layer and provides services in acknowledged, unacknowledged, and transparent modes. For Media Telephony Service for IMS (MTSI), the RLC typically operates in the *unacknowledged mode* to keep link layer delays to a minimum. This allows applications to implement their own error resilience and rate-control schemes. The RLC also controls the payload size that the channel can carry at any instance of time. Due to user mobility, the payload is dependent on the physical channel conditions such as fading, interference, handovers, and cell loading.

Figure 5b presents a mobile environment; we assume that in this environment calls originate and terminate at mobile end-points. For simplicity, we assume that each Internet application uses a different PDP-context and that each PDP-context has access to independent radio resources. For a given PDP-context, the RLC controls the channel capacity available over the air interface, i.e., between the mobile phone and the base station. However, the air interface is prone to bit error losses due to physical properties of the channel. Furthermore, variable radio conditions, together with the queuing of packets at the base stations, add to the network latencies making them longer (from 30-80ms to a few 100ms). The "Core Network" represents the infrastructure or the backbone of the mobile network and is assumed to be a fully provisioned network[6] that connects base stations to the rest of the mobile network. We presume the core network to behave like the wired multimedia environment with minimal bit error losses and largely queuing or congestion losses.

To summarize, mobile environment can be defined as,

$$M_{ME} = f_{ME}(variable \times C, \quad potentially\ high \times D, \quad variable \times (B + Q))$$

2.4.3 Heterogeneous environment

A heterogeneous network contains both Internet and 3G links. The 3G network connects to the Internet through gateways and vice-versa. As shown in Figure 5c, a call can originate from a shared Internet link and traverse the Internet and, using a mobile gateway, enter into the mobile network where it will be quickly routed through the core network and delivered to the mobile user over the air interface. Quite often, intermediate links are the cause of congestion in the wired environment. In a 3G network the channel capacity is largely limited due to the variable radio conditions of the air interface, i.e., quite often the

[6] is a network with sufficient capacity that any subscriber can always complete a call to another idle subscriber.

last hop. Therefore, a heterogeneous environment exhibits properties of both the wired and wireless environments and this makes it difficult to distinguish between congestion losses and bit error losses.

To summarize, heterogeneous environment can be defined as,

$$M_{HE} = f_{HE}(variable \times C, \quad potentially\ high \times D, \quad variable \times (B + Q))$$

2.5 Summary

We notice that the characteristics of the mobile and wired networks are dissimilar and finding consistent cues that work across both the environments is challenging.

3 Congestion in a Multimedia Communication Environment

This chapter introduces the concept of congestion for multimedia and then goes on to discuss methods to control, queuing models, various congestion indicators and metrics to compare them.

3.1 Causes of Congestion

The transmission bit rate available along a communication path in IP networks is highly variable. In wireless links, quality may vary due to various interference and fading phenomena and, peered with radio layer reconfiguration and link layer protection mechanisms, can lead to varying error rates, latencies, and, most importantly, changes in the available channel capacity. Packets lost due to the physical properties of the channel are termed bit-error losses.

Furthermore, in both wired and wireless networks, varying amounts of cross traffic from other nodes (i.e., the total offered load on the individual links of a network path) may lead to fluctuations in queue size (reflected again in a path latency) and to congestion (reflected in packet drops from router queues), as shown in Figure 6. Packets lost due to congested router queues are termed as congestion losses.

To overcome congestion losses, senders have to dynamically adapt their data rate based on the networking conditions. Senders can achieve the sending rate by changing the media encoding rate and possibly other parameters (such as video quality, encoding efficiency, or redundancy) to match the available bit rate while maximizing the media quality perceived at the receiver.

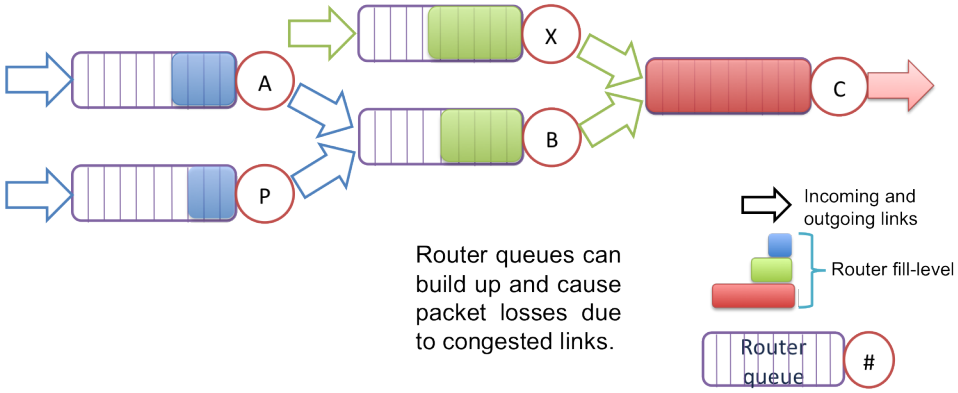


Figure 6: Congestion in a router

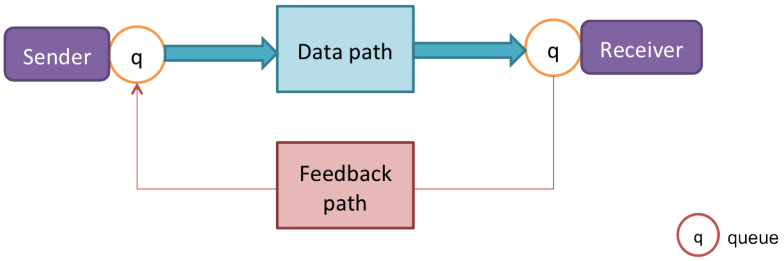
3.2 Simple Feedback Loop

Typically, inelastic traffic is tolerant to small error/loss rate because timeliness[7] is more important than 100% data delivery. On the other hand, elastic traffic is more tolerant to delay and the tightly coupled feedback guarantees better data delivery.

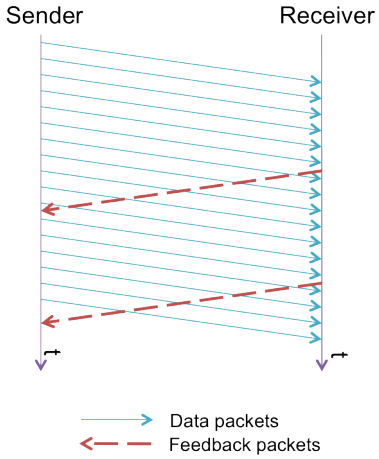
[7] Packets should arrive in time for playback.

Multimedia communication is an example of inelastic traffic. If the multimedia generation rate at all times is lower than the available channel capacity, there is no need for a feedback loop or rate adaptation. However, congestion is not the only cause of losses and available capacity may fluctuate depending on the multimedia environment. Therefore, the sender requires minimal feedback from the receiver to respond to the changing channel properties.

Figure 7a shows a simplified feedback model for video communication. The feedback path carries link characteristics that help the sender to make the rate control decisions. Figure 7b shows the flow of multimedia (blue thick line) and feedback packets (red dashed lines). The receiver routinely sends back feedback packets, based on channel heuristics. Typically, the time interval between such feedback packets is much longer than that of TCP-based traffic. However, the rate adaptation system should be able to cope with delayed and/or lost feedback packets.



(a) multimedia loop feedback



(b) multimedia feedback flow

Figure 7: Feedback loop and Feedback timing

3.3 Methods of congestion control

The decision-making process of rate adaptation can be made at the sender, the receiver, or at some intermediate node (edge or core) in the network. **Sender-driven rate adaptation** requires that the receiver be aware of the current network situation, i.e., latency experienced by a packet, current jitter buffer state at the receiver, current decoding rate, packets lost, etc., and signal this information to the sender which adapts the rate based on the received parameters. In a **receiver-driven rate adaptation** scheme, the receiver gauges the current situation based on the parameters available to it, and signals

the new required bit rate to the sender which, on receiving the new rate, adapts to it. In a **network-driven rate adaptation**, an element in the network will signal to the sender/receiver that the rate is going to drop or increase due to better or worse network conditions arising from handovers, cell-loading, etc. In these cases, the network is possibly aware of the conditions beforehand and can therefore signal the new bit rate to the appropriate node.

3.4 Types of congestion control

In **congestion avoidance**[8], the sender (or receiver) tries to detect if the link is undergoing light congestion and, based on the input, slightly increases or decreases the sending or encoding rate. For example, slight reductions or increases in round-trip time (RTT), jitter, packets in transit (PiT), etc. can be indicators for light congestion or under-utilization. Therefore, in congestion avoidance only small changes may occur in the available bandwidth.

However, in the case of **congestion mitigation**, the rate adaptation module realizes that there is already heavy congestion and needs to take a corrective action immediately. For example, high packet loss might indicate the presence of heavy congestion. Therefore, in congestion mitigation, more drastic changes in bit rate may be made to mitigate the congestion.

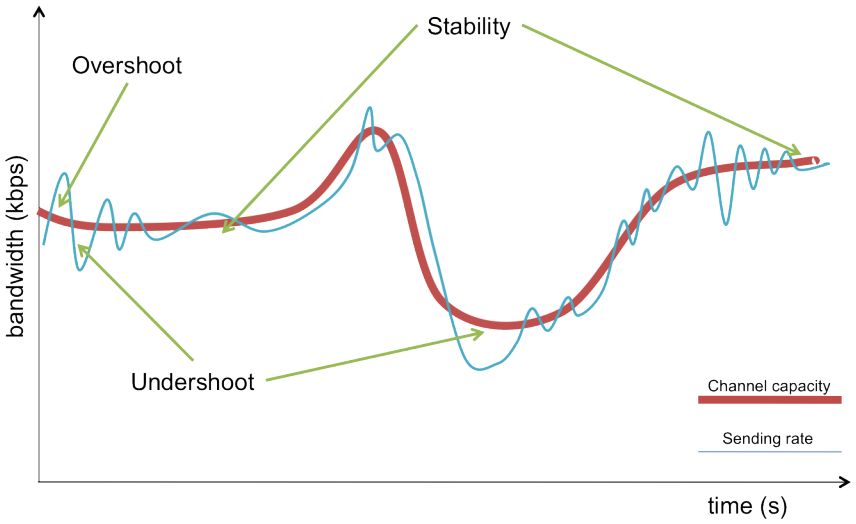


Figure 8: Shows the modes of rate adaptation

3.5 Modes for Rate Adaptation operation

Based on the types of congestion control, a rate-control algorithm operates in three modes, namely:

- **Overshoot**: When the sending rate exceeds the channel capacity. Usually overshoot causes congestion of the link and thus packet losses.

[8]We use this term, for a lack of a better word, not to be confused with TCP Congestion Avoidance algorithms.

- **Undershoot**: When the sending rate dips below the expected path capacity. This may be due to a conservative guess by the sender or the sender is trying to alleviate the stress on the congested link. Usually the sender undershoots after packet loss occurs to quickly mitigate the congestion. (See TCP Sawtooth)
- **Stability**: When the channel capacity is more or less stable. It is possible for the sender to oscillate between overshooting and undershooting. However, the sender should, over a period of time, converge to a stable channel capacity.

Figure 8 shows the above modes of operations.

3.6 Queuing Model for Conversational video communication

Unlike video streaming that is based on client-server architecture, video communication is a full-duplex communication channel. In video communication, receiver queues store only limited amounts of packets due to the real-time constraints and not because of limited storage. The real-time constraints are defined based on the user's perception of quality of the video. For instance, in video communication, audio is the master. Delay can cause audio and video to be out of synchronization thus affecting the user experience. The sender transmits packets as soon as they are generated and these packets should arrive at the receiver and rendered within the application-defined maximum delay, otherwise the packet is useless for the receiver and is discarded before decoding. The application-defined maximum delay is equivalent of the *'mouth-to-ear'* delay in audio communication.

$$Opportunity\ to\ queue\ (congestion\ control) = delay_{max} - OWD - decoding_{delay}$$

$$\forall \begin{cases} OWD & = one - way\ delay \\ delay_{max} & = delay\ budget \\ decoding_{delay} & = time\ taken\ to\ decode\ and\ render\ the\ frame. \end{cases}$$

Therefore, the opportunity to detect congestion gets smaller as network delay increases and approaches the delay budget.

A simple network queuing model at the receiver queues packets at the receiver based on the latency and allowed maximum delay. For example, video encoded at 15 FPS will generate a frame every 66 ms. Depending on the MTU size, the frame may be fragmented into multiple packets. These packets bear the same generation timestamp because they belong to the same video frame. At the receiver, these packets are fed into the decoder based on their timestamps. However, to allow frame reconstruction, these packets are fed into the decoder slightly before they can be decoded. Figure 9a shows the status of the queue with respect to the decoder's clock. Note that each multimedia packet is encoded with a strictly monotonically increasing sequence number; this helps in rearranging out-of-order packets and detecting loss. Figure 9b shows the packet flow from the sender to receiver.

3.7 Congestion Indicators in a Multimedia environment

In this section, we discuss the various congestion indicators in multimedia communication. Figure 9(a) and (b) also visually summarize the congestion indicators such as losses, one-way delay, playout delay, buffer duration and buffer size, etc.

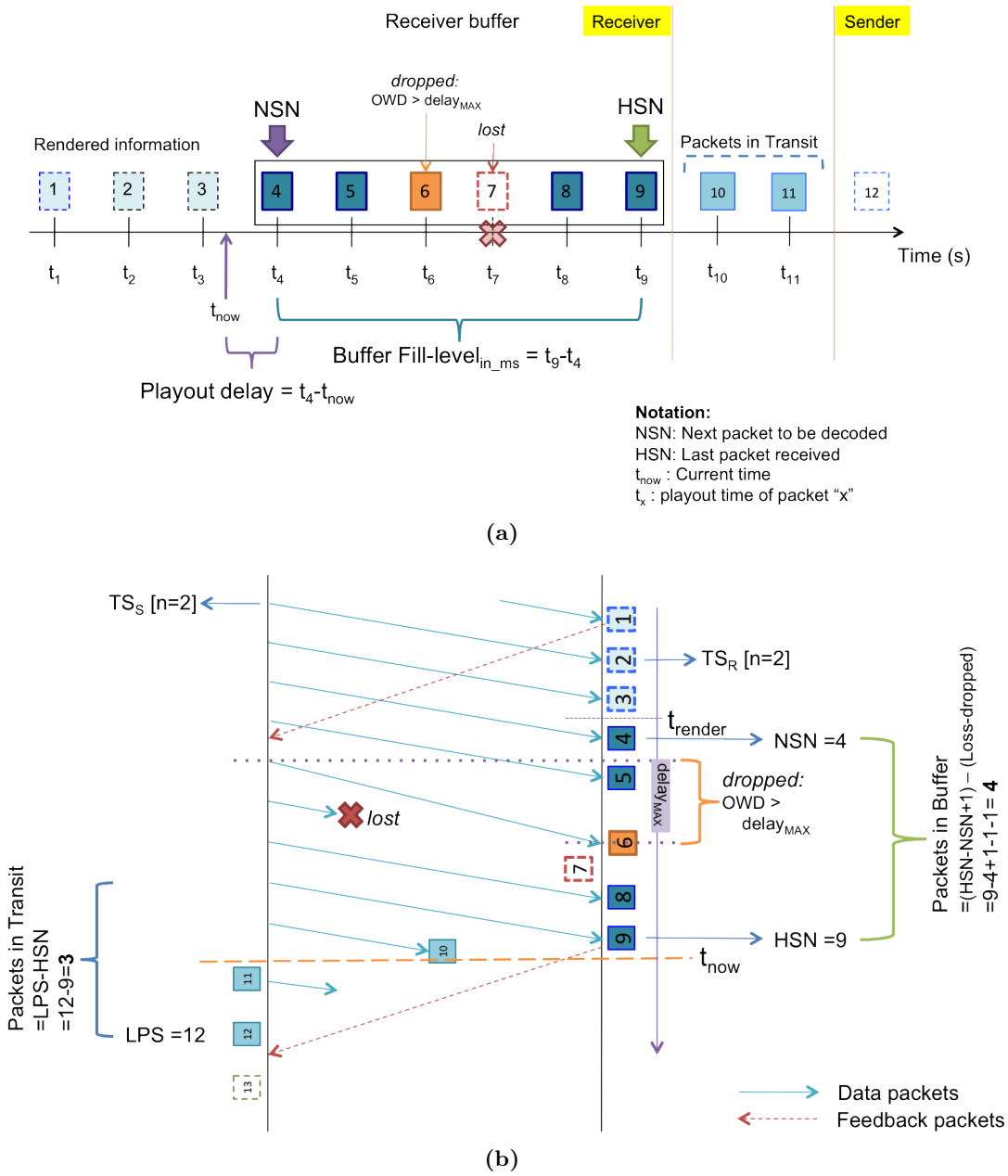


Figure 9: Visualization of the (a) Decoder Timeline during an ongoing VVoIP call, and (b) flow of packets. Both figures describe calculation of congestion indicators

3.7.1 Losses

Packet loss indicates losses in the network, i.e., non-arrival of a packet at the receiver due to the over-utilization of the link or bit-error losses. Packet loss is a strong indicator of congestion but appears only after the onset of congestion. However, before packet loss occurs in the network, the packets begin arriving late at the receiver. If the packets arrive later than the application-defined maximum delay, then these packets are useless as they cause a visible deterioration (e.g. Audio/Video out-of-sync, decoding artifacts).

The delayed packet metric provides a strong early indicator of congestion. If the delayed

packet metrics are signaled to the sender, the congestion control algorithm can avoid the subsequent packet losses. Unfortunately, these packet drops are not reflected in the packet loss metrics, as these drops do not occur in the network.

3.7.2 Receiver Rate, Sending Rate and Goodput

The receiver rate (BR_R) is the rate at which the packets are received in a given reporting interval, while the sending rate (BR_S), is the rate at which the sender sends the packets in a reporting interval. Goodput (GP) is the actual playback rate since it excludes the packets discarded by the decoder in that interval. So, a congestion control algorithm (at the sender or the receiver) can benefit from signaling the reported values.

3.7.3 Round-Trip Time (RTT)

One-way delay is the amount of time it takes the packet to reach its destination from the time it was generated. It is a function of the available bit rate at each hop[9] and the queuing delay the packet experiences at each router. Queuing delay is the amount of time the packet stays in a buffer or queue while waiting to be transmitted. Serialization delay is the time it takes for a packet to percolate through the networking interface. Therefore, the one-way delay (OWD) is a sum of delays along a path:

$$OWD = delay_{propagation} + delay_{queuing} + delay_{serialization} + delay_{processing}$$

Network latency is variable due to changes in routing path. Paths may change due to router policy, load balancing, or intermediate links may appear and disappear. Round-trip time is a sum of the upstream and downstream delay. As a result, RTT can only be calculated at the sender for a probed packet. In video communication, media packets flow in both upstream and downstream direction. Therefore, the delays are more likely to be symmetric.

$$RTT = Upstream\ Delay + Downstream\ Delay.$$

For symmetric networks,

$$RTT = 2 \times OWD.$$

$$or,\ OWD = \frac{RTT}{2}$$

Observing the changes in RTT can provide an early indication of congestion. However, smoothing the RTT (by averaging over a short interval $\approx$ few seconds) protects against over-reacting to the subtle changes in RTT. To summarize, an increase in RTT indicates early congestion while a decrease indicates congestion mitigation. Unfortunately, neither the sender nor the receiver is aware of the network's optimum RTT.

3.7.4 Jitter and Inter-arrival time (IAT) of packets

Due to variable latency, packets arrive at different times, i.e., packets sent periodically may appear aperiodically at the receiver. Congestion causes the increase and decrease in

[9] this affects serialization

inter-arrival time (IAT) between packets, also known as packet jitter. At the beginning of congestion, the packets arrive wider apart, i.e., the IAT between packets and jitter is positive, whereas at the end of congestion, the packets arrive closer to one another; i.e., the inter-arrival time of the packets is lower and jitter is negative.

Jitter and IAT are calculated at the receiver and these indicators need to be signaled to the sender for a sender-driven rate control.

3.7.5 Packets in Transit

Video encoders generate output at a constant frame rate (e.g. 15, 24 or 30 frames/second). The frames can be directly encapsulated into one packet, fragmented based on the MTU size or fragmented into slices [22]. The slice size depends on the type of frame: I-frame will fragment into more packets than a P-frame. The size of I-, B-, and P-frames depends on the type of motion being recorded: e.g. quick motion will generate larger size frames while placid motion will generate smaller size frames.

To calculate the Packets in Transit (PiT), the receiver needs to signal the last packet in the receiver's queue (HSN) and the sender needs to keep track of the last packet it sent (LPS).

$$\begin{aligned} Packets\ in\ Transit, \quad PiT &= LPS - HSN\ packets. \\ Bytes\ in\ Transit, \quad BiT &= \sum_{n=HSN}^{LPS} sizeof(n)_{bytes} \end{aligned}$$

3.7.6 Playout Delay

The application-defined maximum delay is the maximum one-way delay a packet can experience. If a packet arrives earlier than the delay budget, it is buffered until its playout time. Packets with the same generation timestamp belong to one frame and are reconstructed and rendered by decoder at the same time. An incompletely reconstructed frame may be discarded by the decoder or predicted using some advanced algorithms and older frames [22]. Therefore, *decoding delay* is the amount of time it takes for a packet to be rendered ($Time_{render}$) after entering the decoder's queue ($Time_{decoder}$). Playout-delay (PD_{NSN}) is the amount of time the first packet in the receiver's queue (NSN) will take to be rendered ($Time_{render}[NSN]$) from the current time ($Time_{now}[NSN]$), including the decoding delay.

$$\begin{aligned} delay_{decoding} &= time\ spent\ in\ the\ decoder's\ queue \\ &= Time_{render}[k] - Time_{decoder}[k] \\ &\approx order\ of\ a\ few\ to\ tens\ of\ ms. \end{aligned}$$

$$\begin{aligned} Playout\ Delay, PD_k &= Time_{render}[k] - Time_{now} \\ &= (delay_{decoding}) + Time_{decoder}[k] - Time_{now} \end{aligned}$$

$$PD_{NSN} = \begin{cases} Time_{render}[NSN] - Time_{now} \\ 0, \quad or\ empty\ in\ case\ of\ underflow \\ \infty, \quad or\ infinite\ in\ case\ of\ overflow \end{cases}$$

where $Time_{render}[k]$ is the time when the k^{th} packet is rendered and $Time_{decoder}[k]$ is the time when the k^{th} packet is sent to the decoder.

3.7.7 Packets in Buffer

A packet can be buffered at the receiver depending on the one-way delay and the delay budget. In low latency networks, more packets can be buffered at the receiver compared to high latency networks because of the fixed application-defined maximum delay. The buffering time of a packet can be calculated using the application-defined maximum delay and the difference between the rendering and generating timestamp of the video frame (similar to *mouth-to-ear* delay).

For a packet "n", the buffering time can be calculated using the reception timestamp ($TS_R[n]$), generation timestamp ($TS_S[n]$) and the application-defined maximum delay ($delay_{max}$)

$$T_{buffering}[n] = \quad delay_{max} - (TS_R[n] - TS_S[n])$$

Many packets may be stored at the receiver-side buffer awaiting playout. The change in the number of packets in the buffer can indicate underflow and congestion to the receiver. The first packet in the queue is the next packet to be decoded and is denoted as NSN. The last packet received in the queue is denoted as HSN. Using these two queue pointers the Packets in Buffer (PiB) can be calculated,

$$PiB = \quad (HSN - NSN) + 1\ packets.$$

Similarly, the time to drain the receiver's buffer (*Buffer Fill-level*$_{in\_ms}$, BFL_{in_ms}) is calculated using the generation timestamps:

$$BFL_{in\_ms} = \quad TS_S[HSN] - TS_S[NSN]\ ms.$$

The current receiver buffer occupancy in bytes (BFL_{in_bytes}) is calculated by adding up the size of all the packets in the queue:

$$BFL_{in\_bytes} = \quad \sum_{n=NSN}^{HSN} sizeof(n)_{bytes}\ bytes.$$

The receiver calculates the buffer occupancy (in time and bytes), which needs to be signaled to the sender for a sender-driven rate control. However, the sender is unaware of the initial buffering for playout and this too needs to be signaled, either during the session setup or with each feedback report.

3.8 Summary of Indicators

The congestion indicators are observed at the sender, the receiver, or are calculated collaboratively. Depending on the method of congestion control, sender-driven or receiver-driven, the congestion cues need to be signaled to the appropriate peer (sender or receiver). In chapter 4, we discuss in more detail the mechanisms of signaling these cues. Based on the place of detection, we classify the congestion cues as follows:

- At the sender
  - Sending rate
- At the receiver
  - Decoder rate
  - Jitter or inter-arrival time: is time elapsed between receiving two packets
  - Loss rate: packets lost in the network due to congestion or bit error losses
  - Discard Rate: packets discarded by the receiver due to late arrival
  - Playout Delay: time before the first packet in the queue is played-back
  - Packets in Buffer (PiB): packets stored in the receiver buffer
- Collaboratively (Receiver sends information to Sender)
  - RTT: receiver signals the Delay Since Last SR (DSLR) to help calculate RTT at the sender.
  - Packets in Transit (PiT): receiver signals the Highest Sequence Number received using that the sender can calculate PiT.

Figure 10 shows a visual summary of the congestion indicators.

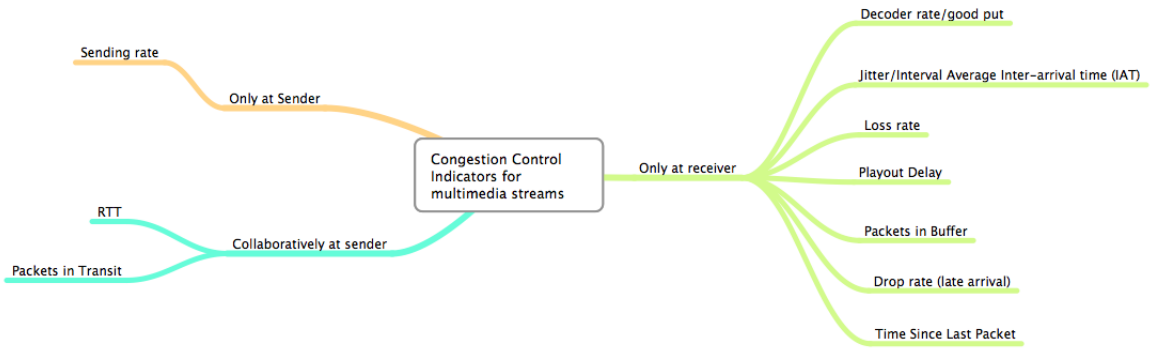


Figure 10: Types of Congestion Indicators

3.9 Metrics for Rate Adaptation

In this subsection, we introduce metrics for evaluating rate control algorithms. The sender's goal is to minimize losses at the receiver. Losses are caused by congestion or bit-errors and are detrimental to video quality. Although, video communication is toler-

ant to small amount of losses, they should be avoided. Bit-errors are due to the physical properties of the network and cannot be predicted ahead of time. Congestion losses are due to over-utilization of the links and may cause long delays or congestion-induced losses at the router.

Rate control algorithms observe the congestion cues, and react to the changes in the cues, by modifying the encoding/sending rate to match the available end-to-end bit rate.

Figure 11 schematically shows the instant per-packet delay over time observed at the receiver. The sender reacts to the changes in the available path bit rate. Exceeding the available path bit rate may lead to a temporary increase in per-packet delay until the rate adaptation measures take effect and, optionally, to packet losses if the queue capacity is exceeded.

For the delay, we define three values:

1. Threshold 1 refers to the mean one-way delay observed under normal operating conditions; this value may be defined statically according to expectations for a certain environment, or determined dynamically. This reflects the mouth-to-ear or camera-to-eye delay.
2. Threshold 2 defines the maximum acceptable one-way delay for a certain scenario after which rendering of the received video packets is no longer meaningful and packets arriving later than threshold 2 will be considered lost. Threshold 2 may be, e.g., 100-500ms for video, since the human eye is more tolerant to video glitches in video communication [25].
3. The short-term delay peak reflects the maximum delay peak encountered during a rate adaptation operation.

For losses, we consider two values:

1. Packets lost in the network due to bit errors and/or increased queue lengths or overflows (e.g., caused by drop-tail or RED queue management).
2. Packets discarded at the receiver because their arrival delay violated threshold 2 above.

3.9.1 Instant and Average Encoder/Decoder Rates

The instantaneous (per second) and average sending rate is calculated at the encoder. Similarly, the receiver rate is calculated upon receiving the packet at the receiver. The goodput is calculated after discarding the late packets and before passing the packets to the decoder. [Refer section 3.7.2 for related information].

3.9.2 Average BW Utilization (ABU)

ABU is a weighted average between the goodput (or encoding, receiving bit rate) and the available bit rate. We represent over-utilization ($> 100\%$) as 100% utilization because over-utilization causes congestion and therefore congestion losses.

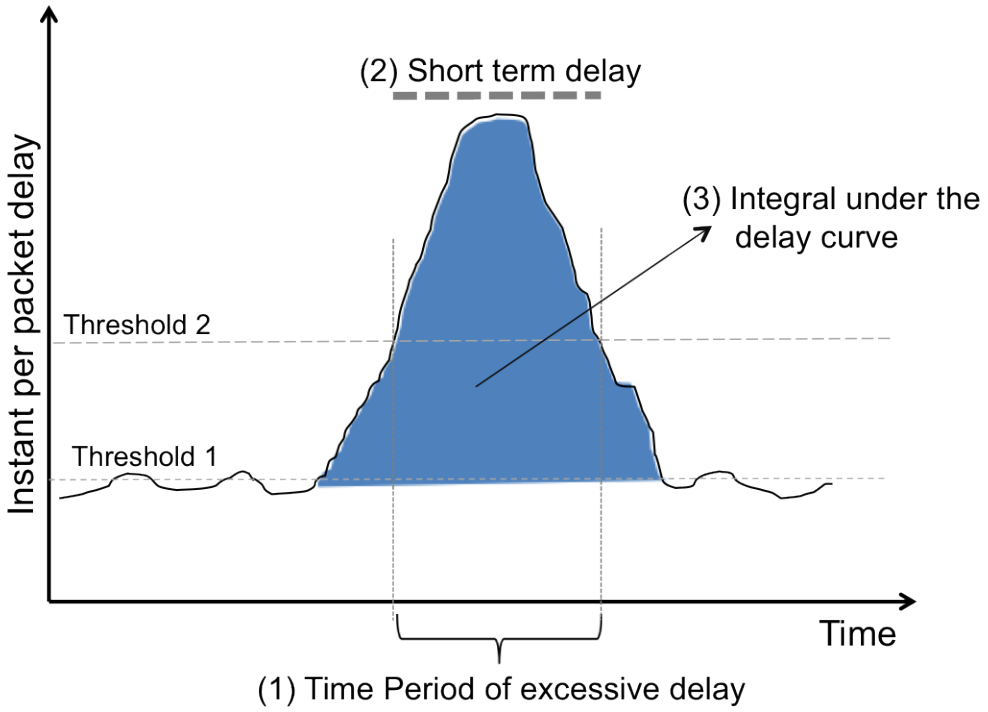


Figure 11: Metrics for Rate Adaptation

$$Average\ BW\ Utilization\ (ABU) = \frac{\int s(t)}{\int f(t)} \quad or \quad \frac{\int r(t)}{\int f(t)} \quad or \quad \frac{\int g(t)}{\int f(t)}$$

$$Usually, \quad s(t) \geqslant r(t) \geqslant g(t), \qquad \forall t \rightarrow duration of call$$

$$\begin{cases} f(t) = Channel\ capacity \\ s(t) = Sender's\ encoding\ rate \\ r(t) = Receiver's\ receiving\ rate \\ g(t) = Receiver's\ goodput \end{cases}$$

3.9.3 Peak Signal-to-Noise Ratio (PSNR)

PSNR is the ratio between the maximum possible power of a signal and the power of a noisy signal. The maximum power signal is presumed to be the original signal while the noisy signal is the received data signal that has undergone the cycle of compression-transmission-decompression.

The MSE represents the cumulative squared error between the compressed and the original

image. The lower the value of MSE, the lower the error [26],

$$MSE_{monochrome} = \frac{\sum_{M,N}[I(m,n) - J(m,n)]^2}{M * N}$$
$$\forall M, N \; is \; the \; size \; of \; the \; images,$$
$$I = original \; frame,$$
$$J = frame \; after \; decompression \; of \; frame$$

However, the above equation is for monochrome frames. The human eye is most sensitive to intensity (or luma) information [22]. The Y (luma), in YCbCr represents a weighted average of RGB channels. G is given the most weight, again because the human eye perceives it most easily. With this consideration, the PSNR is only computed on the luma channel.

$$MSE_{color} = \frac{1}{3} \times MSE_{monochrome}$$
$$= \frac{1}{3} \times \frac{\sum_{M,N}[I(m,n) - J(m,n)]^2}{M * N}$$

PSNR is defined as a function of Mean Square Error (MSE):

$$PSNR = 10 \times \log \frac{R^2}{MSE}$$
$$= 20 \times \log \frac{R}{\sqrt{MSE}}$$

In the previous equation, R is the maximum fluctuation in the input image data type. For example, if the input image is a double-precision floating-point data type, then R is 1. If it is an N-bit unsigned integer data type, then $R = 2^N - 1$.

While PSNR is the most widely used objective video quality metric, it does not perfectly correlate with perceived visual quality due to the non-linear behavior of the human visual system [27]. It has been observed that the pictorial quality perceived by the human visual system is also affected by the overall general impression of the viewed video stream. In addition, recent studies have shown that human visual system awards higher response to more salient image locations and features [28, 29].

3.9.4 Delta loss-rate (DLR)

We define DLR as the additional losses caused by the operation of the rate adaptation algorithm on top of the inherent losses caused by the wireless nature of the link. This delta loss rate occurs whenever the uplink and downlink network buffers overflow, and it is therefore induced by congestion losses.

3.10 Summary

In this chapter, we notice that there are several congestion cues that can be considered as input for a rate-control algorithm. However, it is challenging to ascertain their applicability across different environments. The chapter also defines some metrics that will make it easier to make comparisons between different rate-control techniques.

4 Basic RTP/RTCP and header extensions for congestion control

In this chapter, we introduce the Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP). We also introduce the RTCP extensions used in congestion control.

4.1 Real-time Transport Protocol

RTP is an application layer protocol that provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. RTP transmits the data over IP using a variety of transport layer protocols such as UDP, TCP, and Datagram Congestion Control Protocol (DCCP). RTP encapsulates real-time data that requires immediate transport as well as immediate consumption at the receiver. Therefore, RTP is suitable for scenarios such as live streaming and broadcast, video-on-demand services, as well as conversational services.

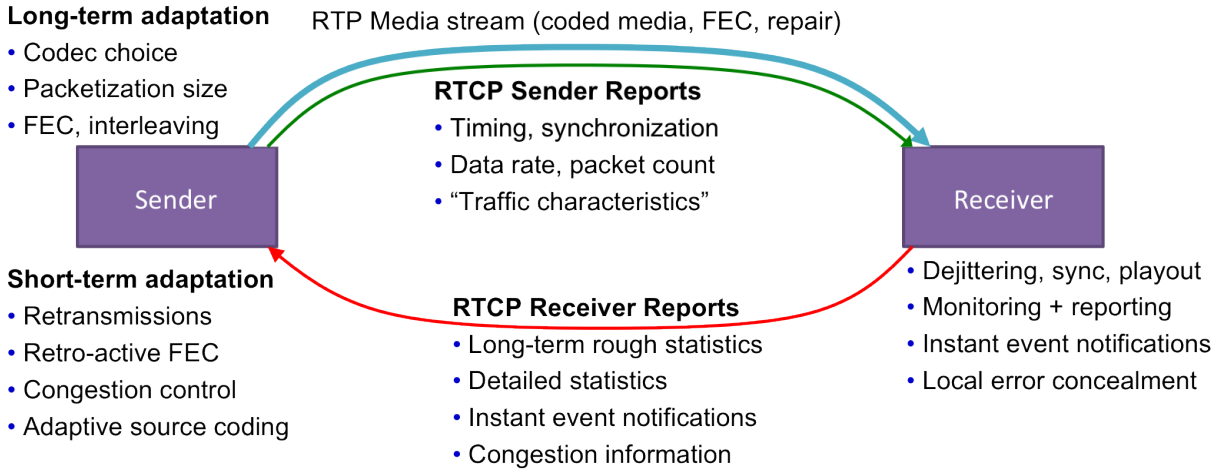


Figure 12: Overview of RTP and RTCP

Source: Jörg Ott, "Network Multimedia Protocols and Systems"

The basic system features are described in Figure 12. The RTP packets typically carry media data but have been extended to provide error protection by Forward Error Correction (FEC) and retransmission, as well as security. RTP uses RTCP reports for monitoring the end-to-end media delivery. There are two types of RTCP reports, namely, RTCP Sender Report (SR) and RTCP Receiver Report (RR). The SR carries important information related to media playout while the RR carries long-term and instantaneous connection statistics [24].

RTP enables media identification and media synchronization using payload types, sequence numbering, and time stamping. Figure 13 shows the protocol header representation of a basic RTP packet. The RTP receiver identifies the payload based on the Payload Type (PT). If the receiver does not understand the payload type it will ignore the packet. The sequence number increments by one for each RTP packet transmitted by the sender. Therefore, no two packets will have the same sequence number. The timestamp is the sampling instant of the first byte in the RTP packet or video frame [24]. Two packets

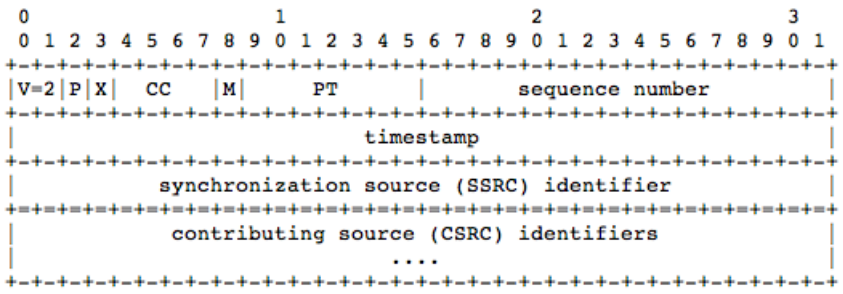


Figure 13: Block representation of a basic RTP Packet

may have the same RTP timestamp if they belong to the same frame and the frame was fragmented due to MTU size limitations.

4.2 RTCP Feedback Signaling

Figure 14 describes the basic RTCP feedback signaling. RTP packets are transmitted based on their RTP timestamps. The RTCP packets are scheduled periodically but to keep the reporting minimal, they are allowed to only consume a fraction of the media rate, typically up to 5%. For high bit rates, the 5% can be quite substantial so, RTP recommends a minimum RTCP interval of 5 ± 2.5 s for point-to-point unicast communication [24]. However, in video communication, if the congestion lasts more than a few seconds, it affects the user experience of the video call and the call is more likely to be terminated by the user. The Audio-visual Profile (AVPF) [30] removes the 5 ± 2.5 s minimum interval restriction and allows higher reporting.

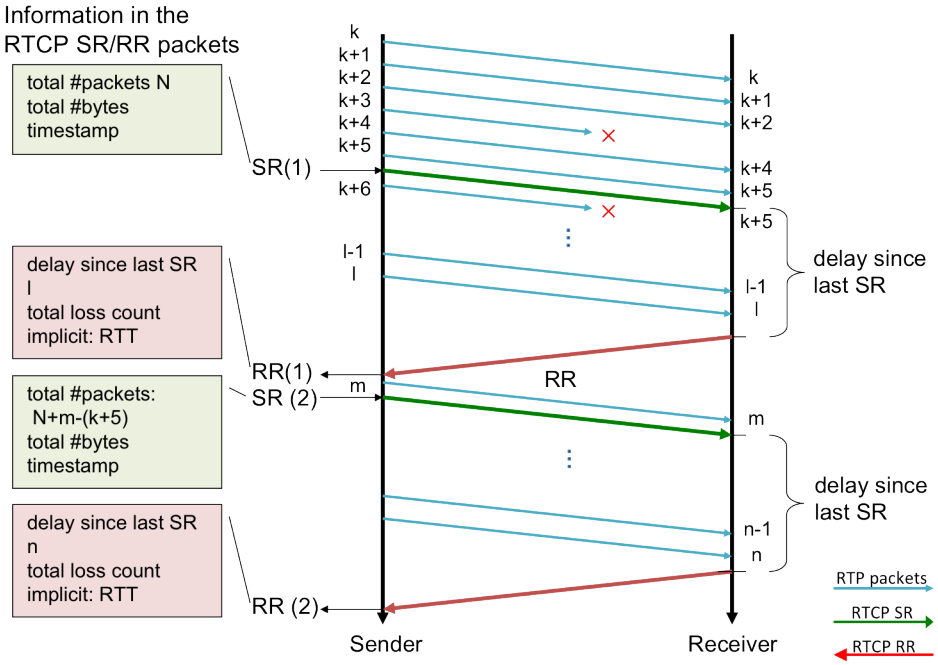


Figure 14: RTP and RTCP Flow Diagram

Figure 15 shows the difference between the two RTCP intervals. In Figure 15 (b), for simplicity, we depict reducing the normal RTCP interval (T_{rr}) by half but the AVPF RTCP interval (T_{avpf}) may be shorter, consuming up to 5% of the media rate.

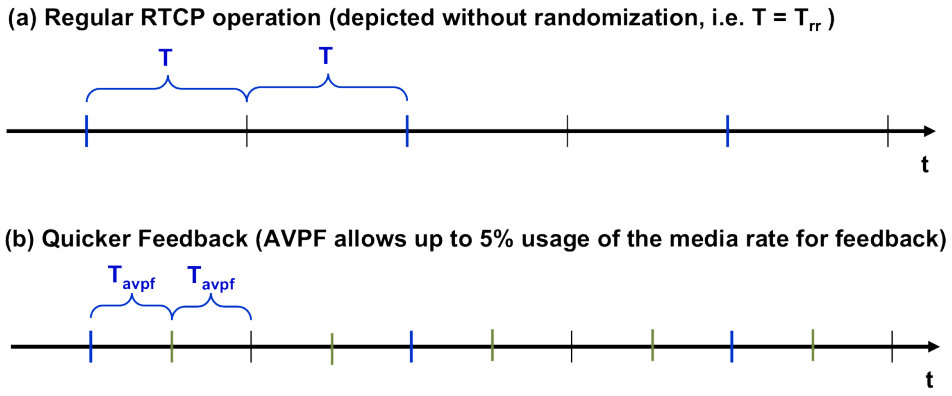
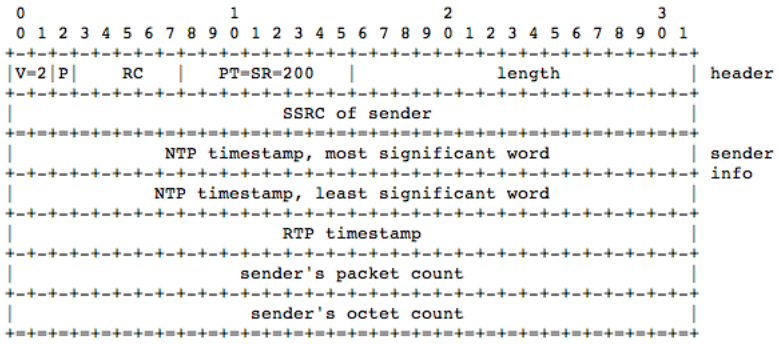
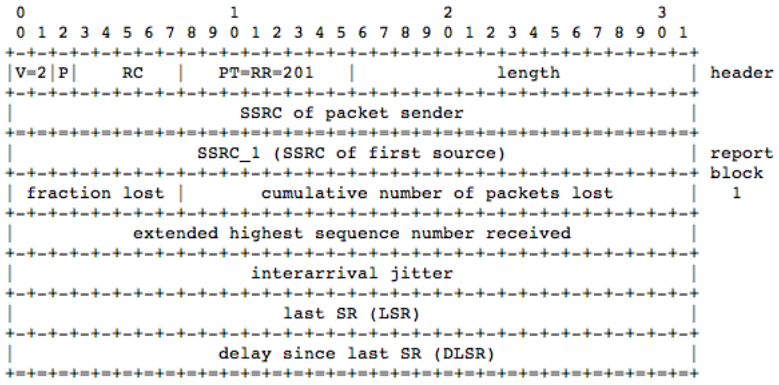


Figure 15: RTCP Feedback Interval



(a) RTCP SR



(b) RTCP RR

Figure 16: Block representation of RTCP (a) Sender Report (SR), (b) Receiver Report (RR)

Figure 16a shows the protocol header representation for RTCP SR. The RTP timestamp (RTP-TS) and NTP timestamp (NTP-TS) provide the synchronization information needed for playback. Total packet count, total octets and the current session length give the session's average packet rate and bit rate. Calculating the sender's packet count and

packet size between consecutive SRs gives the sender's interval packet rate and interval bit rate (BR_s).

Similarly, RTCP RRs report the link conditions as perceived by the receiver. Figure 16b, shows the protocol header representation for RTCP RR. The fraction lost and cumulative loss provides the sender with loss conditions of the link. The extended highest sequence number (HSN) received notifies the sender if any new packets were received since the last interval. The Last SR (LSR) and Delay since last SR (DSLR) help in RTT calculations at the sender. The RTT along with interval jitter provides the delay characteristic of the link.

4.3 Congestion cues in Normal RTCP Reports

In this section we discuss in detail the congestion cues reported in the RTCP SR and RTCP RR.

4.3.1 Sending and Receiver Rate

The difference between the sending rate and receiver rate can provide congestion cues. The receiver can calculate the sending rate (BR_s) from the sender's octet count in the SR and by storing the pertinent values from the previous SR report:

$$Sending\ Rate\ (BR_S) = \begin{cases} \frac{\sum_{i=last\_SR_{LPS}}^{LPS} sizeof(i)_{bytes}}{Time_{now} - Time_{last\_SR}} & (at\ sender) \\ \\ \frac{octet_{now} - octet_{last\_SR}}{Time_{now} - Time_{last\_SR}} & (at\ receiver) \end{cases}$$

The receiver rate (BR_R) is not signaled by the receiver, but a rough estimated can be made by the sender using the information in the normal RTCP RR report. Additionally, if packets are discarded due to late-arrival, the receiver can calculate the goodput (GP). The equations for calculating the receiver rate or goodput are as follows:

$$Receiver\ Rate\ (BR_R) = \frac{\sum_{i=last\_RR_{HSN}}^{HSN} sizeof(i)_{bytes}}{Time_{now} - Time_{last\_RR}}$$
$$\forall \quad i \ni \{lost\}$$

$$Goodput(GP) = \frac{\sum_{i=last\_RR_{HSN}}^{HSN} sizeof(i)_{bytes}}{Time_{now} - Time_{last\_RR}},$$
$$\forall i\ satisfies \Longrightarrow,$$
$$delay_i = TS_R[i] - TS_S[i] \quad < \quad delay_{max}$$
$$i.e.,\ i \ni discarded$$

The information signaled in the SR and the local information (for e.g. losses, discarded packets, jitter) available at the receiver could provide input to a receiver-driven rate control. The only caveat is that, the receiver is not aware of the network latency or RTT.

4.3.2 Delay characteristics

The receiver signals loss and delay information to the sender. The sender could use jitter, RTT and packets in transit information for making congestion control decisions.

The receiver calculates jitter for a reporting interval using the following equation [24]:

$$\begin{aligned} J_m = \quad & J_{m-1} + \frac{|D_{m-1,m}| - J_{m-1}}{16} \\ & \implies m-1, m \; are \; successive \; packets \end{aligned}$$

The sender could possibly spot congestion trends by observing the reported jitter over several reporting intervals. This trend can be an input to a congestion control algorithm.

The sender can calculate RTT using the Last SR (LSR) and Delay since last SR (DSLR) signaled in the RTCP RR. The RTT is calculated by the following equation:

$$\begin{aligned} Round\ Trip\ Time,\ RTT = \quad & TS_R[RR_{last}] - DSLR[RR_{last}] - TS_S[SR_{last}] \\ & \begin{cases} TS_R = & RTP\ Timestamp\ at\ Receiver, \\ TS_S = & RTP\ Timestamp\ at\ Sender, \\ SR_{last} = & LastSenderReport \\ RR_{last} = & LastReceiverReport \end{cases} \end{aligned}$$

Similarly, observing the RTT over several reporting intervals may provide cues to a sender-driven congestion control algorithm.

4.4 RTCP extensions for congestion control

Today's RTCP, like RTP, has been extended to provide feedback for error resilience and advanced QoS signaling for various multimedia applications. RFC3611 [31] defines RTCP extension reports (RTCP-XR) that communicates extended receiver statistics to the sender, while RTCP extensions for Audio/Video profiles (AVPF) [30] define transport layer and protocol-specific feedback messages. AVPF together with RTCP-XR can be used for error concealment and repair [32].

The RTCP-XR defines three categories of reports. The first category consists of packet-by-packet reports on received and lost RTP packets. The second category reports reference time information between RTP participants. In the third category, the receiver reports summary statistics that are more detailed than a normal RTCP report. The statistics report max, min, average, standard-deviation of losses, duplicates, jitter, and Hop Limit (TTL) values. They also report audio metrics for VoIP monitoring.

Temporary Maximum Media Stream Bit Rate Request(TMMBR) and Temporary Maximum Media Stream Bit Rate Notification (TMMBN) are Codec Control Messages (CCM) [33] and can be used for rate control. TCP Friendly Rate Control (TFRC), an equation-based congestion control algorithm [34] also defined as a profile for Datagram Congestion Control Protocol (DCCP) [35], calculates the new bit rate based on average packet size, RTT, and loss-rate [36]. A draft specification [37] extends TFRC [38] for multimedia applications by defining new RTP and RTCP extensions to control the algorithm. Furthermore, 3GPP defines a new congestion control mechanism using an RTCP extension called New Application Data Unit (NADU) [39] for real-time video streaming services.

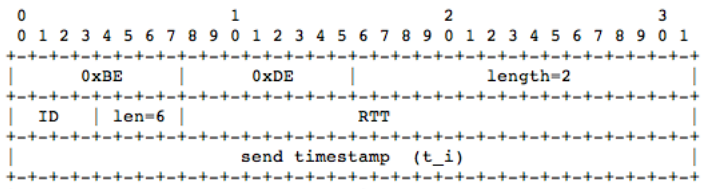
4.5 Rate adaptation schemes and techniques

Rate adaptation schemes are techniques to signal congestion cues or congestion notification between the caller and the callee. The sender responds to these by adapting the sending rate to the end-to-end link capacity.

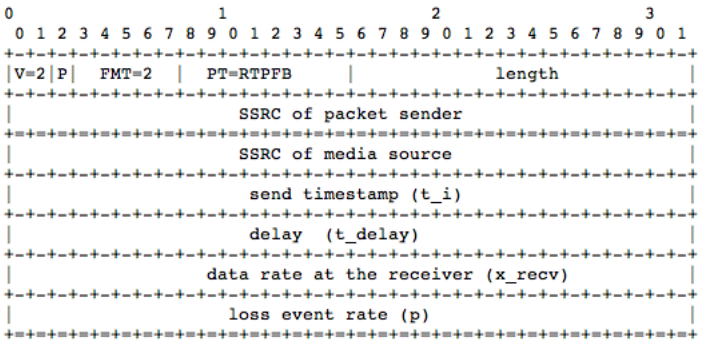
4.5.1 TCP Friendly Rate Control (TFRC)

TFRC is an equation-based congestion control scheme for unicast flows operating [34] in a best effort Internet environment and fairly competes for bandwidth with other TCP flows. It is implemented as a profile in the Datagram Congestion Control Protocol (DCCP). TFRC is a sender-driven rate control mechanism that calculates a TCP-friendly data rate based on the current network conditions, as represented by the average packet size, RTT and loss rate [38].

RTP/TFRC [37] proposes an extension to TFRC [36] for multimedia applications. It extends the RTP/RTCP feedback loop to control the algorithm instead of using TCP. TFRC requires the receiver to send a feedback packet at least once per RTT or per packet received. This ensures timely reaction to congestion. RTP/TFRC redefines the timing rules in AVPF [30] for very short RTTs ($< 20ms$) because sending feedback once-per-RTT may exceed the 5% bandwidth upper-limit imposed by RTP/AVPF [30]. Assuming a 100 byte RTCP packet sent once per RTT, it would require a 0.8 Mbps media rate for a 20ms RTT or an 8 Mbps media rate for a 2ms RTT.



(a) RTP for TFRC



(b) RTCP for TFRC

Figure 17: TFRC adapted for multimedia sessions: (a) RTP header extension (RTP/TFRC), (b) RTCP extension (TFRC-FB)

RTP/TFRC extends the RTP header extension to include the sending timestamp and the calculated RTT. This enables the receiver to schedule the RTCP messages and not exceed the RTCP rate limit. Furthermore, the sending timestamp enables precise RTT

calculation. Figure 17a shows the RTP extension for TFRC. The sending timestamp and delay since last report (t_delay) help precisely calculate the RTT at the receiver. Additionally, the TFRC receiver signals the interval receiver rate (X_recv) and loss event rate. Figure 17b shows the RTCP extension for TFRC-FB. Using the signaled information, an RTP/TFRC sender can calculate the new media rate using the following equation:

$$X_{kbps} = \frac{size_{bytes}}{R \times \sqrt{\frac{2*b*p}{3}} + t_{RTO} * (3 * \sqrt{\frac{3*b*p}{8}}) * loss * (1 + 32 * p^2)}$$

$$\begin{cases} R = RTT \\ b = Acknowledged\ packets \\ p = loss - events, \quad \forall p \in [0.0, \cdots, 1.0] \\ t_{RTO} = re - transmission\ timeout\ value\ (in\ secs) \end{cases}$$

4.5.2 Temporary Maximum Media Bit-rate Request (TMMBR) / Temporary Maximum Media Bit-rate Notification (TMMBN)

TMMBR/TMMBN are codec control messages defined for AVPF in RFC5104 [33]. Figure 18(a) and (b) describe two example use-cases for TMMBR/TMMBN in a point-to-point (PtP) unicast scenario. In use-case (a), the receiver sends a TMMBR request to the sender to limit its maximum sending rate to the requested value; i.e., the sender may choose a media rate less than or equal to the one signaled in TMMBR. Alternatively, the receiver can also send a request with multiple choices of media rates; the sender may choose one of them or ignore the request. The chosen rate is communicated back the receiver using a TMMBN. In use-case (b), TMMBN is used as a notification by a middlebox to a peer to notify the bounding rate it is using. The receiver may then generate a TMMBR request for the sender for this limiting rate.

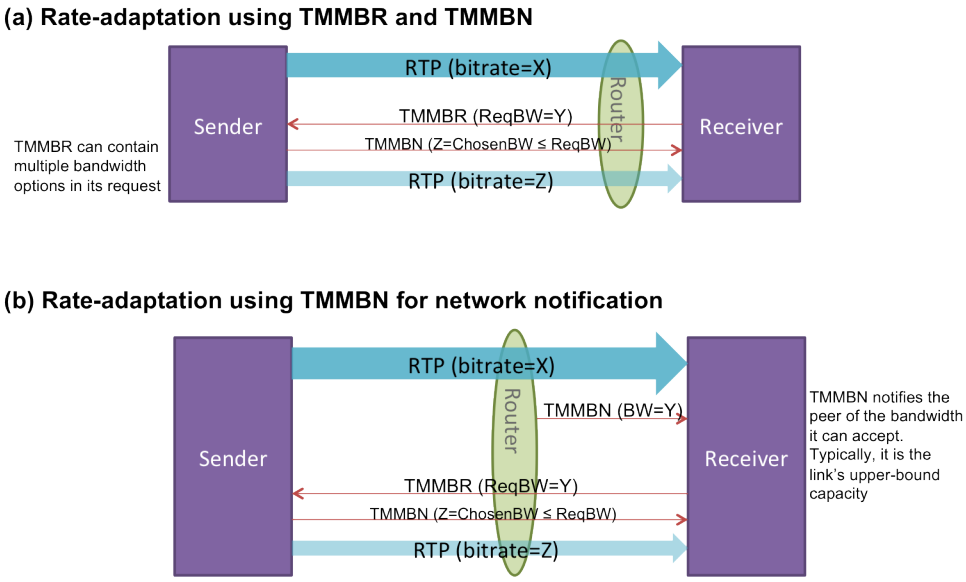


Figure 18: RTCP header extension for TMMBR/TMMBN specified in Codec Control Message

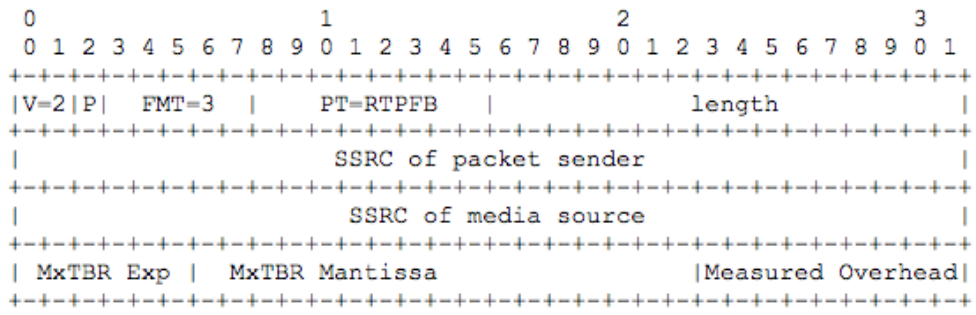


Figure 19: RTCP header extension for TMMBR/TMMBN specified in Codec Control Message

Figure 19 shows the data format block for reporting TMMBR/TMMBN. The requested bit rate is represented as combination of exponent and mantissa.

4.5.3 Next Application Data Unit (NADU) for Streaming video

NADU indicates the status of the receiver-side RTP buffer to the sender [40,41]. NADU is defined [39] in 3GPP for a Packet-switched Streaming Service (PSS). It signals three key elements that help the sender to recreate the receiver-side buffer. They are: Playout delay of the first packet in the queue, the sequence number of the first packet in the queue (also the associated NAL Unit Number, if H.264) and the free buffer space. Figure 21 shows the block representation for signaling NADU [39].

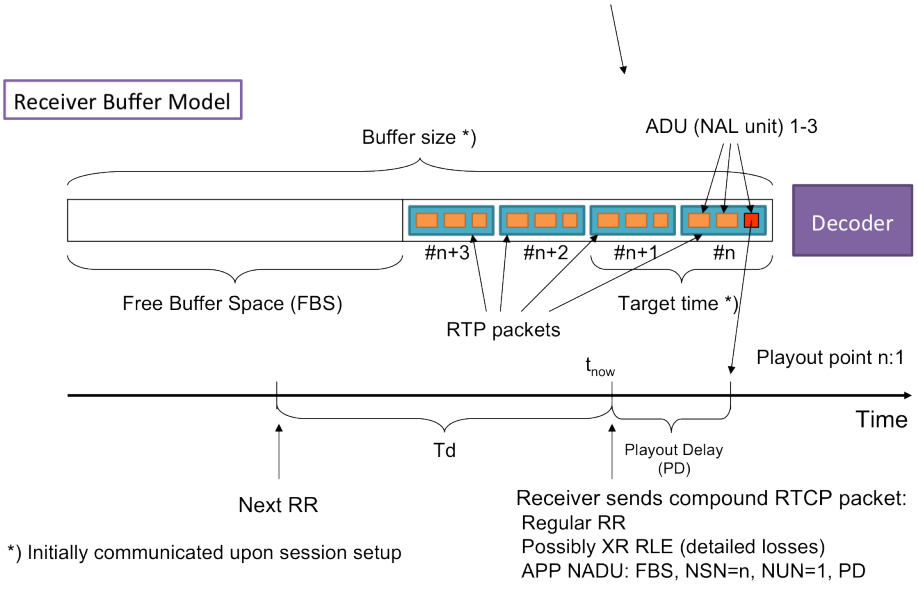


Figure 20: Receiver Side Model for NADU

Using the normal RTCP RR along with the NADU packet, the sender can accurately calculate the buffer fill-level[10], and the number of packets in the buffer. Additionally, the receiver can provide extended network characteristics by using RTCP-XR [31]. For example, run-length encoded packet reception can show the packet loss pattern that can

[10] the amount of time it will take to drain the receiver buffer

help in discovering if the losses occurred at the beginning or at the end of the reporting interval. These elements are described in the receiver-side model shown in Figure 20. Using this type of information, the sender can accurately estimate the receiver's playout buffer and then choose a new encoding bit rate to expand or reduce the receiver's playout buffer.

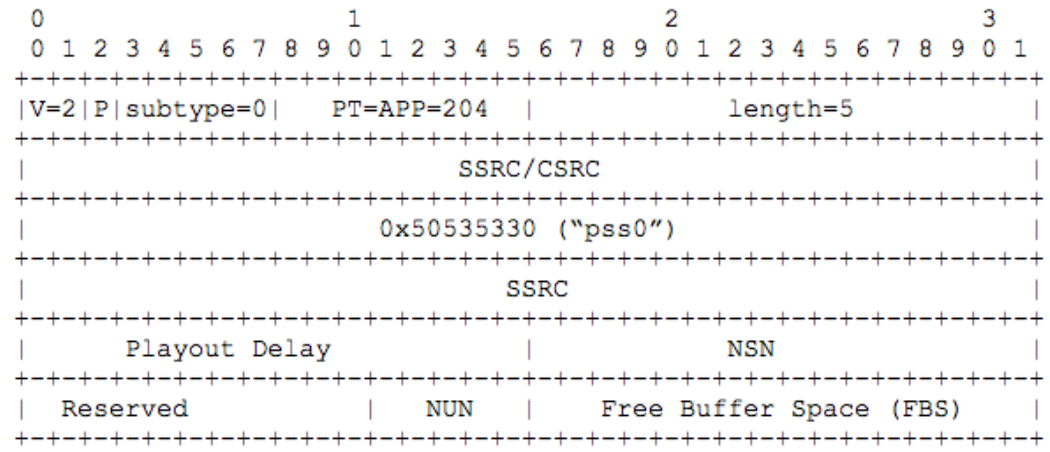


Figure 21: RTCP header extension for reporting NADU)

4.6 Summary

We notice that there are many RTCP extensions that can be used in combination with the normal RTCP report to make rate-control decisions. However, each additional extension adds to the size of the feedback packet that can adversely affect the RTCP interval by making them longer. Appendix A tabulates the header overheads for common protocol headers and RTP/RTCP extensions.

5 Simulation Environment

In this chapter, we explain the design and implementation details of the multimedia environments in a network simulator.

5.1 Simulator Design

Network Simulator 2 (*ns-2*) is used to simulate the different scenarios. *Ns-2* is a discrete event simulator, wherein each event in the system is maintained and scheduled in a chronological order. The working is described aptly by D. Mahrenholz et. al in [42]:

> "*Ns-2* is a single threaded discrete event simulator. The ns-2 scheduler maintains an internal virtual clock. The simulator objects use this virtual clock as a time reference. The scheduler also maintains a timely-ordered list of events and processes them one by one. It takes the next earliest event from the list, advances the virtual clock till the firing time of the event, and executes it till completion. Then the control returns back to the scheduler to execute the next event. There are two basic scheduler categories that differ in the method used to advance the virtual clock - non real-time and real-time. In a non real-time scheduler, the virtual clock simply jumps between firing times of consecutive events. The real-time scheduler in contrast tries to execute events in the actual moments in real-time. It uses the physical clock of the machine as a real-time reference. If the firing moment of a next earliest event is in the future, the scheduler waits until that moment in time."

Typically, in *ns-2*, end-points generate virtual traffic or packets that carry dummy data but are of correct length. This method is sufficient as long as the reception pattern of the packets is adequate to draw conclusions about the protocol or link performance. However, PSNR is an important metric for comparing multimedia streams - comparison between encoded and decoded frames is *critical* to draw conclusions.

To make true comparisons with real-time systems, we built an interface that connects an H.264 codec with the sending and receiving node inside *ns-2* [32]. This interface link between the codec and simulator is established over TCP/IP[11]. The RTP packets transparently flow through the interface. In addition, we implemented a lightweight control protocol to provide two functions:

1. Synchronize the media clock of the codec with that of the timer in *ns-2* to provide real-time emulation.
2. Control messages to modify encoder parameters (bit rate, quantization factor, GoP size, slice size indication, reference picture selection, RTP receiver buffer status, playout delay, packet retransmission etc).

Figure 22 shows the system overview of the simulator. We use Nokia's H.264/SVC codec [43] for encoding and decoding video. The REAL interface packetizes H.264 video frames into RTP packets based on RFC3984 [44]. It provides callbacks for the codec to generate codec-specific messages such as Reference Picture Selection, Picture and Slice Lost Indication, Full Intra Request etc. Additionally, the codec [43] provides APIs to the RTP stack to change the encoding rate, slice size and also provide metrics such as decod-

[11] The interface tries to imitate inter-process communication.

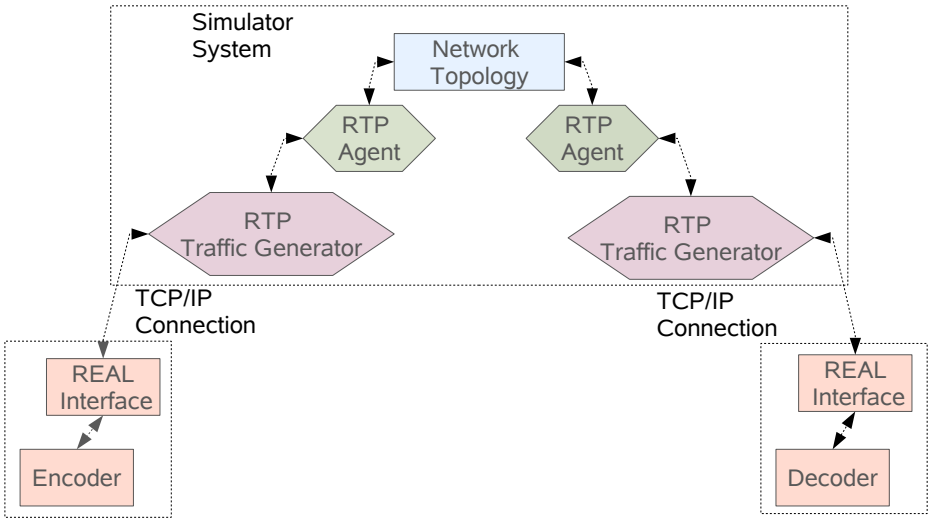


Figure 22: Simulator System Overview

ing delay. The REAL interface also maintains a sender buffer for retransmitting packets and a de-jitter buffer at the receiver. Additionally, it also implements the communication interface with the simulator for sending/receiving RTP packet, control messages, and synchronizing the clock with the simulator.

The *RTP Traffic Generator module* is the entry and exit point for the packets from and to the codec. This enables it to make rate-adaptation decisions. At the sender, it generates the control messages for the encoder based on the congestion cues.

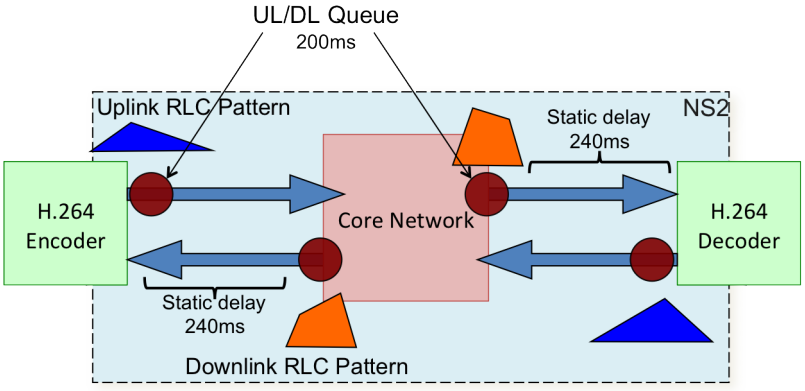
The *RTP Agent module* is responsible for scheduling and transmitting the RTCP reports. RTP Agent conforms to the RTCP timing rules described in [24, 30] and supports the following RTCP extensions: RTCP-XR [31], Transport layer feedback messages [30], Payload-specific feedback messages [30], Codec control messages [33], NADU [39].

The *Network topology module* simulates the behavior of the links in a specific multimedia environment. The details of each multimedia environment are described later in the chapter.

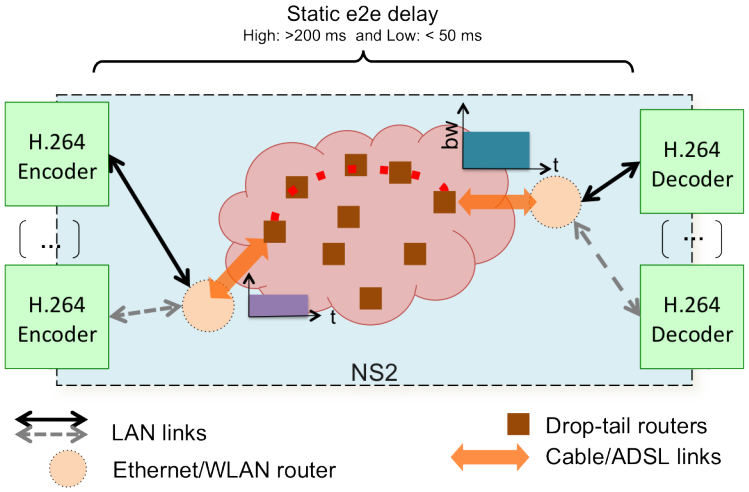
5.2 Simulation Settings

A typical conversational mobile multimedia system, such as MTSI, requires that capture-to-display delay does not exceed 400 ms [25] to provide acceptable media quality and a good user experience. One should also note that the 400ms delay includes the decoding and rendering delay. Therefore, we presume 400ms as the upper bound ($delay_{max}$), even for the heterogeneous and fixed Internet scenarios. Additionally, a suitable lower-bound is assumed to be 200ms; this should provide the opportunity for the decoder to cache at most 3 frames of a 15 FPS video ($\frac{1000}{15} \times 3 = 200ms$). So, packets arriving later than this are discarded upon receiving. We use the same upper bound ($delay_{max} = 400ms$) for all the multimedia environments.

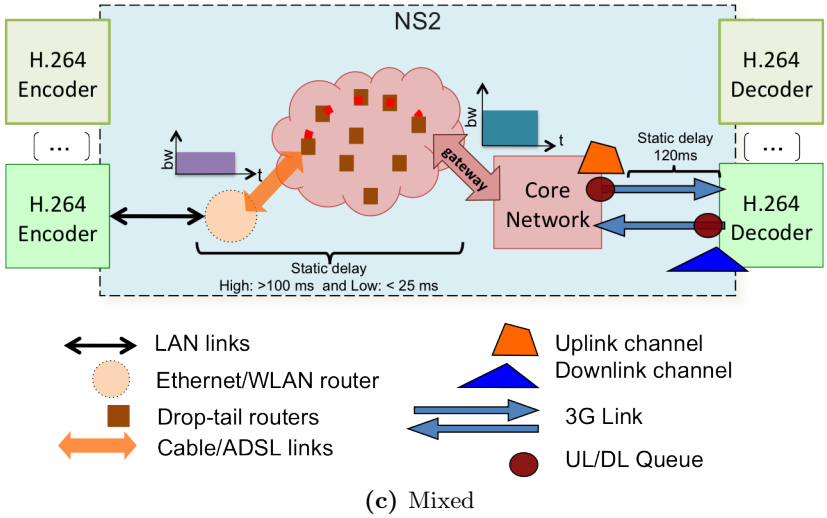
We use a medium motion media sequence ("Foreman" QCIF sequence) encoded at $15FPS$



(a) Mobile



(b) Wired



(c) Mixed

Figure 23: shows the (a) 3G, b) wired Internet (c) mixed simulation environment

and for simplicity, the sender encapsulates $1 frame/packet$ [12]. Furthermore, in all the

[12]Even though the H.264 codec [43] supports smart slicing of frames.

scenarios, the sender begins with an initial sending rate of 128 *kbps*[13] and the encoder is not restricted by a maximum encoding rate. As discussed earlier, video communication does not allow sufficient buffering (in time) to make use of B-frames[14]. Therefore, the encoder is configured to only produce I- and P-frames. To quickly overcome the bit-error losses in wireless networks, conversational video communication uses short Group of Pictures (GOP), we use $N = 4$, i.e., *"I P P P I"*, for a 15 FPS video would create 3-4 I-Frames in a 1 second interval.

5.3 3G Scenario

We investigate the congestion cues in the 3G environment and develop a rate-adaptation scheme for conversational video communication. Figure 23a illustrates the 3G scenario. Each pair of 3G links represents the uplink and downlink for the user. We presume that the 3G core network is a well provisioned error-free network. The 3G links conform to the behavior described in [45]. The Radio Link Control (RLC) [46] controls the scheduling and the amount of data (inclusive of the headers) that flows on the 3G link. The RLC is configured to unacknowledged mode to keep link layer delays to a minimum.

The 3G RLC frame sizes and scheduling are based on real-world traces from different scenarios [25]. For example, the sender's uplink is a concatenated pattern based on "Excellent", "Poor", and "Elevator" call scenarios (60s each) [47], while the receiver uses the elevator RLC pattern file concatenated three times. There are four different RLC pattern files, one for each 3G link – i.e., two for the sender side: uplink (UL) / downlink (DL); and two for the receiver side: uplink/downlink. Figure 24 shows the variation in uplink and downlink capacity of an upstream 3G channel over 5 minutes duration.

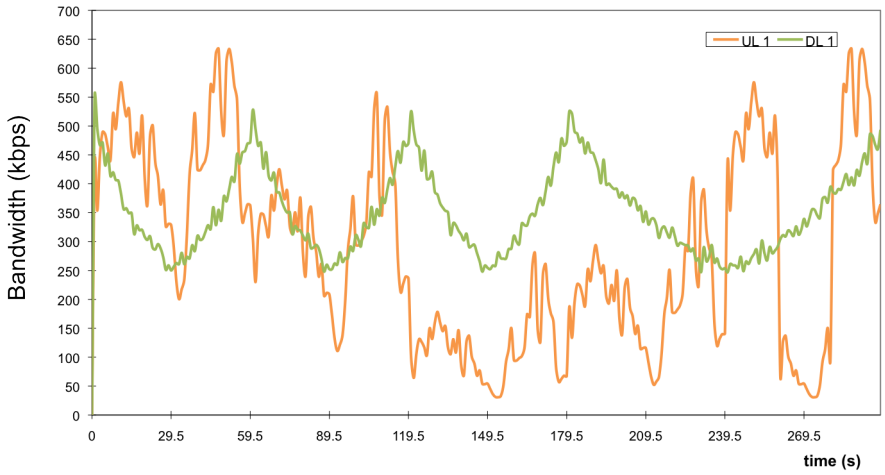


Figure 24: Mobile network's dynamic channel capacity

The simulation environment can also produce 0.5% to 1.5% link layer losses (3G Link) using error patterns defined in [48]. To simulate the 0.5% losses, the RLC frames [25] are further broken down into 40-byte frames and sent over the 3G link. If a 40-byte frame is dropped, reconstruction of the associated IP packets fails, therefore, a 0.5% loss rate may

[13] starting rate is an open problem, and not tackled in this research.
[14] Prediction based on future packets.

cause higher IP layer packet loss [32]. It should also be noted that no header compression is used over the 3G links.

The uplink and downlink queues in the network are long queues with $200ms$ time-to-live for a packet in the queue. Apart from the queuing delay caused by the RLC scheduling of each packet at the UL/DL queues, the packets are queued for a further $240ms$ as static one-way delay (OWD) just before they are delivered to the receiver. This simulates the link delay at the uplink and downlink wireless channel.

5.4 Internet Scenario

We investigate the congestion cues in the Internet scenario, and if needed, we then tweak the rate adaptation scheme developed for the mobile environment for the wired environment.

The modern Internet is very complex with millions of nodes, many types of applications and different types of application behaviors and traffic generations. S. Floyd et. al in [49] states:

> "Simple scenarios that illustrate the underlying principles are the *best*. However, when a scenario is scaled up based on real-life topology or traffic generation, and if the simulation still holds then it is *good*. Although, one should pay attention to the *choices* one makes in picking the underlying models that need to be explored. One of the biggest issues with simulating the Internet is to verify the simulator implements *exactly* what was intended. Furthermore, a small change in the model could affect or have a big effect on the final result. However, using a third party implementation might help in limiting this risk. Finally, [49] leaves it open to discussion on how to simulate the Internet based on heterogeneity, size (or scale), and unanticipated change."

Based on the above recommendations, we chose to keep our simulation environment really simple. Figure 23b illustrates a dumbbell configuration wherein, senders and receivers share the same bottleneck link with other users. The bottleneck link has a capacity of 1Mbps and the fan-out link to each receiver and sender is 10Mbps. The delay on the bottleneck link is 20ms for the low delay simulation and 160ms for the high delay simulation. The fanout links have a 5-20ms delay. The link delay and capacity are stable in the short term but may vary over longer timescales. Figure 25 shows an example of variation in channel capacity for an end-user over a 24 hour duration.

In a wired environment, there are possibly lower bit-error losses; on average, 1 in a million bits get corrupted due to bit-errors. However, queuing drops are a common cause of lost packets on the Internet. Exceeding the link capacity or the combined traffic from other applications or users may cause the router queue to overflow. Therefore, the routers in the simulator are configured as drop-tail routers with a capacity of 200 packets each.

5.5 Heterogeneous Scenario

Nowadays, calls originate on the Internet and may terminate on a mobile phone or vice versa; this simulation scenario attempts to emulate the real world. We combine the knowledge of the applicability of the congestion cues from the above two scenarios. If needed, we

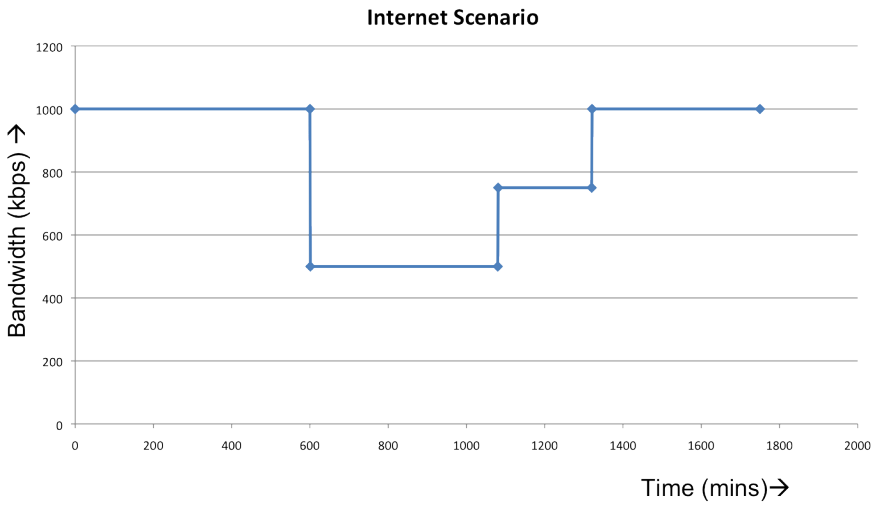


Figure 25: Wired network's stable channel capacity

also tweak the congestion control algorithm to perform optimally in this mixed network scenario.

This is a combination of the previous two simulation setups. Herein, two or more senders share a bottleneck link while the receivers are on independent 3G links undergoing different link conditions. The limiting link changes between the fairness on the bottleneck link to one of or both of the 3G links. Figure 23c shows a simplified illustration of this scenario. The simulation characteristics of each wired and 3G link follows the characteristics described in the previous sections.

6 Algorithms and Signaling

In this chapter, we introduce the algorithms and signaling extensions we developed for rate-control in video communication.

6.1 Signaling Receiver Rate vs Estimating it at the Sender

Section 3.7.2 introduced the basics of receiver rate and goodput. Unfortunately, the receiver measures the receiver rate/goodput and these either need to be explicitly signaled to the sender or a sender has to be able to estimate it. [50] and [37] explicitly signal this information. However, the receiver rate can also be estimated by using loss and discarded packet statistics.

To estimate the rate at the sender, the sender maintains a ring buffer with the size of all video packets sent since the last received RR. Additionally, the sender is aware of the fraction loss rate but it is unaware of which packets were lost [24]. The sender can estimate the receiver rate (RR_{est}) using the following equation:

$$RR_{est}(in\ kbps) = \frac{\sum_{i=HSN_{last\_RR}}^{HSN} sizeof(i)_{bytes} \times (1.0 - FL) \times 8}{1000 \times (t_{now} - t_{last\_RR})}$$

$$\forall \begin{cases} t_{now} & = current\ time \\ t_{last\_RR} & = reception - time\ of\ last\ RR \\ FL & = Fractional - loss\ reported\ in\ the\ RR \\ HSN & = Highest\ Sequence\ Number\ reported\ in\ the\ RR \\ HSN_{last\_RR} & = Highest\ Sequence\ Number\ reported\ in\ the\ last\ RR \end{cases}$$

The above equation gives a rough estimate of the receiver rate because the sender does not know which packets got lost in the interval and instead, has to approximate it using the fractional packet loss. For example, if two out of 10 packets are not received at the receiver, the interval fraction loss rate is 20%, but if these 2 packets were larger than the rest, then the fraction loss in bytes would be larger than 20% and thus the estimated receiver rate would be higher than the actual receiver rate.

Estimating Receiver Rate using additional information from the RTCP-XR [31]

Using the run-length encoding (RLE) defined in RTCP-XR [31], the receiver can signal exactly which packet sequence numbers were lost. This not only helps the sender to exactly calculate the bytes lost in the interval but also provides the interval loss pattern. This loss pattern helps the sender to gauge if the losses occurred at the beginning or at the end of the reporting interval. Additionally, it reports if the losses were bursty or randomly occurring. The sender can calculate the precise RR_{est} using the following equation:

$$bytes_{lost} = \sum_{i=HSN_{last\_RR}}^{HSN} sizeof(i)_{bytes}$$
$$\forall \quad i \in lost, \qquad \text{(signaled in RFC3611)}$$

$$RR_{est}(in\ kbps) = \frac{(\sum_{i=HSN_{last\_RR}}^{HSN} sizeof(i)_{bytes} - bytes_{lost}) \times 8}{1000 \times (t_{now} - t_{last\_RR})}$$

Discard Rate

Alternatively, in video communication, goodput is a better metric for measurement (and indicator for congestion) than receiver rate, because goodput excludes the packets discarded at the receiver[15] from the calculation. We defined an extension at the IETF titled, *"Real-time Transport Control Protocol Extension Report for Run Length Encoding of Discarded Packets"* (draft-discard-packets) [51]. The RTCP extension reports the packets discarded due to early and late arrival. It proposes two methods for reporting the discarded packets; they are:

1. The total number of bytes discarded at the receiver in the last reporting interval or in the whole session. This is signaled using the format shown in Figure 26a.
2. Run-length encoding for discarded packets. This method is similar to the RLE coding for lost packets. Figure 26b shows the protocol format to signal RLE discarded packets.

$$bytes_{discarded} = \sum_{i=HSN_{last\_RR}}^{HSN} sizeof(i)_{bytes}$$
$$\forall i\ satisfies: \quad delay_i = TS_R[i] - TS_S[i] > delay_{max}$$
$$i.e.,\ i \in discarded, \qquad \text{(signaled in draft-discard-packets)}$$

Therefore, by updating the previous equations with the new information, we get:

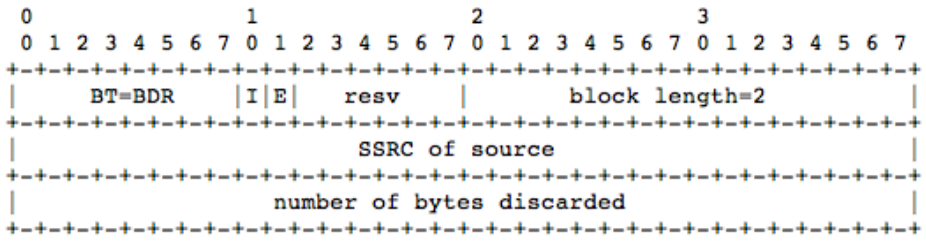
$$Goodput_{est}(in\ kbps) = \frac{(\sum_{i=HSN_{last\_RR}}^{HSN} sizeof(i)_{bytes} - bytes_{lost} - bytes_{discarded}) \times 8}{1000 \times (t_{now} - t_{last\_RR})}$$
(using RFC3550, RFC3611 and draft-discard-packets)

Additionally, by combining the RLE formats described in [51] and [31] the sender can recreate the congestion pattern in the last reporting interval.

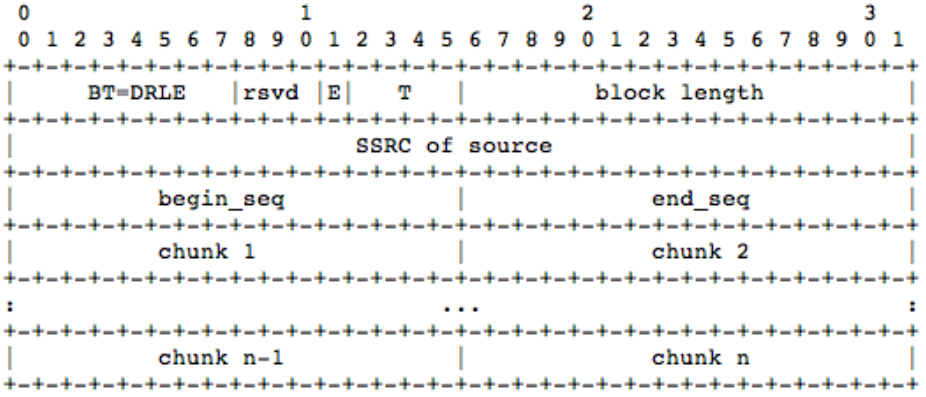
An example based on a simulation shows the correlation between the actual receiver rate and the estimated goodput at each RTCP interval. The correlation is calculated using the following equation:

$$corr. = \frac{ActualReceiverRate}{Goodput_{est}(in\ kbps)}.$$

[15] due to late arrival ($delay_{max}$)



(a) BDR



(b) DRLE

Figure 26: RTCP header extension for signaling the (a) bytes discarded per RTCP interval during a multimedia session, (b) discarded packet pattern at the receiver

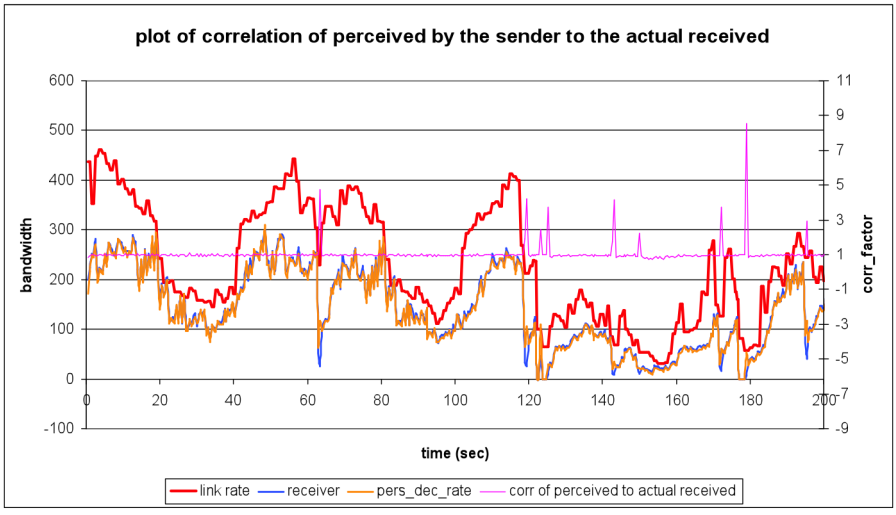


Figure 27: Goodput estimated at the sender vs Actual Goodput (does not include discarded packets)

Figure 27 shows that the estimated goodput matches the signaled receiver rate most of the time because the correlation is 1.0 except for a few instances. Since the receiver rate also includes the discarded packets, the signaled receiver rate is higher than the estimated goodput. Therefore, by using the lost and discarded packet information the sender can

estimate receiver goodput and discover the congestion and loss pattern.

6.2 Controlling the RTCP Reporting Interval

The end-to-end channel capacity in a 3G or heterogeneous network can change dramatically at any moment in time, therefore any over-utilization by the sender will cause congestion. Furthermore, the longer it takes the sender to adapt its sending rate, the longer it will take the congestion to mitigate. However, the reporting interval at an end-point is limited to 5% of the media bit rate. Therefore, to have a shorter reporting interval the media rate should be higher. For example, 200ms reporting interval would require a media bit rate of 80kbps, presuming a 100 byte RTCP packet. We also do not recommend sending feedback quicker than RTT, as any rate control decision[16] will take one RTT time to take effect unless it wishes to update that decision. Therefore, the reporting interval can neither be too short nor too long.

RFC4585 [30] allows the RTCP feedback rate to be 2.5% of the media rate for each end-point in a point-to-point scenario, which is quicker than the $5 \pm 2.5s$ restriction described in [24]. To quickly adapt to congestion, [50] sends RTCP feedback packets every *200ms-380ms*, but uses non-compound RTCP [52] to conserve RTCP bandwidth. We use compound RTCP [52] reporting because normal RTCP packets report essential congestion cues such as RTT, jitter, fractional loss, etc. to the sender.

In algorithm 1, we describe a reactive mechanism to change the RTCP feedback timing based on the bad packet rate (Γ_{bpr}) that takes into account both lost and discarded packets in an RTCP interval for throttling RTCP feedback. We limit the RTCP RR interval to the lower bound set by the timing rules of [30], and if available, the reported RTT [37].

The algorithm records the number of bad packets in an interval; conversational video communication is tolerant to occasional packet loss because the group of pictures is small. In Section 5.2 we assumed GOP=4 and 15 FPS, i.e., the impact of a loss of a packet or frame is limited to 4 frames [17], which is approximately $\approx 350ms$. However, over-utilization of a link causes more bursty losses and therefore impacts more frames and a longer interruption in video rendering. Therefore, to provide an immediate feedback, we assume $\Gamma_{bpr} = 0.3$ or 30%, i.e., for an average reporting interval of 1 second, 5 bad packets[18] will cause the reporting interval to halve.

6.3 Inter-arrival Time (IAT) instead of Jitter

For congestion control purposes the jitter value is not expected to be useful as an absolute value. It is more useful as a means of comparing the reception quality at two instances of time or between two receivers [53]. In RFC3550 [24], the jitter is computed by averaging over 16 packets and the RR is sent on average every $5s$, i.e., for a 15 FPS video, at least 75 packets would be sent. In this case, the jitter value captures a fraction of the reporting interval. However, in video communication, the RTCP RRs may be sent quicker than 1 second and, because of rate adaptation, the jitter experienced by the packets at the beginning of a 1 second interval may be different from the packets at the end of the

[16] This behavior is compatible with TCP rate control algorithms.
[17] Loss of an I-frame is the worst case.
[18] We assumed 1 Frame/packet

Algorithm 1 Algorithm for adapting the of RTCP RR Interval

Require: RTCP RR timeout
Ensure: Compliance to RTCP Timing as defined in [30].

Fetch $Pkt\_Count_{discarded}$ for the current interval
Fetch Pkt_Count_{lost} for the current interval
Fetch $Pkt\_Count_{received}$ for the current interval

Calculate $bad\_packet\_rate = \frac{Pkt\_Count_{received}}{Pkt\_Count_{received}+Pkt\_Count_{discarded}+Pkt\_Count_{lost}}$

Set $rtcp\_interval_{new} \leftarrow rtcp\_interval_{prev}$
$//initial\_value \in (500ms, 2000ms]$, we use $initial\_value = 1000ms$

// calculate minimum rtcp interval based on RFC4585
$min\_rtcp\_intvl = Min4585RTCPInterval()$

// RTT may be signaled using [37]
if (RTT) **then**
  $min\_rtcp\_intvl = max(min\_rtcp\_intvl, RTT)$
end if

// we use, $\Gamma_{bpr} = 30\%$
if $(bad\_packet\_rate > \Gamma_{bpr})$ **then**
  set $rtcp\_interval_{new} = \frac{rtcp\_interval_{prev}}{2}$
  if $(rtcp\_interval_{new} < min\_rtcp\_intvl)$ **then**
    set $rtcp\_interval_{new} = min\_rtcp\_intvl$
  end if
else if $(bad\_packet\_rate == 0\%)$ **then**
  set $rtcp\_interval_{new} = min(rtcp\_interval_{prev} \times \frac{3}{2}, 2000ms)$
end if

$//max_{delay} = 400$ms
if $(time\_since\_last\_packet > \frac{max_{delay}}{2})$ **then**
  set $rtcp\_interval_{new} = min\_rtcp\_intvl$
end if

set $rtcp\_interval_{prev} = rtcp\_interval_{new}$

interval. Especially if we presume the same 15 FPS video and 1 frame/packet, the jitter calculation (see Section 4.3.2) will cover multiple RTCP intervals. Therefore, for a highly dynamic bit rate and short RTCP reporting interval, the jitter value is inconsistent with the reporting interval's average inter-arrival time.

Instead, we use per packet inter-arrival time for receiver-side rate adaptation. The inter-arrival difference can be calculated using the reception timestamp (TS_R) and the sending timestamp (TS_S) of two successively received RTP packet:

$$TS_R[n] - TS_R[n-1] < TS_S[n] - TS_S[n-1]$$
$$\implies packet - jitter\ negative,\ congestion\ mitigating$$
$$TS_R[n] - TS_R[n-1] > TS_S[n] - TS_S[n-1]$$
$$\implies packet - jitter\ positive,\ congestion\ starting$$

The inter-arrival time calculation is used for making rate-control decisions in TMMBR-U (see Section 6.7).

6.4 Playout delay and Time Since Receiving Last Packet, (ΔT_{HSN})

The sender transmits a real-time media stream and each packet number includes a sequence number and timestamps so that the receiver knows the time base for rendering each packet. The sender also keeps a ring buffer with the aforementioned sent packet information between two receiver reports.

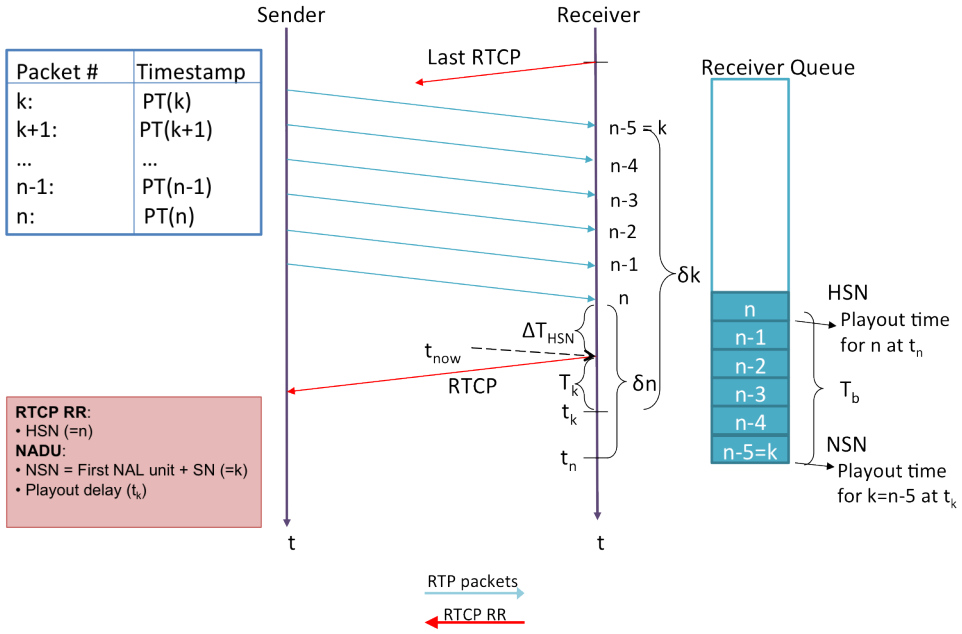


Figure 28: Time Since Receiving Last Packet (ΔT_{HSN})

From the received NADU [39] packets, the sender learns about the receiver state:

1. which packet is the first one in the receiver buffer, also known as Next Sequence Number (packet n-5=k in Figure 28);

2. which NAL unit inside this packet is going to be rendered next; and
3. when the specific NAL unit will be rendered.

This information allows the sender to reconstruct the contents of the receiver buffer in the absence of packet losses: it knows the first packet's sequence number (NSN) from NADU and the highest sequence number (HSN) from the RR packet. Thus, the sender knows which packets are currently buffered.

$$Packets\ in\ Buffer,\ PiB = \quad (HSN - NSN) + 1\ packets$$

Furthermore, from its local table (see the left hand side in Figure 28), the sender can determine the duration of real-time media by computing the difference in timestamps of the first packet (NSN) and the last packet (HSN). This yields the playout time of all the media contained in the buffer T_b.

$$T_b = BFL_{in\_ms} = \quad TS_S[HSN] - TS_S[NSN]\ s$$

But this is insufficient to determine the total playout delay of the last received packet; this value is of interest because it defines when a decoder under-run will occur and impact the perceived media quality.

For calculating this value, the period between the reception of the last packet (HSN) in the buffer "n" (received at $t_r(n)$) and its playout point needs to be determined: $\delta_n = t_n - t_r(n)$. It is known how long the rendering of the entire buffer contents will take: T_b. From NADU, it is also known how long it will take until the next (piece of the) first packet in the queue will be handed to the decoder for rendering and how long it will take to render; i.e., the playout delay: T_k. The playout delay for the last packet received is thus $T_n > T_k + T_b$ or $T_n = T_k + T_b + \Delta t_{HSN}$.

Δt_{HSN} represents the time between the reception of packet "n" at the receiver and the generation of the RTCP receiver report. Depending on the latency of the network, the rate at which media packets are generated and the jitter they experience inside the network, this value may be significant or, at least, non-negligible.

This can be mathematically summarized as follows:

$$\begin{aligned} delay_{decoding} = &\ time\ spent\ in\ the\ decoder's\ queue \\ = &\ Time_{render}[k] - Time_{decoder}[k] \\ = &\ order\ of\ a\ few\ to\ tens\ of\ ms. \end{aligned}$$

$$\begin{aligned} PD_k = &\ Time_{render}[k] - Time_{now} \\ = &\ (decoding - delay) + Time_{decoder}[k] - Time_{now} \end{aligned}$$

$$\begin{aligned} PD_{NSN} = &\ Time_{render}[NSN] - Time_{now} \\ PD_{HSN} = &\ Time_{render}[HSN] - Time_{now} \\ = &\ (PD_{NSN} + BFL_{in\_ms}) \end{aligned}$$

$$
\begin{aligned}
delay_{HSN} \Longrightarrow & \\
&= Time_{render}[HSN] - TS_S[HSN] \\
&= PD_{HSN} + OWD + \underbrace{Time_{now} - TS_R[HSN]}_{} \\
&= \overbrace{PD_{NSN} + BFL_{in\_ms}} + OWD + \Delta T_{HSN} \\
delay_{HSN} &= PD_{NSN} + BFL_{in\_ms} + \frac{RTT}{2} + \Delta T_{HSN} \\
&< delay_{max} \\
delay_{HSN} \quad should\ be \quad &< delay_{max}
\end{aligned}
$$

To summarize, variation in bandwidth affects both buffer fill-level and RTT. The total delay experienced by an HSN packet ($delay_{HSN}$) is a function of both RTT and buffer fill-level, variation in bandwidth indicates congestion and underflow. We conclude that this is an important parameter for estimating the underflow point and should be signaled in addition to the NADU. We will denote the algorithm and signaling scheme as C-NADU, which stands for Conversational NADU [54]. Figure 29 presents the updated RTCP application header for C-NADU.

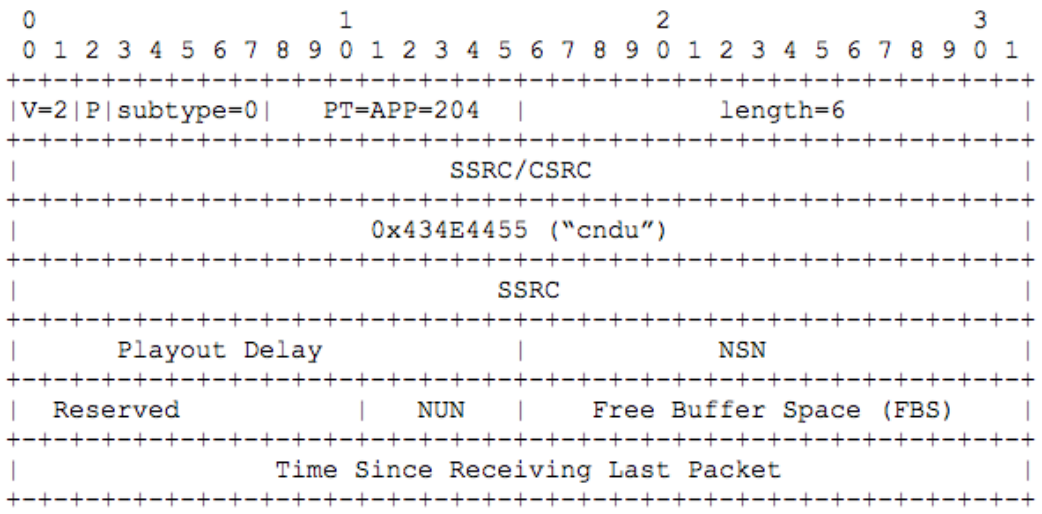


Figure 29: RTCP header extension for C-NADU

6.5 Sender Driven Rate Adaptation: Conversational NADU (C-NADU)

C-NADU signaling is a sender-side rate adaptation algorithm for conversational video that borrows its basic elements from NADU, which is defined for a Packet-switched Streaming Service (PSS) in 3GPP [39]. C-NADU attempts to recreate the receiver side buffer at the sender based on the feedback information. The operating model is shown in Figure 30. The new bit rate is calculated based on input from many parameters signaled in various RTCP extensions. The following RTCP header extensions are used: [24, 30, 31, 33, 39, 51]

- Normal RTCP Receiver Report (RR) [24].
  - Fraction Loss (FL)

  - Inter-arrival Jitter ($Jitter$)
  - Calculated RTT (RTT)
  - Highest Sequence Number, (HSN)
- NADU Packet [39] reports
  - Next Sequence Number (NSN) is the RTP sequence number of the next packet to be decoded from the receiver queue. If no packets are available for playout then, $NSN = HSN + 1$ (this packet has not been received by the receiver yet).
  - Playout Delay of NSN (PD_{NSN}) is the difference between the scheduled playout time of the NSN packet and the time the receiver sends the RTCP report [39]. If no packets are available for playout then the receiver can signal $PD_{NSN} = 0 \times FFFF$.
- RTCP XR Loss Metrics [31] packet reports
  - the packets that were not received at the receiver.
- RTCP XR Discard Metrics [51] packet reports
  - the packets that were discarded at the receiver due to late arrival of packets.

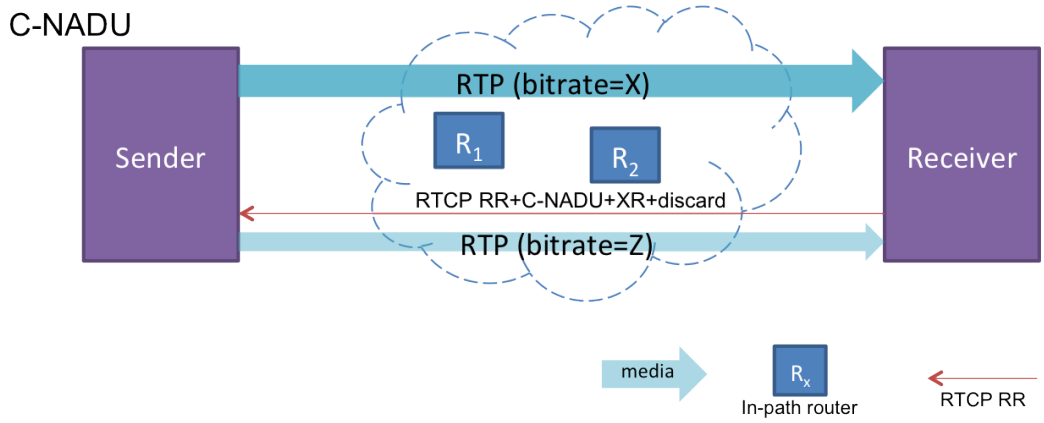


Figure 30: Operating model for C-NADU

6.5.1 Sender-side algorithm

In addition to the above signaling information, the sender maintains a ring buffer with the size of all video packets sent since the last RR received. The sender also keeps a short history of some of the above parameters, namely PiT, PiB, Jitter, and RTT, by calculating the correlation of the current value with a moving average of the last 3 values or the 90^{th}-*percentile* (90P) values of lossless reports.

1. Correlated RTT, by using the 90^{th}-*percentile* value of all loss-less RTTs it is possible to calculate the correlation of the current RTT,
   $CorrRTT = \frac{90P_{lossless}(RTT)}{RTT_{now}}$
2. Correlated PiT and PiB are calculated to ascertain if the queues in the network and at the receiver are increasing or decreasing.
   $CorrPiT = \frac{PiT_{avg\_last3}}{PiT_{now}}$ and $CorrPiB = \frac{PiB_{avg\_last3}}{PiB_{now}}$

Algorithm 2 Sender-side Rate Adaptation Algorithm

Require: Encoder maintains a ring-buffer with sizes of packets sent since the HSN of Last RR
Ensure: Reception of Latest RR from receiver
Parse $(RR) \Rightarrow (RTT_{now}, Jitter, FL, HSN_{now})$
if available, Parse $(NADU) \Rightarrow (NSN, PD_{NSN})$
if available, Parse *(RTCP XR Discard Metric)* $\Rightarrow (bytes_{discarded})$
Calculate $PiB_{now}, PiT_{now}, CorrRTT, CorrPiT, CorrPiB$
and $RR_{est}, Goodput_{est}, PD_{HSN}$
if $(HSN_{now} = HSN_{last\_RR})$ **then**
  //No Packets were received!
  $NewBw \leftarrow CurrentBw \times \alpha$;
  $\forall\ \alpha \in (0, 1)$, we use $\alpha = 0.5$
else
  if $((FL > 0) \vee (bytes_{discarded} > 0))$ **then**
    //Congestion mitigation!
    if $(CurrentBw > Goodput_{est}$ **then**
      $NewBw \leftarrow Goodput_{est} \times \delta_{undershoot}$
      $\forall \delta_{undershoot} \in (0, 1]$
    else
      //High congestion!
      if $(CorrRTT < 1.0)$ **then**
        $NewBw \leftarrow CurrentBw \times CorrRTT$
      else
        $NewBw \leftarrow CurrentBw \times \beta$
        $\forall\ \beta \in (0, 1)$, we use $\beta = \frac{\sqrt{2}}{2}$
      end if
    end if
  else
    //Congestion Avoidance!
    if $(CorrPiT < 1.0)$ **then**
      $NewBw \leftarrow CurrentBw \times CorrPiT$
    else if $(CorrPiB < 1.0)$ **then**
      $NewBw \leftarrow CurrentBw \times CorrPiB$
    else if $((CorrPiT > 1.0) \wedge (CorrPiB > 1.0))$ **then**
      $NewBw \leftarrow CurrentBw \times corrPiT$
    end if
    if $(PD_{HSN} \neq 0 \times FFFF)$ **then**
      $NewBw \leftarrow CurrentBw \times \frac{delay_{max}}{PD_{HSN}}$
      $\forall\ PD_{max} = 400ms$
    else
      //Underflow!
      if $(CurrentBw < RR_{est})$ **then**
        $NewBw \leftarrow RR_{est}$
      else if $(CorrRTT > 1.0)$ **then**
        $NewBw \leftarrow CurrentBw \times CorrRTT$
      else
        $NewBw \leftarrow CurrentBw \times \Psi$;
        $\forall\ \Psi \in (1, 2)$, we use $\Psi = 1.1$
      end if
    end if
  end if
end if

When making rate control decisions, the algorithm takes the following into account:

- If any packets were received in the last reporting interval, or
- If any packets were lost or discarded, or
- Change in congestion cues based on past reports.

When the receiver reports that no packets were received during the last interval, the algorithm assumes severe congestion along the path and halves the bit rate ($\alpha = 0.5$). In the case where the bad packet rate is known, the algorithm estimates the receiver's goodput and undershoots ($\delta_{undershoot}$) to compensate for the network congestion it has caused. In the scenario when neither packets are discarded nor lost at the receiver, the algorithm calculates the new encoding bit rate based on the receiver's playout buffer (PD_{HSN}, see Section 6.4). The algorithm also uses the variation in RTT over the last few intervals to tweak the calculated encoding bit rate to avoid underflow. Underflow occurs when the receiver consumes data more quickly than the sender sends the data, in this case, the algorithm increases the encoding rate by probing a higher bit rate (Ψ).

In Algorithm 2, line 15, $\delta_{undershoot}$ is calculated only for the first loss event of a new downward trend, and is done to quickly mitigate congestion because of higher rate packets in transit, and *lines* 9, 22, *and* 45 use constants (α, β, Ψ) to reduce and increase the bandwidth, when no conclusive cues are available to determine the path characteristics[19].

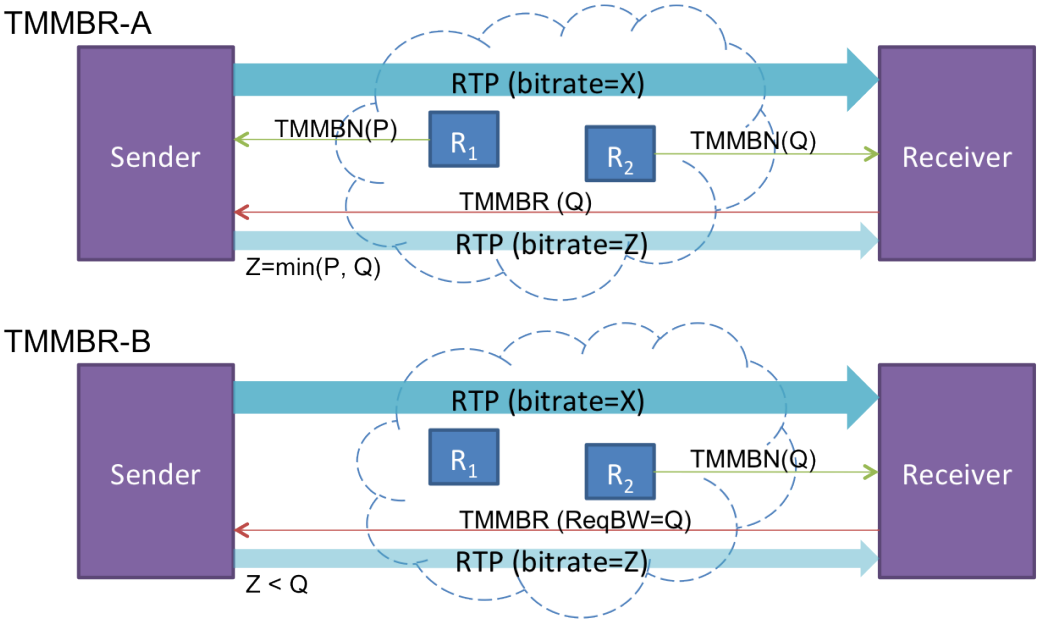


Figure 31: Operating model for TMMBR-A, and TMMBR-B

6.6 Network Assisted Rate Adaptation: TMMBR-A and TMMBR-B

In TMMBR-A, the network notifies the sender and receiver of the Uplink and Downlink rates, respectively. The sender is now aware of the downlink capacity, but this information arrives at the sender with a delay in the order of a one-way delay from the receiver. Since

[19] This may happen in cases of extreme congestion or underflow.

the downlink may not be the constraining link, the sender also receives information about the Uplink rate. This setup reflects an ideal scenario, as the routers of the bottleneck link will signal the available link rate to its peers. Therefore, the sender chooses the lowest of the reported bit rates as its new encoding rate. Figure 31 shows the operating model of TMMBR-A.

Using TMMBR-B, the network notifies the receiver of the Downlink rate. As before, the sender is notified about the current downlink capacity by the receiver; however the sender is not aware of the Uplink rate. Hence, the TMMBR messages from the receiver are considered as an upper bound for the current encoding rate and the bit rate requested in TMMBR is never exceeded. Figure 31 shows the operating model of TMMBR-B.

The receiver also enhances the performance of TMMBR in all scenarios (TMMBR-A, and TMMBR-B) by signaling the number of discarded bytes [51] to the sender. This information helps in undershooting and thus temporarily alleviating the stress on the network queues.

6.7 Receiver Driven Rate Adaptation: Unassisted TMMBR (TMMBR-U)

Unlike the network-assisted TMMBR-A and TMMBR-B mechanisms, TMMBR-U is not assisted by the network. TMMBR-U is a receiver-driven rate adaptation scheme, wherein the receiver signals the new encoding rate to the sender and the sender uses the new bit rate to encode the media. TMMBR-U uses the packet inter-arrival time (IAT) to make rate-control decisions. Figure 32 shows the operating model of TMMBR-U. The receiver calculates the new bit rate taking the current Goodput (GP), receiver rate (BR_R) and bad packet rate into account. The receiver then sends the new bandwidth request as a TMMBR request. The basic algorithm is described in Algorithm 3. The receiver calculates the new encoding bit rate based on the variation of the inter-packet arrival time in that interval and the expected inter-arrival time (see Section 6.3). Due to link-induced losses, the sender implements some congestion mitigation techniques based on increasing RTT to gradually vary the encoding bit rate. Typically, the encoding bit rate is similar to the one recommended by the receiver in the TMMBR message.

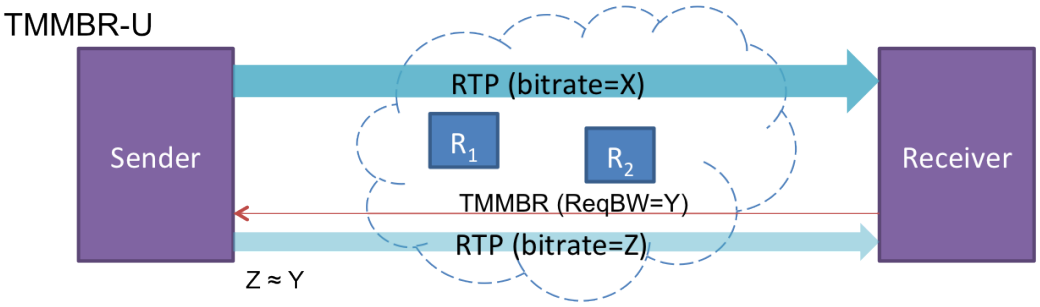


Figure 32: Operating model for TMMBR-U

Algorithm 3 Algorithm for adapting the of RTCP RR Interval

Require: RTCP RR timeout
Ensure: Compliance to RTCP Timing as defined in [30].
Fetch $Pkt\_Count_{discarded}$ for the current interval
Fetch Pkt_Count_{lost} for the current interval
Fetch $Pkt\_Count_{received}$ for the current interval

Calculate BR_R, receiver rate for the current interval
Calculate $Goodput(GP)$ based on the packets rendered in the current interval

//IAT is the average inter-arrival time for each received video frame
Fetch IAT_{avg} for the current interval

$//IAT_{exp}$ is the expected inter-arrival time for each frame,
$IAT_{exp} = \frac{1000}{15} = 66.66ms$ for a 15 FPS video call.
// can also be $90000/15 \approx 6000$

$\zeta_{IAT} = \frac{IAT_{exp}}{IAT_{avg}}$
if $((Pkt\_Count_{lost} > 0)||(Pkt\_Count_{discarded} > 0))$ **then**
    // $\zeta_{IAT} < 1.0$
    set $NewBW_{TMMBR} \leftarrow GP \times \zeta_{IAT}$
else
    // $\zeta_{IAT} > 1.0$
    set $NewBW_{TMMBR} \leftarrow BR_R \times \zeta_{IAT}$
end if

6.8 Summary

This chapter describes in detail the algorithms we developed for rate-control. We considered assisted and autonomously operating adaptation schemes such as, TFRC, TMMBR and C-NADU. We also considered sender-driven and receiver-driven rate adaptation algorithms. In the next chapter, we evaluate the performance of the algorithms in different multimedia environments.

7 Evaluation of Rate-control Algorithms

In this chapter, we evaluate the rate-adaptations schemes for Video and Voice over IP (VVoIP) calls in different multimedia environments. In subsections 7.1, 7.2, and 7.3, we evaluate rate-adaptation in a 3G environment, a wired environment and a Heterogeneous environment, respectively.

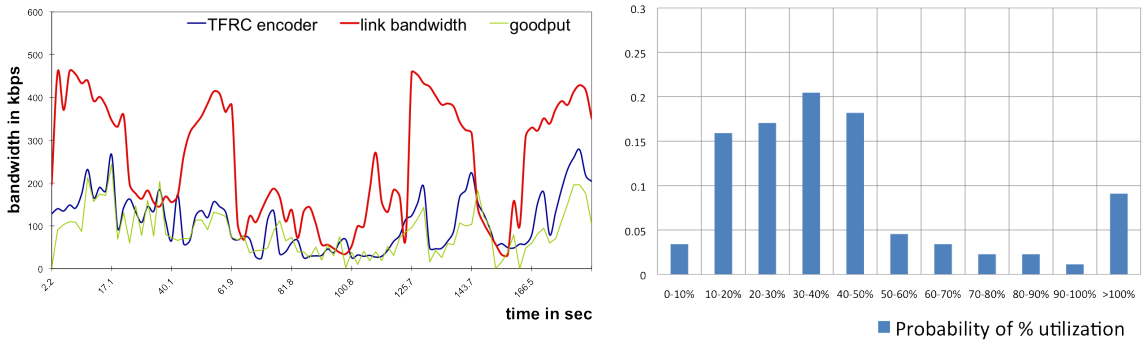
7.1 Rate control in 3G environments

3G environments are challenging because of the variable link capacity, i.e., the channel capacity changes quite often and it remains stable only for a few moments. In this subsection, we quantitatively compare rate-control algorithms: TFRC, C-NADU, TMMBR-A, TMMBR-B and TMMBR-U for conversational video in 3G environments.

TFRC is implemented as defined in the base specifications [36,38] and adapted to RTP/UDP based on [37]. All extensions were implemented as discussed in Section 4.5.1. TFRC-FB is sent along with each RR every $500ms$.

In TMMBR-A and TMMBR-B (see sec. 6.6), the network assists the sender, or receiver, or both. The bandwidth updates are generated at the end of every 1s interval (by averaging the RLC bytes available in that interval). Therefore, TMMBR feedback from the receiver is sent every $1s$.

In TMMBR-U (see sec. 6.7) there is no network assistance, and the receiver recommends a new encoder rate to the sender based on the receiver's bad packet rate and the variations in the inter-arrival time of media packets (refer to Algorithm 3).

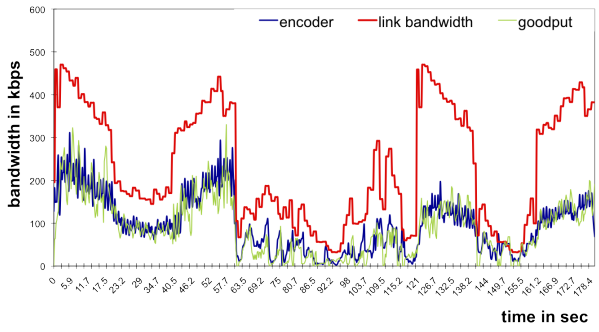


(a) TFRC **(b)** TFRC: ABU = 33%

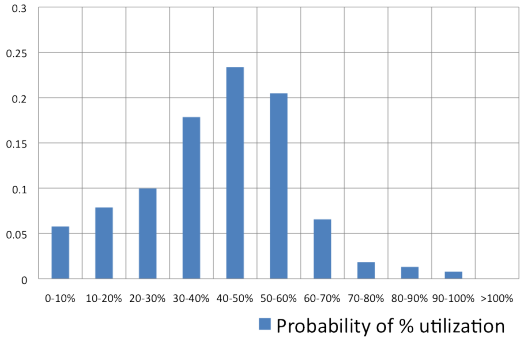
Figure 33: Plot of Link rate, encoder rate, goodput (left column) and Histogram of Probability of per-instance %Utilization (right column) and Average BW Utilization (ABU) of Dynamic 3G Links by TFRC

C-NADU uses the algorithm described in Section 6.5 and the signaling defined in [39]. The C-NADU feedback packet is sent along with every RTCP RR (see Section 6.4). However, the bytes discarded extension [51] is only sent by the receiver when it actually discards packets due to late arrival.

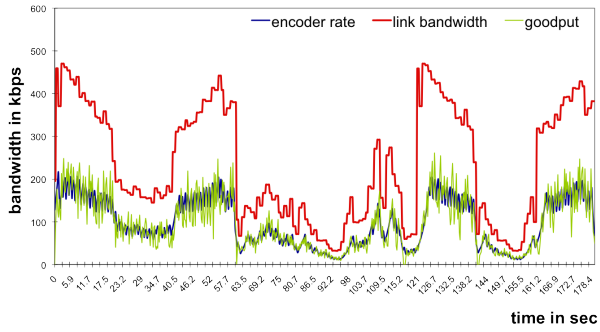
In the case of TMMBR-U and C-NADU, the receiver adapts the reporting interval based on the algorithm described in Section 6.2.



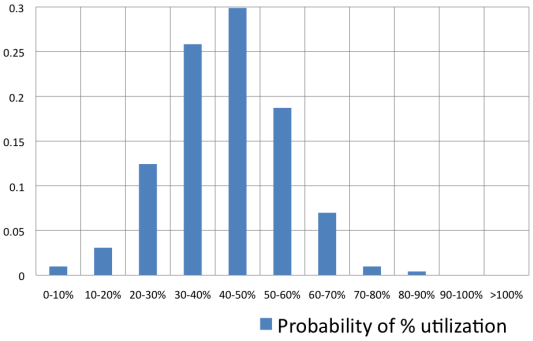
(a) TMMBR-U



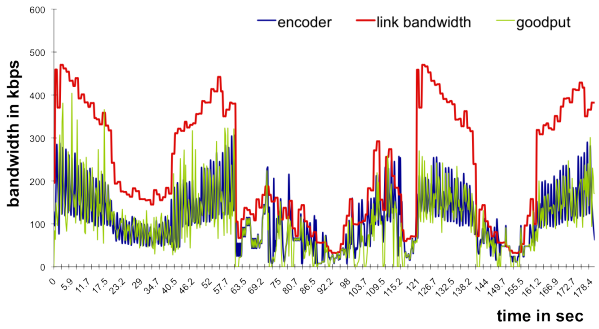
(b) TMMBR-U: ABU = 40%



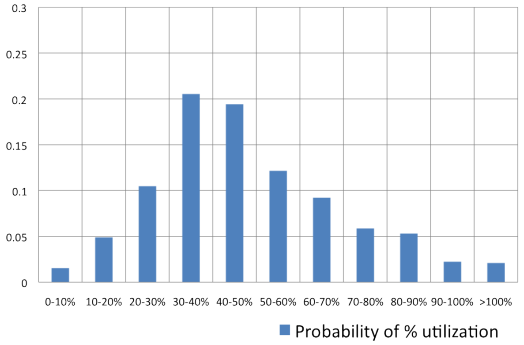
(c) TMMBR-A



(d) TMMBR-A: ABU = 60%



(e) TMMBR-B

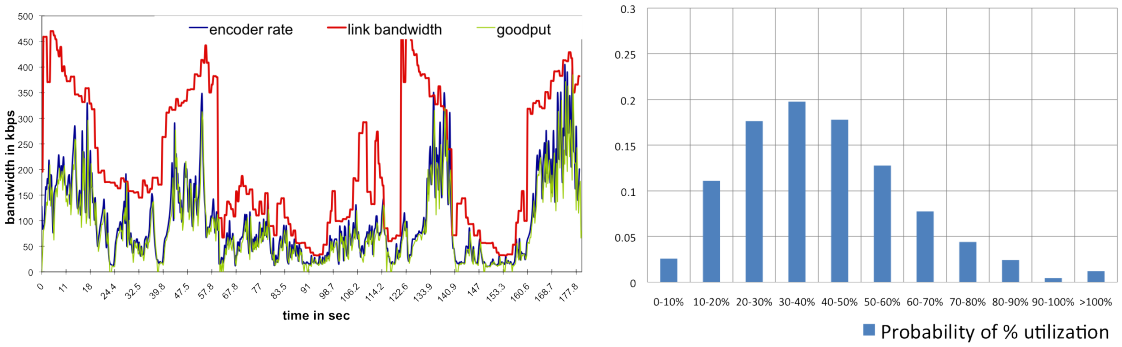


(f) TMMBR-B: ABU = 50%

Figure 34: Plot of Link rate, encoder rate, goodput (left column) and Histogram of Probability of per-instance %Utilization (right column) and Average BW Utilization (ABU) of Dynamic 3G Links by (a) TMMBR, (b)TMMBR-A, (c)TMMBR-B

7.1.1 Results

Figures 33, 34, and 35 (left column) show the instantaneous variation of the encoder rate and decoder goodput to the link bandwidth. Figures 33, 34, and 35 (right column) plots the percentage of bandwidth utilization. Table 2 presents the average encoder rate, average goodput, average PSNR and the delta loss rate (DLR) for the scenarios (metrics are defined in Section 3.9). It has to be pointed out that in our simulations, the air interface loss rate



(a) C-NADU **(b)** C-NADU: ABU = 55%

Figure 35: Plot of Link rate, encoder rate, goodput (left column) and Histogram of Probability of per-instance %Utilization (right column) and Average BW Utilization (ABU) of Dynamic 3G Links by C-NADU

in normal conditions was 1.9% for TFRC%, 1.8% for TMMBR-U, 1.9% for TMMBR-A, 2% for TMMBR-B and 1.8% for C-NADU in the dynamic 3G link scenarios.

Table 2: Scenario: Point-to-Point calls in a 3G Network

	Avg. enc. rate (kbps)	Avg. goodput (kbps)	DLR (%)	Avg. PSNR (dB)
TFRC	98.6	84.1	6.9%	29.3
TMMBR-U	99.7	89.8	3.7%	30.5
TMMBR-A	97.7	90.1	1.3%	32.3
TMMBR-B	98.5	90.5	2.9%	31.7
C-NADU	99.4	92	2.2%	31.9

TMMBR-A, due to its knowledge of the network conditions at the UL and DL, provides the best adaptation (1.3% delta loss rate (see Section 3.9.4) and 60% ABU (See section 3.9.2)), while TFRC, basing its knowledge solely on normal RRs, suffers from the maximum packet loss (6.9%) and underutilizes the link (33% ABU). TMMBR-B receives the upper-bound bandwidth information of the downlink, and is therefore able to provide better utilization (50%) of the link when compared to TFRC. However, due to probing (based on RTT, inter-arrival times of packets at the receiver), it causes a delta loss rate of 2.9%. C-NADU, on the other hand, without any assistance from the network, produces better results in terms of delta loss rate (2.2%) and ABU (55%) when compared to TFRC and unassisted TMMBR (TMMBR-U), which produces 3.7% delta loss rate and only 40% ABU. C-NADU has higher bandwidth utilization because it adapts based on the variation in the receiver's playback buffer information.

7.1.2 Summary

In the simulated scenario, network-assisted rate adaptation provides the best adaptation, which can be useful in scenarios such as handovers and cell-loading where the operator has knowledge of an event before it takes place. In this case, TMMBR-A (TMMBR with network-assisted adaptation) shows the best performance. When no direct information

about the uplink and downlink bit rates is available from the network, our new algorithm (C-NADU) shows a performance close to that of TMMBR. Results also show that TFRC is not well suited for multimedia applications because it underutilizes the link. We believe that C-NADU can be extended to operate in the general internet because it does not get link updates like TMMBR and makes decisions based on perceived network conditions.

7.2 Rate control in the Internet

The wired Internet links are different from 3G links because they have more stable capacity and significantly fewer bit-error losses. For a rate-control algorithm to be successful, it should be able to operate in this environment as well.

In the following section, we first quantitatively compare two rate adaptation algorithms in a wired environment, namely, TFRC and C-NADU. The purpose of the simulation setup is to determine which of these two algorithms maximizes utilization of a bottleneck link. In the second, we choose a candidate algorithm and compare its performance against other similar and dissimilar traffic when competing for capacity on a bottle-neck link. This simulation illustrates the fairness of the chosen rate-control algorithm compared with other types of traffic.

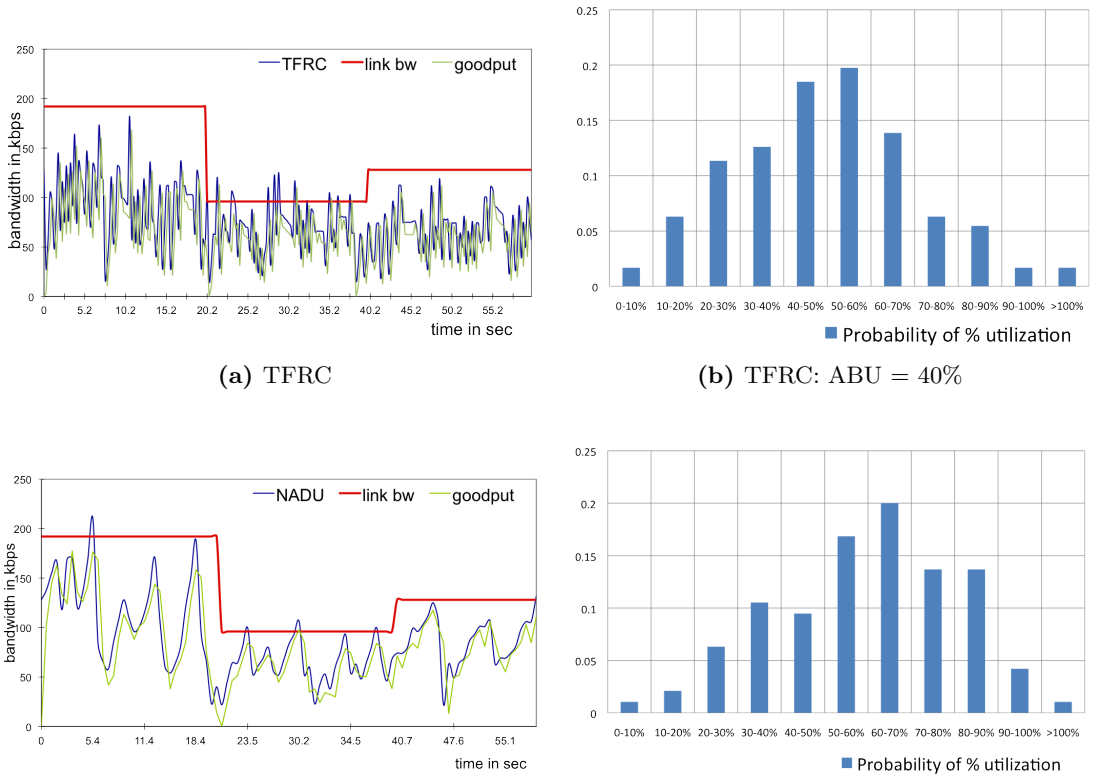
7.2.1 Single Flow

This scenario is chosen to test the stability of the rate adaptation algorithms. The scenario is simulated with a slowly changing link bit rate; i.e., at $t = 0$, 20, and 40 seconds, the bit rate changes to 192, 96, 128, respectively. All the links have a one-way delay of $240ms$. Figure 36 (left column) shows the instantaneous variation of the encoder rate and decoder goodput to the link bandwidth as described above. Table 3 presents the average encoder rate, average goodput, average PSNR and the delta loss rate (DLR) for the scenario.

TFRC bases its knowledge solely on normal RRs, and thus suffers from the large packet loss (4.4%) and underutilizes the link (40% ABU). C-NADU, on the other hand, produces better results in terms of delta loss rate (2.1%) and ABU (60%) when compared to TFRC. By comparing the instantaneous sending rates of TFRC and C-NADU, one notes that TFRC exhibits more rapid oscillations in sending rate than C-NADU. Therefore, TFRC is not well-suited for multimedia applications in the Internet scenario because it underutilizes the link. At the time of implementing the rate adaptation schemes, the draft Explicit Congestion Notification (ECN) for RTP over UDP [55] was not yet published. So C-NADU could not be compared with any network-assisted technology.

Table 3: Scenario: Internet links with stable and slow BW changes

	Avg. enc. rate (kbps)	Avg. goodput (kbps)	Loss Rate (%)	Avg. PSNR (dB)
TFRC	75.7	66.1	4.4%	30.5
C-NADU	88.5	80.9	2.1%	31.2



(a) TFRC

(b) TFRC: ABU = 40%

(c) C-NADU

(d) C-NADU: ABU = 60%

Figure 36: Plot of Link rate, encoder rate, goodput and Histogram of Probability of per-instance %Utilization (right column) and Average BW Utilization (ABU) in stable and slowly varying bandwidth scenario.

7.2.2 Streams competing for bandwidth against streams with similar and dissimilar traffic

The purpose of the following simulations is to observe the performance of C-NADU when 1) competing with other applications like Constant bit rate (CBR) & TCP-type traffic and 2) competing with other C-NADU users in the same local area network.

A VVoIP stream competing for capacity with dissimilar applications

This setup compares the performance of C-NADU with a TCP-type (File Transfer) and a CBR-type of traffic flow. Figure 37 describes two networks that are inter-connected by a $1Mbps$ bottleneck link and the users themselves are connected to the local area network over a high capacity link ($10Mbps$).

We run two simulation setups, one with $30ms$ end-to-end delay (low) and the other with $200ms$ end-to-end delay (high). In both the scenarios, the VVoIP stream starts with an ongoing CBR connection ($200kbps$), the TCP flow starts after 5 seconds. The simulation ends after $90s$ when the VVoIP call is terminated. The CBR and TCP flows finish at $80s$ and $85s$, respectively.

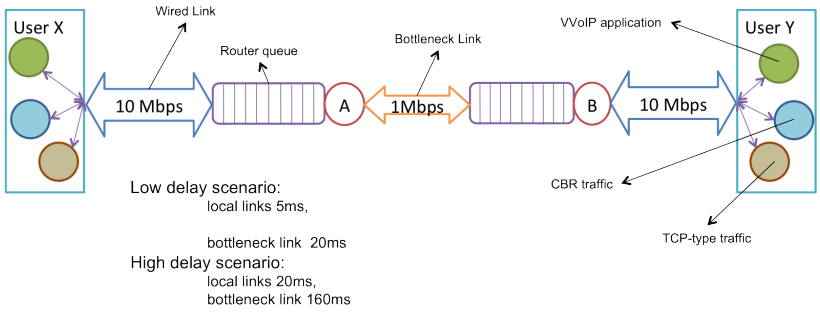
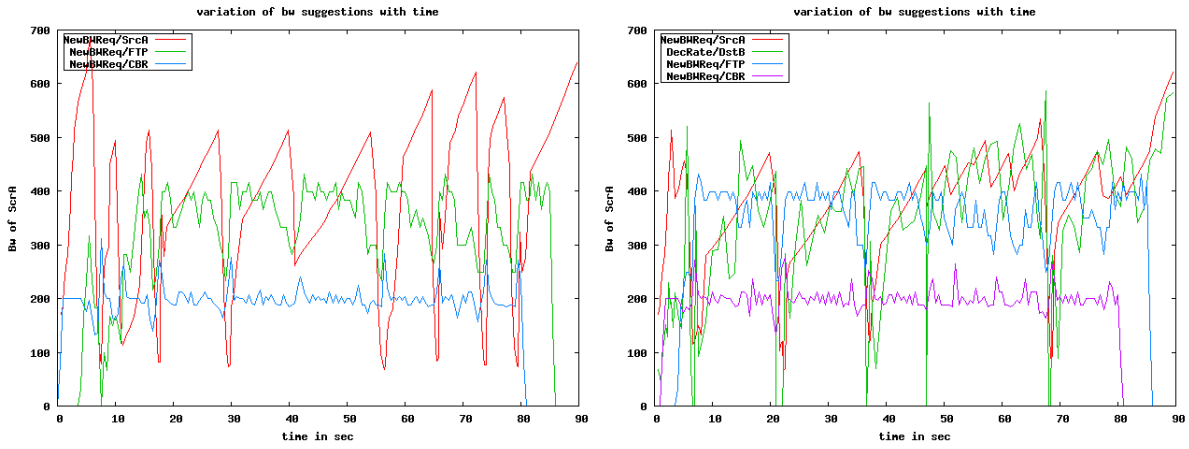


Figure 37: Simulation setup for comparing dissimilar traffic



(a) low delay **(b)** high delay

Figure 38: Plot of Application bit rates for a VVoIP stream, TCP and UDP sharing a common link of 1Mbps.

Figures 38 (a) and (b) show the channel utilization by each flow (TCP, CBR and C-NADU) in the low delay and high delay scenario, respectively. Initially, only C-NADU and the CBR flows are present, C-NADU begins to ramp-up the sender's encoding rate, attempting to capture the remaining channel capacity ($1000 - 200 = 800kbps$). However, C-NADU reduces its rate dramatically when the TCP flow is introduced at $t = 5s$.

One of the main difference between the low and high delay scenario is the oscillations in the sending rate of C-NADU. C-NADU tries to keep the receiver's buffer as full as possible, but the application-defined max delay ($delay_{max}$) and the low RTT makes the receiver buffer appear larger, i.e., with more packets (or frames) in the buffer. This larger buffer space makes the C-NADU behave more aggressively. This can be fixed by having an optimum playout delay that varies between perfect A/V synchronization and $delay_{max}$. This would correct the appearance of a large receiver buffer at the sender, and C-NADU would not be as aggressive in low delay networks[20].

These oscillations also affect the overall utilization of the bottleneck link. The total utilization is about $800kbps$ in the low and about $900kbps$ in the high end-to-end delay scenario, respectively. This may be attributed in part to the difference in the encoding rate (or

[20]This is not taken into account in this thesis but is part of ongoing work.

goodput) of C-NADU ($\approx 70kbps$). Therefore, higher utilization contributes to higher encoding bit rate for VVoIP application, which would translate in to better video quality. Table 4 shows the performance of C-NADU in the low and high end-to-end delay scenario, respectively.

Table 4: Scenario: Comparison between C-NADU, TCP and CBR Traffic

	Avg. enc. rate (kbps)	Avg. goodput (kbps)	Loss Rate (%)	Avg. PSNR (dB)
C-NADU (low e2e delay)	312.2	286.3	2.5%	31.3
C-NADU (high e2e delay)	382.9	351.6	1.25%	31.7

Two VVoIP streams sharing a bottleneck link

In this scenario, we compare the performance of two VVoIP calls, both using C-NADU for rate adaptation. We also observe the interaction between the two flows. In Figure 39, Users P and X are senders while Q and Y are receivers, the pairs also form two local area networks. The two networks are connected by a $1Mbps$ bottleneck link and all users are connected to their local area network over a high capacity link ($10Mbps$).

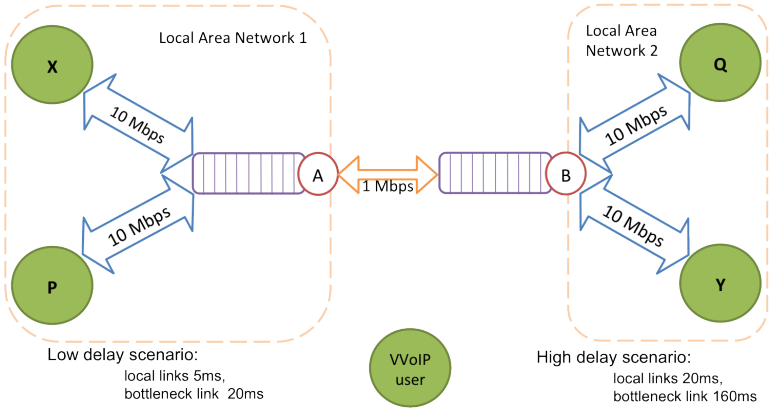
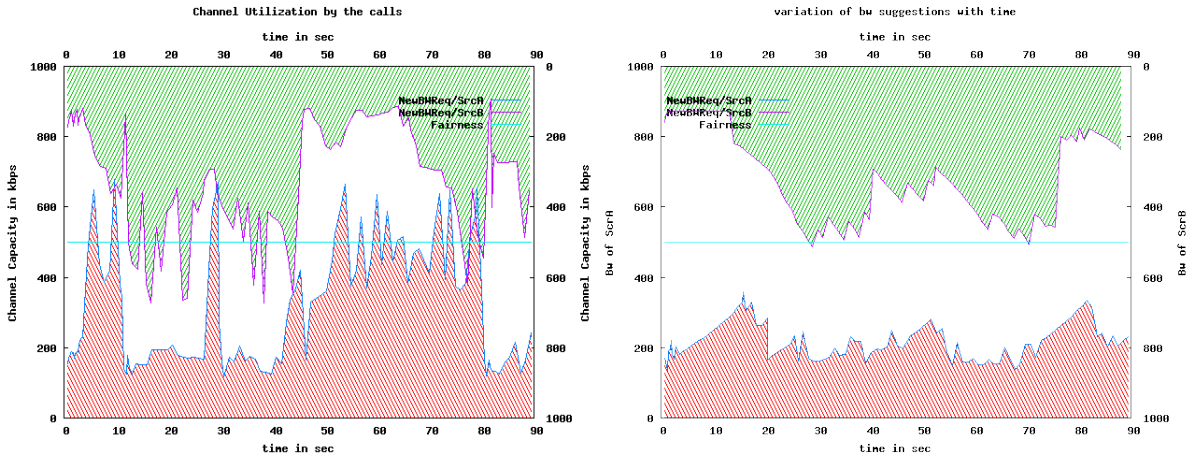


Figure 39: Simulator representation and setup

As in the previous scenario, we run two simulation setups, one with low ($30ms$) and the other with high ($200ms$) end-to-end delay. Figure 40 shows the load variation of the two VVoIP calls on the bottleneck link in both scenarios. The white space represents unused capacity on the bottleneck link that may be used by other applications. All the calls start at the same instant of time and have the same initial encoding rate ($128kbps$). Similar to the observations in the previous scenario, the C-NADU flows in the low delay scenario are more aggressive than in the high delay scenario because the low delay scenario has higher overall throughput and more losses when compared to the high delay scenario.

We also observe a pattern of "push back", i.e., one flow pushes back the other flow. This is more noticeable in the low delay scenario than in the high delay scenario due to the ability of C-NADU, to aggressively ramp-up. In Figure 40a, the Source B consumes a larger share in the period 10 to 40 sec. At the 40^{th} second, the Source A pushes back Source B and the situation reverses. In Figure 40b, these kind of push-backs are seen at instants t=15, 30, 50, 70 and 80, but the behavior is more conservative. This conservative



(a) low delay **(b)** high delay

Figure 40: Plot of Goodput of two VVoIP streams sharing a common bottleneck link of 1Mbps.

behavior is also reflected by the lower loss rate in the high delay scenario compared to that in the low delay scenario. The Table 5 summarizes the metrics for each call in the two scenarios.

Table 5: Scenario: Bottleneck sharing between Two VVoIP calls

	Avg. enc. rate (kbps)	Avg. goodput (kbps)	Loss Rate (%)	Avg. PSNR (dB)
Call 1 (low delay)	324.4	310.9	1.19%	31.8
Call 2 (low delay)	328.8	319.2	0.43%	32.2
Call 1 (high delay)	218.8	214.1	0.21%	32.9
Call 2 (high delay)	319.7	315.8	0.24%	32.7

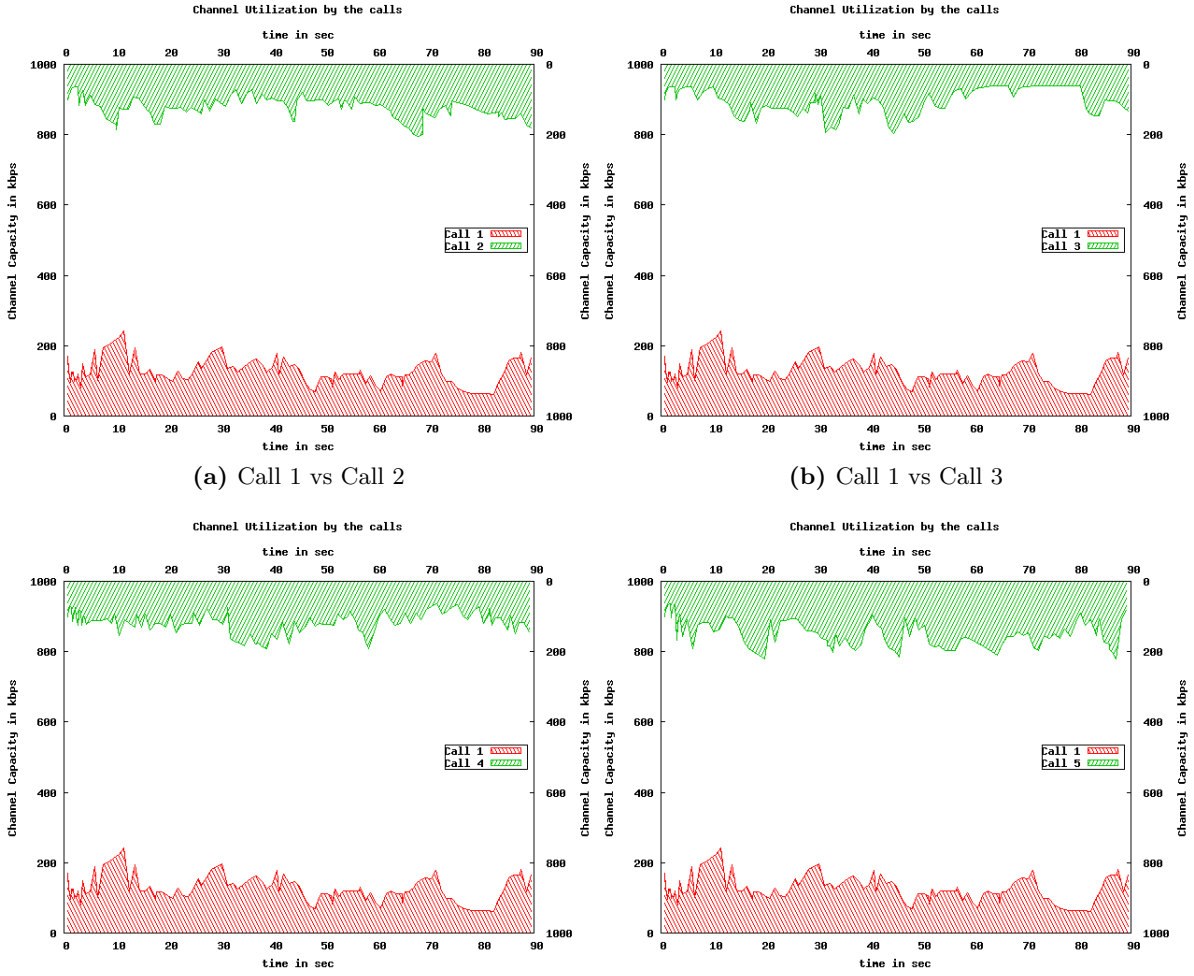
Many VVoIP streams sharing a bottleneck link

This scenario makes five VVoIP calls compete for channel capacity on a $1Mbps$ bottleneck link. Similar to the previous scenario, five callers and callees are connected to a local area network over a $10Mbps$ link and the end-to-end delay is $200ms$. It should also be noted that all the calls start and end at the same time.

Figures 41 (a), (b), (c), and (d) show the channel utilization of each call in relation to Call 1. For the five flows the average utilization varies between 50% to 75% for the expected $200kbps$/VVoIP flow. Table 6 presents the loss rate, average encoder rate and goodput for each VVoIP flow. Similar to the high delay scenario in the previous section, the flows are conservative (utilizing $\approx 600kbps$ of $1000kbps$) and have low loss rate ($\approx 0.35\%$).

7.2.3 Summary

In the simulated scenarios, we noticed that TFRC is not suitable for conversational video communication in the Internet because it underutilizes the link. C-NADU, on the other



(a) Call 1 vs Call 2

(b) Call 1 vs Call 3

(c) Call 1 vs Call 4

(d) Call 1 vs Call 5

Figure 41: Plots showing share of each VVoIP stream on the bottleneck link on the Internet

Table 6: Scenario: Bottleneck link shared between Five VVoIP calls on the Internet

	Avg. enc. rate	Avg. goodput	Loss Rate	Share
	(kbps)	(kbps)	(%)	(%)
Call 1	127.7	126.6	0.34%	63.85%
Call 2	116.5	113.9	0.26%	58.25%
Call 3	106.2	104.3	0.64%	53.1%
Call 4	112.1	110.9	0.39%	56.1%
Call 5	152.2	150.6	0.21%	76.1%

hand is able to perform better with higher bandwidth utilization and lower losses. Additionally, C-NADU is fair and competitive with similar and dissimilar traffic types. However, C-NADU is found to be conservative in the high-delay scenario and aggressive in the low-delay scenario. This is due to the estimated receiver buffer by the sender and can be fixed by an adaptive playout.

7.3 Rate control in a Heterogeneous networks

This scenario simulates a real-world scenario, where the user on the Internet initiates a VVoIP call with a user on a 3G connection, or vice-versa. This is known as a heterogeneous environment or network. The purpose of the simulation is to evaluate the behavior of C-NADU in a heterogeneous environment, where it competes for capacity on a bottleneck link with other similar VVoIP calls, and also adapts to the variable end-to-end capacity.

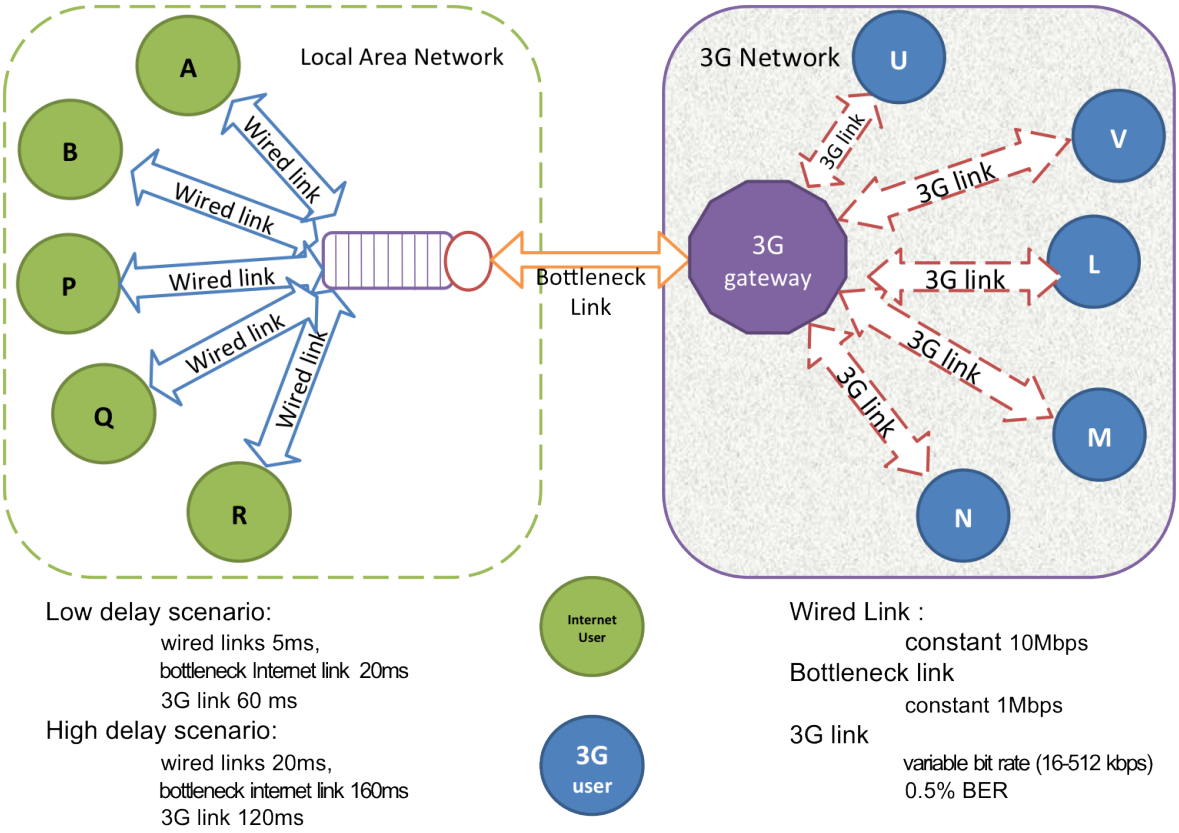


Figure 42: Mixed or heterogeneous simulation setup

Figure 42 shows the simulation setup with five callers and five callees. The five callers are connected to the wired Internet by $10Mbps$ links and the five callees are using the 3G mobile network. The channel of each callee is separate and undergoes different type of fading and interference, which results in different channel bandwidth for each user. The capacity of each 3G channel can vary from about $16kbps$ to $512kbps$. The 3G channel variation for each callee is as follows:

1. Call 1 - Also called "Excellent Call" follows the pattern: Excellent-Poor-Elevator.
2. Call 2 - Also called "Good Call" follows the pattern: Good-Good-Poor.
3. Call 3 - Also called "Poor Call" follows the pattern: Poor-Poor-Poor.
4. Call 4 - Also called "Fair Call" follows the pattern: Fair-Fair-Poor.
5. Call 5 - Also called "Elevator Call" follow the pattern: Excellent-Elevator-Poor.

Each pattern is $60s$ long and the "Excellent" ($\approx 400kbps$), "Elevator" ($\approx 300kbps$), "Good" ($\approx 250kbps$), "Fair" ($\approx 200kbps$), and "Poor" ($\approx 60kbps$) patterns are based

on 3GPP RAN traces [47]. Unlike the Internet scenarios, where no bit-error losses were simulated, the heterogeneous scenario has bit-errors that generates additional losses. Each 3G link exhibits a 0.5% link-layer bit-error rate (BER) [48]. The two networks are interconnected by a $1Mbps$ bottleneck link. Depending on the behavior of the 3G link, the constraining link for each call can alternate between the 3G link and the bottleneck link. Therefore, the rate-control algorithm has to take the overall path characteristic into account and not only the individual links.

Similar to the Internet simulations, there are two scenarios: low ($85ms$) and high ($300ms$) end-to-end delay, respectively.

7.3.1 High-delay Heterogeneous network

Figures 43 (a), (b), (c) and (d) show the channel utilization of each call with reference to Call 1 and Figure 43e shows the evolution of channel utilization with time. Table 7 presents the average goodput, loss rate and ABU for each call.

Table 7: Scenario: Five calls in a heterogeneous network with high delay

	Avg. goodput	Loss Rate	Avg. PSNR	ABU
	(kbps)	(%)	(dB)	(%)
Call 1	140.10	2.15%	31.4	70.1%
Call 2	133.55	1.61%	31.9	66.8%
Call 3	35.18	1.55%	32.2	17.59%
Call 4	114.96	2.75%	31.1	57.5%
Call 5	130.23	2.25%	31.3	65.1%

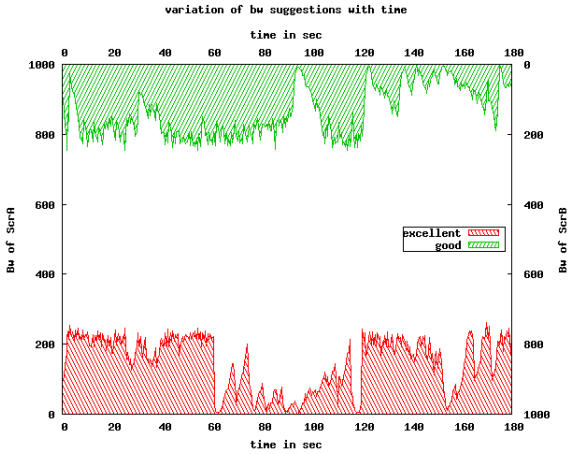
The end-to-end capacity for a call depends on the fair sharing of capacity on the bottleneck link and the capacity of the 3G link. Figure 43b, Call 3 ("Poor Call"), shows that during the whole simulation, the 3G link was the constraint and the average goodput of the call is $\approx 35kbps$.

In the first $60s$ of the simulation, the average channel capacity on the remaining four 3G links is greater than $250kbps$ and in this period the associated calls exhibit fair usage of the bottleneck link ($\approx 180 - 220kbps$). Furthermore, in the last $60s$ of the simulation, when four out of five 3G links have poor connectivity, then Call 1 ("Excellent Call"), which is transitioning from poor to better connectivity, is able to quickly adapt its encoding rate to occupy more of the end-to-end capacity.

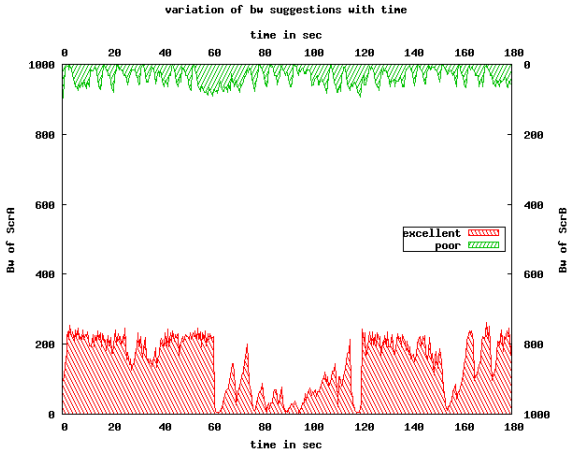
7.3.2 Low-delay Heterogeneous network

Figures 44 (a), (b), (c) and (d) show the channel utilization of each call with reference to Call 1 and Figure 44e shows the evolution of channel utilization with time. Table 8 presents the following metric for each call: average goodput, loss rate and ABU.

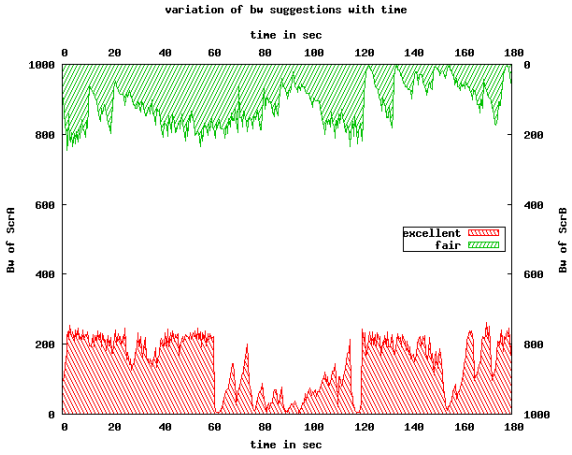
Similar to the results in the high-delay scenario, each VVoIP call is fair to one another as shown by the first $60s$ of the simulation, and Call 1 recovers quickly after poor connectivity in the last $60s$ of the simulation. However, one of the main differences between the two scenarios is the aggressiveness of C-NADU in the low-delay scenario compared with the high-delay scenario. This is observed by the oscillations in the encoding rate of all the



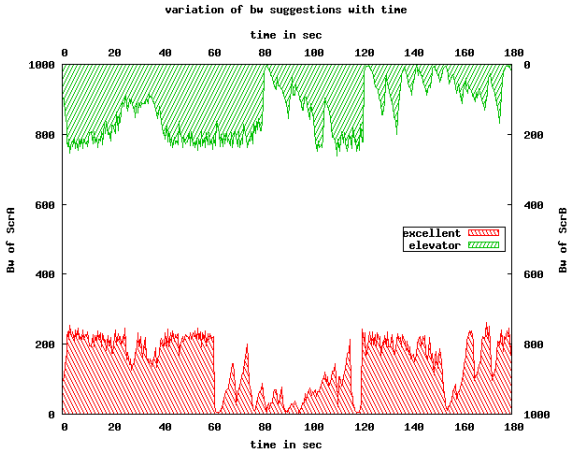
(a) Call 1 vs Call 2



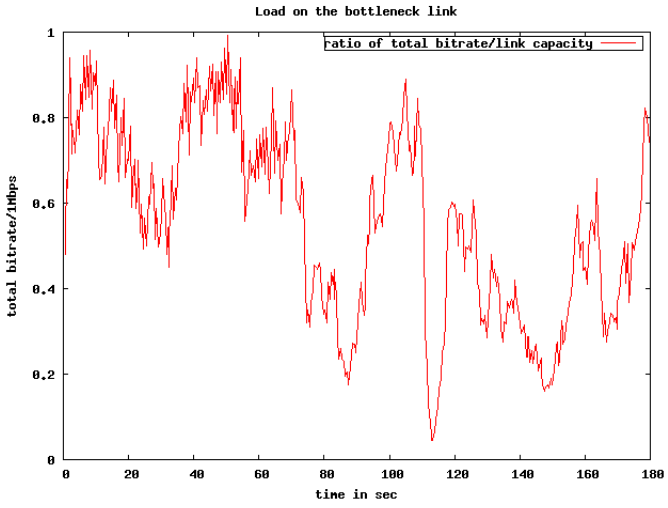
(b) Call 1 vs Call 3



(c) Call 1 vs Call 4

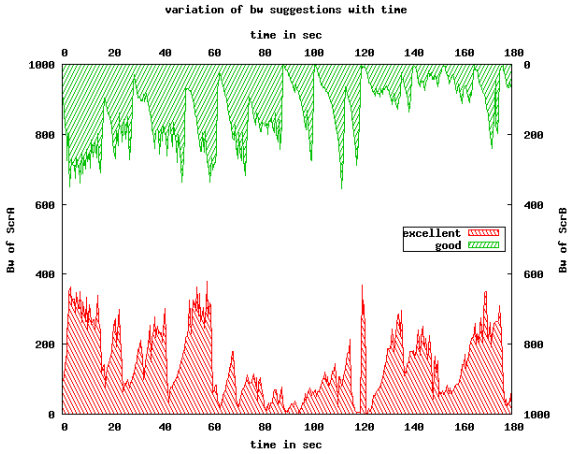


(d) Call 1 vs Call 5

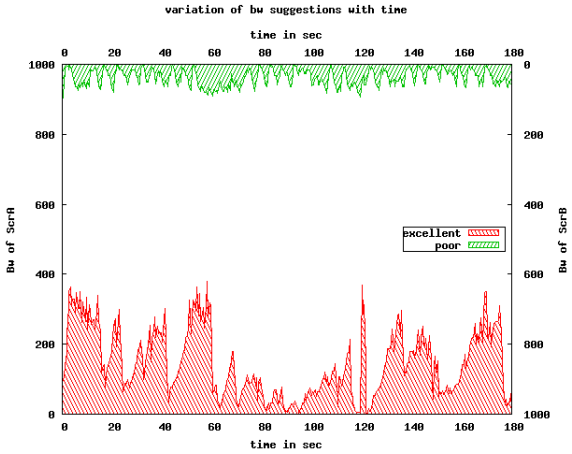


(e) stress on the bottleneck link

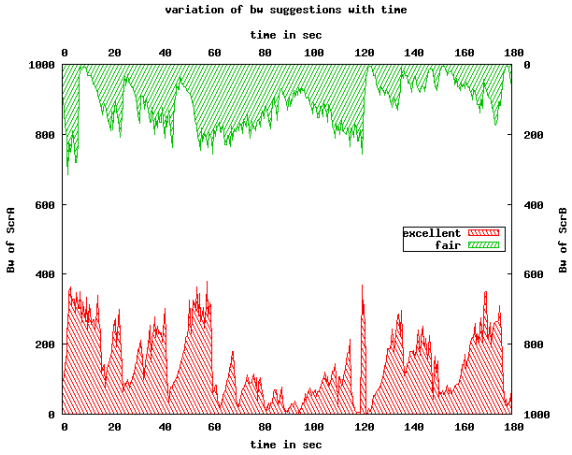
Figure 43: Five calls competing for bandwidth in a high-delay mixed network



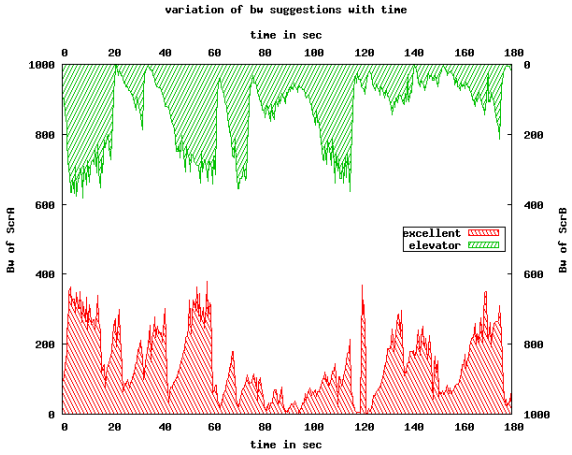
(a) Call 1 vs Call 2



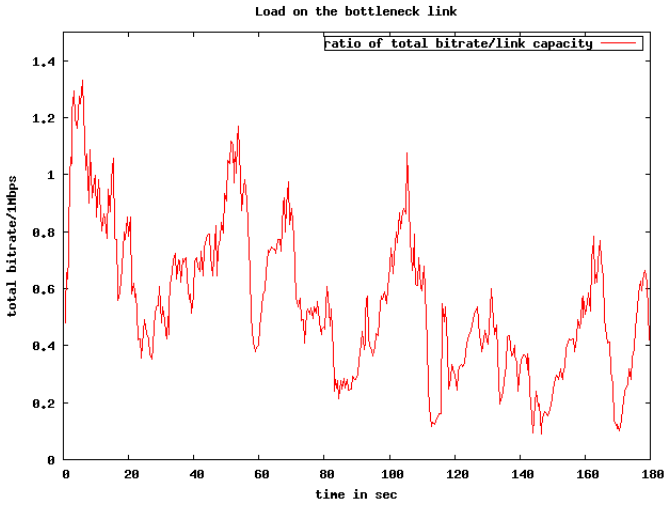
(b) Call 1 vs Call 3



(c) Call 1 vs Call 4



(d) Call 1 vs Call 5



(e) stress on the bottleneck link

Figure 44: Five calls sharing bandwidth in a low-delay heterogeneous network

calls and the over-utilization of the bottleneck link at $t = 10$, $t = 50$, $t = 110$ seconds (see Figure 44e). The over-utilization causes a slightly higher congestion loss in the low-delay scenario when compared to the high-delay scenario (compare DLR of Tables 7 and 8, average difference $\approx 0.25\%$).

Table 8: Scenario: Five calls in a heterogeneous network with low delay

	Avg. goodput (kbps)	Loss Rate (%)	Avg. PSNR (dB)	ABU (%)
Call 1	140.45	2.67%	31.3	70.2%
Call 2	132.38	2.32%	31.2	66.2%
Call 3	35.18	1.21%	32.4	17.6%
Call 4	112.85	2.95%	31.1	56.4%
Call 5	136.46	2.37%	31.2	68.2%

7.3.3 Summary

In the simulated scenarios, C-NADU is able to adapt to the constraints imposed by the shared bottleneck link and the variable capacity of the 3G links. The lack of an adaptive playout at the receiver in the low-delay scenario makes C-NADU more sensitive to the receiver's playout buffer, which causes oscillations and a slightly higher loss rate, when compared to the high-delay scenario. However, C-NADU in both scenarios is fair to other VVoIP calls on the shared bottleneck, yet aggressively utilizes the available end-to-end channel capacity.

7.4 Conclusion

Based on the simulated scenarios, we conclude that network-assistance in 3G networks provides the best results; however, this is not always realistic in other scenarios such as those based on the Internet and heterogeneous networks.

Out of the unassisted rate-control algorithms, we conclude that TFRC is unsuitable for multimedia communication because of oscillating sending rates that causes high losses. TMMBR-U (unassisted) is conservative and causes underutilization of the end-to-end capacity, which leads to lower video quality for the end-user. C-NADU has a comparable loss rate to TMMBR-U but higher end-to-end channel utilization.

Furthermore, in the simulated scenarios, C-NADU exhibits fair sharing of channel capacity on the bottleneck link with similar and dissimilar type of traffic. While its performance is suboptimal in low-delay networks, using an adaptive size of playout buffer might correct the problem.

8 Conclusion

In this thesis, we developed and evaluated rate-control mechanisms for conversational video communication in the 3G, wired Internet, and heterogeneous multimedia environments. In the process of developing them, we studied the behavior of the congestion indicators in each environment and evaluated them for possible application in a rate-control algorithm. Table 9 summarizes our observations and describes the possible application of each congestion indicator in an heterogeneous environment. In the Table, the "Type of cue" column denotes the timeliness of the cue to indicate congestion. This can be classified as, *early*, *slow*, *long-term*, and *late*. "Early" means it appears at the onset of congestion, while "late" signifies that the congestion cannot be avoided any more. "Long-term" requires capturing the variation of the cue over time (history) to make decisions and "slow" means that it takes time for the cue to indicate congestion. The "observation" column describes the behavior of the cue and the "Possible usage" column describes the application of the cue. We have applied some of these cues in congestion control algorithms, showing that they can be used effectively.

To minimize the transmission overhead, non-compound RTCP packets [52] may be used, e.g., to remove the need for SDES items as no source identification is needed in a point-to-point setup. To introduce network-assistance in the simulation system, ECN for RTP [55] may be used by the routers. We believe that marking packets with ECN would provide congestion cues similar to the run-length encoded lost and discarded packet information in RTCP-XR. To reduce the oscillations in the sending rate in low-delay networks, an adaptive playout may be used. Validation of these hypotheses remains for further study.

While this thesis focussed on using RTP for transmitting real-time streams, we nowadays, observe trends towards using TCP, in spite of its shortcomings - particularly for better integration with the web infrastructure.

HTML 5 [56] defines new tags such as `<video>` and `<audio>`. With these it is possible to stream (progressive download) audio/video without the use of additional plugins such as Adobe flash [57]. However, additional tags such as `<devices>` and `<stream>` provide new APIs for accessing a web camera and to do real-time streaming [56]. "Global Mobile Broadband Traffic Report" [58] studied 190 million subscribers worldwide. The report reveals that VoIP and IM is the second largest growing application in 2010 with an 84% increase behind video streaming (at 92%). However, VoIP only occupies 3% of the subscriber's bandwidth when compared to video streaming which occupies 35%. In 2010, Skype has been making foray in to VoIP calls over 3G Mobile Internet [59–61] and this is because of the service providers transitioning to a tiered pricing model [62]. Furthermore, FaceTime from Apple [63] and ūmi Telepresence from Cisco [64] are laying the pathway for more consumer voice and video products to appear.

Table 9: Summary of Congestion Indicators.

Indicator	Type of cue	Observation	Possible usage
RTT	long-term cue	Fluctuates when the network queues build-up and reduce. The RTT variation is smooth depending on adaptation	sender needs to collect history to observe a trend, it can be used for fairness
Jitter	monitoring, values also depend on cross-traffic	Spikes in fixed networks can be correlated but not in mobile networks	A receiver or sender needs to compare jitter values over long time scales
Loss Rate	slow and late	Hard to distinguish bit-error losses from congestion losses.	if link are already congested, the sender should undershoot the current channel capacity to offset the loading.
Discard Rate	early	Discard indicates congestion along the path, packet loss followed by discards may suggest congestion as opposed to bit-error losses.	undershoot the receiver rate to decongest the link.
Playout Delay	early	can indicate underflow and overflow, better indicator than RTT but dependent on maximum allowed delay.	fine tunes the sending rate
Frames in Transit (FiT)/ Frame in Buffer FiB)	early but computationally intensive if a frame gets fragmented in to multiple packets	Useful, if the video is encoded in constant frame rate	for ramp-up after undershooting
Inter Arrival Time (IAT)	early	better indicator than jitter in case of conversational video in 3G because of quicker feedback and larger variation in Bandwidth.	needs to be dampened else it is quite aggressive in ramp-up and ramp-down.
ΔT_{HSN} (Delta_T)	slow	very good for low delay networks, conservative for high latency networks	can cause rapid oscillations in sending rates, needs to dampened.

References

[1] comScore, Fetched January 2010. [Online]. Available: http://www.comscore.com

[2] comScore, "November Sees Number of U.S. Videos Viewed Online Surpass 30 Billion for First Time on Record," Fetched January 2010. [Online]. Available: http://comscore.com/Press_Events/Press_Releases/2010/1/November_Sees_Number_of_U.S._Videos_Viewed_Online_Surpass_30_Billion_for_First_Time_on_Record

[3] Youtube, Fetched January 2010. [Online]. Available: http://www.youtube.com

[4] Hulu, Fetched January 2010. [Online]. Available: http://www.hulu.com

[5] SlingBox, Fetched January 2010. [Online]. Available: http://www.slingbox.com/

[6] TiVo, Fetched January 2010. [Online]. Available: http://www.tivo.com/whatistivo/tivois/index.html

[7] Hava Player, Fetched January 2010. [Online]. Available: http://www.myhava.com/

[8] FCC, "IP-Enabled Services: Voice over IP (VoIP)," Fetched January 2010. [Online]. Available: http://www.fcc.gov/voip/

[9] Wikipedia, "List of SIP Software," Fetched January 2010. [Online]. Available: http://en.wikipedia.org/wiki/List_of_SIP_software

[10] Skype, Fetched January 2010. [Online]. Available: http://www.skype.com

[11] Gizmo Project, Fetched January 2010. [Online]. Available: http://www.google.com/gizmo5/

[12] J. Nielsen, "Nielsen's Law of Internet Bandwidth," Fetched January 2010. [Online]. Available: http://www.useit.com/alertbox/980405.html

[13] Wikipedia, "Nielsen's Law of Internet Bandwidth," Fetched January 2010. [Online]. Available: http://en.wikipedia.org/wiki/Jakob_Nielsen_(usability_consultant)

[14] ——, "Moore's Law," Fetched January 2010. [Online]. Available: http://en.wikipedia.org/wiki/Moore's_law

[15] International Telecommunication Union (ITU), "Introduction to Mobile Cellular Technology," Fetched January 2010. [Online]. Available: http://www.itu.int/osg/spu/imt-2000/technology.html

[16] ITU-T G.992.1, "Asymmetric digital subscriber line (ADSL) transceivers," Fetched January 2010, 1999-2007. [Online]. Available: http://www.itu.int/rec/T-REC-G.992.1/en

[17] K. Jack, *Video Demystified: A Handbook for the Digital Engineer*. L L H Technology Publishing, 1995.

[18] BitTorrent, Fetched January 2010. [Online]. Available: http://www.bittorrent.com/

[19] Wikipedia, "Web browser," Fetched January 2010. [Online]. Available: http://en.wikipedia.org/wiki/Web_browser

[20] ——, "Instant Messaging (IM)," Fetched January 2010. [Online]. Available: http://en.wikipedia.org/wiki/Instant_messaging

[21] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681 (Draft Standard), Internet Engineering Task Force, Sep. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5681.txt

[22] ITU-T Rec. H.264, "Advanced video coding for generic audiovisual services." [Online]. Available: http://www.itu.int/rec/T-REC-H.264/en

[23] L. Hanzo, P. Cherriman, and J. Streit, *Video Compression and Communications: From Basics to H.261, H.263, H.264, MPEG4 for DVB and HSDPA-Style Adaptive Turbo-Transceivers*. Wiley-IEEE Press, 2007.

[24] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550 (Standard), Jul. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt

[25] 3GPP S4-080771, "MTSI Video Dynamic Rate Adaptation: Evaluation Framework ver 1.0." 3rd Generation Parnership Proejct (3GPP), Proposal S4-080771, Oct. 2008. [Online]. Available: http://www.3gpp.org/FTP/tsg_sa/WG4_CODEC/TSGS4_51/Docs/S4-080771.zip

[26] Mathworks, "Simulink: PSNR Calculation," Fetched April 2010, 1982-2010-. [Online]. Available: http://www.mathworks.com/access/helpdesk/help/toolbox/vipblks/ref/psnr.html

[27] ITU-T J.247, "Objective perceptual multimedia video quality measurement in the presence of a full reference," Fetched April 2010, 1999-2008, pre-published. [Online]. Available: http://www.itu.int/rec/T-REC-J.247/en

[28] Z. Li, "A saliency map in primary visual cortex," 2002.

[29] E. Ong, X. Yang, W. Lin, Z. Lu, S. Yao, X. Lin, S. Rahardja, and B. Seng, "Perceptual quality and objective quality measurements of compressed videos," *Journal of Visual Communication and Image Representation*, vol. 17, no. 4, pp. 717–737, 2006.

[30] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)," RFC 4585 (Proposed Standard), Jul. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4585.txt

[31] T. Friedman, R. Caceres, and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)," RFC 3611 (Proposed Standard), Nov. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3611.txt

[32] J. Devadoss, V. Singh, J. Ott, C. Liu, Y.-K. Wang, and I. Curcio, "Evaluation of error resilience mechanisms for 3G conversational video," *Multimedia, International Symposium on*, vol. 0, pp. 378–383, 2008.

[33] S. Wenger, U. Chandra, M. Westerlund, and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)," RFC 5104 (Proposed Standard), Feb. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5104.txt

[34] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. New York, NY, USA: ACM, 2000, pp. 43–56.

[35] S. Floyd, E. Kohler, and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)," Internet Engineering Task Force, RFC 4342, Mar. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4342.txt

[36] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," Internet Engineering Task Force, RFC 5348, Sep. 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5348.txt

[37] L. Gharai, "RTP with TCP Friendly Rate Control," work in progress, January 2008. [Online]. Available: http://tools.ietf.org/id/draft-ietf-avt-tfrc-profile-10.txt

[38] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," Internet Engineering Task Force, RFC 3448, Jan. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3448.txt

[39] 3GPP TS 26.234, "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs." [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26234.htm

[40] I. Curcio and D. Leon, "Application rate adaptation for mobile streaming," *WOWMOM '05: Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*, pp. 66–71, 13-16 June 2005.

[41] ——, "Evolution of 3gpp streaming for improving qos over mobile networks," *ICIP 2005: IEEE International Conference on Image Processing, 2005.*, vol. 3, pp. III–692–5, 11-14 Sept. 2005.

[42] D. Mahrenholz and S. Ivanov, "Real-time network emulation with ns-2," in *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*, 2004, pp. 29 – 36.

[43] Nokia, "Nokia's public h.264 codec." [Online]. Available: http://research.nokia.com/page/4988

[44] S. Wenger, M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RTP Payload Format for H.264 Video," Internet Engineering Task Force, RFC 3984, Feb. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3984.txt

[45] 3GPP TR 26.902, "Video codec performance," 3rd Generation Parnership Proejct (3GPP), TR 26.902, Jan. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26902.htm

[46] 3GPP, "Radio Link Control (RLC) protocol specification," 3rd Generation Partnership Project (3GPP), TS 25.322, Sep. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25322.htm

[47] 3GPP R1-081955, "LTE Link Level Throughput Data for SA4 Evaluation Framework." 3rd Generation Parnership Proejct (3GPP), Proposal R1-081955, May 2008. [Online]. Available: http://www.3gpp.net/ftp/tsg_ran/WG1_RL1/TSGR1_53/Docs/R1-081955.zip

[48] 3GPP S4-050560, "Software Simulator for MBMS Streaming over UTRAN and GERAN," 3rd Generation Parnership Proejct (3GPP), Proposal S4-050560, Sep. 2005. [Online]. Available: http://www.3gpp.org/FTP/tsg_sa/WG4_CODEC/TSGS4_36/Docs/S4-050560.zip

[49] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 392–403, 2001.

[50] H. Garudadri, H. Chung, N. Srinivasamurthy, and P. Sagetong, "Rate adaptation for video telephony in 3G networks," *Packet Video 2007*, pp. 342–348, Nov. 2007.

[51] J. Ott, I. Curcio, and V. Singh, "Real-time Transport Control Protocol Extension Report for Run Length Encoding of Discarded Packets," work in progress, September 2009. [Online]. Available: http://tools.ietf.org/id/draft-ott-avt-rtcp-xt-discard-metrics-01.txt

[52] I. Johansson and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences," Internet Engineering Task Force, RFC 5506, Apr. 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5506.txt

[53] Henning Schulzrinne, "Some Frequently Asked Questions about RTP: How is the jitter computed?" Fetched November 2010. [Online]. Available: http://www.cs.columbia.edu/~hgs/rtp/faq.html#jitter

[54] V. Singh, J. Ott, and I. Curcio, "Rate adaptation for conversational 3G video," in *INFOCOM Workshops 2009, IEEE*, April 2009, pp. 1–7.

[55] M. Westerlund, I. Johansson, C. Perkins, P. O'Hanlon, and K. Carlberg, " Explicit Congestion Notification (ECN) for RTP over UDP," work in progress, March 2010. [Online]. Available: http://tools.ietf.org/id/draft-ietf-avt-ecn-for-rtp-01.txt

[56] D. H. Ian Hickson, "HTML5: A vocabulary and associated APIs for HTML and XHTML," W3C, Working Draft 5506, 2010. [Online]. Available: http://www.w3.org/TR/html5/

[57] Adobe, "Flash player," Fetched May 2010. [Online]. Available: http://www.adobe.com/products/flashplayer/

[58] A. MobileTrends, "Global Mobile Broadband Traffic Report," Allot Communications, Technical Report, 2010. [Online]. Available: http://www.allot.com/MobileTrends_Report_H1_2010.html

[59] Peter Parkes, "Say hello to Skype for Maemo (N900)," Fetched March 2010. [Online]. Available: http://blogs.skype.com/en/2009/11/video_demo_skype_for_nokia_n90.html

[60] ——, "Say hello to Skype for Symbian," Fetched March 2010. [Online]. Available: http://blogs.skype.com/en/2010/03/symbian.html

[61] ——, "Skype now available for Android phones," Fetched October 2010. [Online]. Available: http://blogs.skype.com/en/2010/10/android.html

[62] ——, "iPhone update: now supports multitasking, no additional charges for calling over 3G," Fetched August 2010. [Online]. Available: http://blogs.skype.com/en/2010/07/iphone_multitasking_3g.html

[63] Apple, "FaceTime: Video Communication," Fetched September 2010. [Online]. Available: http://www.apple.com/iphone/features/facetime.html

[64] Cisco, "ūmi Telepresence," Fetched October 2010. [Online]. Available: http://umi.cisco.com/

[65] D. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," Internet Engineering Task Force, RFC 2234, Nov. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2234.txt

Appendix A: Header overhead for RTP/RTCP packets

Protocol	Size in bytes	Size in bits	Notes:
IP	20	160	
UDP	8	64	
TCP	20	160	+32bits if OPTIONS included.
RTP	12	96	+CSRC+Ext Headers (extra bytes)
TFRC+RTP ext	12	96	have to add RTP header. RTP header extension with RTT and send TS
RTCP	8	64	
SR	20	160	have to add RTCP. PT = 200.
RR	24	192	
NADU	24	192	is similar to RTCP APP. Incl. RTCP + "PSS0"+NADU data (96 bytes)
XR	12	96	Incl. RTCP + no. of XR blocks. To skip just read length in RTP.
rle-disc-bytes	12	96	have to add XR basic header. Plus for disc-pkt 64 bits for every extra chunk.
rle-disc-pkt	16	128	
AVPF header	12	96	modified RTCP with FMT and PT to distinguish. SSRC of packet and media incl
TMMBR	8	64	have to add AVPF header. PT=RTPFB (205) and FMT=3
TFRC-FB	20	160	have to add AVPF header. PT=RTPFB (205) and FMT=2

Appendix B: SDP Signaling

SDP for NADU

The RTCP Rate adaptation capability can be signaled in the session setup [39] using the attribute defined below in Augmented Backus-Naur Form (ABNF) [65]

```
sdp-Adaptation-line  = "a" "=" "3GPP-Adaptation-Support"
                                  ":" report-frequency CRLF
report-frequency  = NonZeroDIGIT [ DIGIT ]
NonZeroDIGIT = \%x31-39 ;1-9
```

SDP for Discarded Packets

The RTCP packets discarded after arrival capability can be signaled in the session setup using the attribute defined below in Augmented Backus-Naur Form (ABNF) [65]

```
rtcp-xr-attrib = "a=" "rtcp-xr" ":" [xr-format *(SP xr-format)]
                 CRLF   ; defined in [RFC3611]
xr-format       =/ xr-discard-rle
                 / xr-discard-bytes
xr-discard-rle   = "discard-rle"
xr-discard-bytes = "discard-bytes"
```

SDP for TMMBR

The TMMBR capability can be signaled in the session setup using the attribute defined below in Augmented Backus-Naur Form (ABNF) [65]

```
rtcp-fb-val        =/ "ccm" rtcp-fb-ccm-param
rtcp-fb-ccm-param  = SP "tmmbr" [SP "smaxpr=" MaxPacketRateValue]
                                  ; Temporary max media bit rate
                                  ; defined in [RFC 5104]
                               / SP token [SP byte-string]
                                 ; for future commands/indications
subMessageType = 1*8DIGIT
byte-string = <as defined in section 4.2 of [RFC4585] >
MaxPacketRateValue = 1*15DIGIT
```

SDP for C-NADU

The Video Communication Rate Adaptation capability can be signaled in the session setup using the attribute defined below in Augmented Backus-Naur Form (ABNF) [65]

```
sdp-Adaptation-line  = "a" "=" "3GPP-Conversational-Adaptation-Support"
                                  ":" report-frequency CRLF
                                  ; extended from 3GPP TS 26.234
report-frequency  = NonZeroDIGIT [ DIGIT ]
NonZeroDIGIT = \%x31-39 ;1-9
```

Appendix C: TCL files for $ns2$

3G script

```
############NS2 RTP Simulation##########################
set ns [new Simulator]
# Simulation start Time
set SimStart 0.0
# Simulation End Time
set SIMTIME 300.0
# Bandwidth of the link 128Kbps, over-rided if using -linkBwChanges
set LINK_BW 128
# Propagation delay in the link 60ms or 120ms
set DELAY 120ms

#Initialize two nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Initialize two RTP Transport Agents
set rtptran1 [new Agent/IVI_RTP]
set rtptran2 [new Agent/IVI_RTP]
$rtptran1 set iD_ 10
$rtptran2 set iD_ 20

$rtptran1 set user_rtcp_fdbk_rate_ 1
$rtptran2 set user_rtcp_fdbk_rate_ 1

$rtptran1 set EncoderBw_ 128
$rtptran2 set EncoderBw_ 128

$rtptran1 set ExtendedRRInfo_ 1
$rtptran2 set ExtendedRRInfo_ 0

###################
# IVI-simplex-link usage:
# ( To fill in the parameters Refer the file "bearer.txt")
# Parameters Passed to the link
# n1
# n2
# Link Bandwidth (In bits per second)
# Link delay (In seconds)
# Link Frame Size (In bytes)
#
# CRUIH -> Compressed RTP/UDP/IP Header length in bytes
# (This is the value that would be added after
# removing the RTP/UDP/IP header
#
# RLC_Hdr_Size -> Header length that would be added
# for every RLC frame
#
# MaxSendingDelay (In milliseconds) ->
# If a packet has been delayed
# more than this value, then it would dropped at the sender side.
# (if 0 - then this check is disabled)
#
# MaxE2EDelay (In milliseconds) ->
# If a packet has took more time than this value,
# then it would be dropped at the receiver side.
# (if 0 - then this check would be disabled)
#
# Error File Name (A list of error files is available
# at ns-allinone/ns-2.29/rtpsim/link/ErrorPatterns/)
#
# ErrorFormat (ascii/binary - currently only tested for "ascii")
# Seed (A random seed number)
#
# linkBwChanges (a list of RLC sizes per millisecond)
####################

#Configuring the links
$ns hut-link-config -startNode $n1 \
               -endNode $n2 \
               -duplex ON \
               -linkBw $LINK_BW \
               -linkDelay $DELAY \
               -frameSize 41 \
               -cruiHdrSize 0 \
               -rlcHdrSize 1 \
               -maxSendingDelay 0 \
               -maxE2EDelay 400 \
               -errorFile PSC__128kbps_20ms_BLER_var_orig.txt \
               -errorFormat ascii\
               -seedValue 0 \
               -iDNo 1 \
       -linkBwChanges "UL_poor_1.txt"

$ns hut-link-config -startNode $n2 \
               -endNode $n3 \
```

```
                -duplex ON \
                -linkBw $LINK_BW \
                -linkDelay $DELAY \
                -frameSize 41 \
                -cruiHdrSize 0 \
                -rlcHdrSize 1 \
                -maxSendingDelay 0 \
                -maxE2EDelay 400 \
                -errorFile PSC__128kbps_20ms_BLER_var_orig.txt \
                -errorFormat ascii\
                -seedValue 0 \
                -iDNo 3 \
        -linkBwChanges "UL_poor_1.txt"

#Attaching the agents to the nodes
$ns attach-agent $n1 $rtptran1
$ns attach-agent $n3 $rtptran2

#Creating traffic generators and sink
set rtpgen1 [new Application/RTPTrafficGenerator]
set rtpgen2 [new Application/RTPTrafficGenerator]
$rtpgen1 set iD_ 100
$rtpgen2 set iD_ 200
#$rtpgen1 set ModifyTmmbrRequest_ 1
#Application/RTPTrafficGenerator set ModifyTmmbrRequest_ 1

#Attaching transport agents to the application
$rtpgen1 attach-transport-agent $rtptran1
$rtpgen2 attach-transport-agent $rtptran2

#Attaching applications to the transport agent
$rtptran1 attach-application $rtpgen1
$rtptran2 attach-application $rtpgen2

$rtptran1 generate-rtcp $LINK_BW
$rtptran2 generate-rtcp $LINK_BW

#$rtptran1 rtcp-dump "encrtcp.txt"
#$rtptran2 rtcp-dump "decrtcp.txt"


#Attaching Interfaces
set interface "REAL"
set ipaddr "130.233.x.x"
set port1 "5000"
set port2 "7000"

$rtpgen1 attach-interface $interface $ipaddr $port1
$rtpgen2 attach-interface $interface $ipaddr $port2

#Connecting the agents
$ns connect $rtptran1 $rtptran2
$ns connect $rtptran2 $rtptran1

$ns at 0 "$rtpgen1 generate"
$ns at 0 "$rtpgen2 generate"
$ns at $SIMTIME "$rtpgen1 stop"
$ns at $SIMTIME "$rtpgen2 stop"

$ns at $SIMTIME "$ns halt"

#Run the simulation
$ns run

$rtptran1 delete
$rtptran2 delete
$rtpgen1 delete
$rtpgen2 delete

exit 0
########################################
```

Internet

Two competing RTP sources

```
############NS2 RTP Simulation#########################
set ns [new Simulator]
# Simulation start Time
set SimStart 0.0
# Simulation End Time
set SIMTIME 90.0
# Bandwidth of the link 128Kbps
set LINK_BW 64

set fbdelay 1.0
```

```
#Initialize two nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

#
# n1 sends RTP data to n6
# n3 sends RTP data to n5
# n2-n4 is the bottleneck link
#
#
#   n3                n5
#     \              /
#     n2 ========== n4
#     /              \
#   n1                n6
#

#Initialize two RTP Transport Agents
set rtptran1 [new Agent/IVI_RTP]
set rtptran6 [new Agent/IVI_RTP]

set rtptran3 [new Agent/IVI_RTP]
set rtptran5 [new Agent/IVI_RTP]



$rtptran1 set iD_ 10
$rtptran6 set iD_ 60

$rtptran3 set iD_ 30
$rtptran5 set iD_ 50

$rtptran1 set user_rtcp_fdbk_rate_ $fbdelay
$rtptran6 set user_rtcp_fdbk_rate_ $fbdelay

$rtptran3 set user_rtcp_fdbk_rate_ $fbdelay
$rtptran5 set user_rtcp_fdbk_rate_ $fbdelay


$rtptran1 set ExtendedRRInfo_ 1
$rtptran6 set ExtendedRRInfo_ 0

$rtptran3 set ExtendedRRInfo_ 1
$rtptran5 set ExtendedRRInfo_ 0

#links

#sender
$ns duplex-link $n1 $n2 10Mb 20ms DropTail
$ns duplex-link $n3 $n2 10Mb 20ms DropTail

#bottleneck link
$ns duplex-link $n2 $n4 1Mb 160ms DropTail

#receiver
$ns duplex-link $n4 $n6 10Mb 20ms DropTail
$ns duplex-link $n4 $n5 10Mb 20ms DropTail

$ns queue-limit $n2 $n4 200

#Attaching the agents to the nodes
$ns attach-agent $n1 $rtptran1
$ns attach-agent $n6 $rtptran6

$ns attach-agent $n3 $rtptran3
$ns attach-agent $n5 $rtptran5

#Creating traffic generators and sink
set rtpgen1 [new Application/RTPTrafficGenerator]
set rtpgen6 [new Application/RTPTrafficGenerator]
$rtpgen1 set iD_ 100
$rtpgen6 set iD_ 600

set rtpgen3 [new Application/RTPTrafficGenerator]
set rtpgen5 [new Application/RTPTrafficGenerator]
$rtpgen3 set iD_ 300
$rtpgen5 set iD_ 500



#Attaching transport agents to the application
$rtpgen1 attach-transport-agent $rtptran1
$rtpgen6 attach-transport-agent $rtptran6

$rtpgen3 attach-transport-agent $rtptran3
$rtpgen5 attach-transport-agent $rtptran5
```

```
#Attaching applications to the transport agent
$rtptran1 attach-application $rtpgen1
$rtptran6 attach-application $rtpgen6

$rtptran3 attach-application $rtpgen3
$rtptran5 attach-application $rtpgen5


$rtptran1 generate-rtcp $LINK_BW
$rtptran6 generate-rtcp $LINK_BW

$rtptran3 generate-rtcp $LINK_BW
$rtptran5 generate-rtcp $LINK_BW



set interface "REAL"
set ipaddr "130.233.x.x"
#"80.221.x.x"
set port1 "5000"
set port6 "7000"

set port3 "5010"
set port5 "7010"

$rtpgen1 attach-interface $interface $ipaddr $port1
$rtpgen6 attach-interface $interface $ipaddr $port6

$rtpgen3 attach-interface $interface $ipaddr $port3
$rtpgen5 attach-interface $interface $ipaddr $port5



#Connecting the agents
$ns connect $rtptran1 $rtptran6
$ns connect $rtptran6 $rtptran1

$ns connect $rtptran3 $rtptran5
$ns connect $rtptran5 $rtptran3

$ns at 0 "$rtpgen1 generate"
$ns at 0 "$rtpgen6 generate"

$ns at 0 "$rtpgen3 generate"
$ns at 0 "$rtpgen5 generate"

$ns at $SIMTIME "$rtpgen1 stop"
$ns at $SIMTIME "$rtpgen6 stop"

$ns at $SIMTIME "$rtpgen3 stop"
$ns at $SIMTIME "$rtpgen5 stop"

#$ns at $SIMTIME+1 "finish"
$ns at $SIMTIME+2 "$ns halt"


#Run the simulation
$ns run

$rtptran1 delete
$rtptran2 delete
$rtpgen1 delete
$rtpgen2 delete

exit 0

####################################
```

RTP source competes with FTP/TCP and CBR/UDP

```
############NS2 RTP Simulation#########################
set ns [new Simulator]
# Simulation start Time
set SimStart 0.0
# Simulation End Time
set SIMTIME 90.0
# Bandwidth of the link 128Kbps
set LINK_BW 64

$ns color 1 Blue
$ns color 2 Red
$ns color 3 Green

#Open the nam trace file
set nf [open out_200907_1.nam w]
$ns namtrace-all $nf

#set f0 [open out0.tr w]
set f3 [open out3_200907_1.tr w]
```

```
set f7 [open out7_200907_1.tr w]

#Define a 'finish' procedure
proc finish {} {
        global ns nf

global f3 f7
#Close the output files
close $f3
close $f7
#Call xgraph to display the results
exec xgraph out3_200907_1.tr out7_200907_1.tr -geometry 800x400 &

        $ns flush-trace
#Close the trace file
        close $nf
#Execute nam on the trace file
        exec nam out_200907_1.nam &
        exit 0
}

#Define a procedure which periodically records the bandwidth received by the
#traffic sinks
proc record {} {
        global sinkudp sinktcp f3 f7

#Get an instance of the simulator
set ns [Simulator instance]

#Set the time after which the procedure should be called again
        set time 0.5

#How many bytes have been received by the traffic sinks?
        set bw7 [$sinkudp set bytes_]
        set bw3 [$sinktcp set bytes_]

#Get the current time
        set now [$ns now]

#Calculate the bandwidth (in KBit/s) and write it to the files
        puts $f3 "$now [expr $bw3/$time*8/1000]"
        puts $f7 "$now [expr $bw7/$time*8/1000]"

#Reset the bytes_ values on the traffic sinks
        $sinktcp set bytes_ 0
        $sinkudp set bytes_ 0

#Re-schedule the procedure
        $ns at [expr $now+$time] "record"
}


#Initialize two nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

set n7 [$ns node]
set n8 [$ns node]
#
# n1 sends RTP data to n6
# n3 sends FTP/TCP data to n5
# n7 sends UDP/CBR data to n8
# n2-n4 is the bottleneck link
#
#
#
#   n3 n7           n8 n5
#     \|            |/
#     n2 ========== n4
#     /             \
#   n1               n6
#
#Initialize the RTP Transport Agents (1 source and destination)
set rtptran1 [new Agent/IVI_RTP]
set rtptran6 [new Agent/IVI_RTP]

$rtptran1 set iD_ 10
$rtptran6 set iD_ 60
$rtptran1 set user_rtcp_fdbk_rate_ 1.0
$rtptran6 set user_rtcp_fdbk_rate_ 1.0


$rtptran1 set ExtendedRRInfo_ 1
$rtptran6 set ExtendedRRInfo_ 0
#links

#sender
```

```
$ns duplex-link $n1 $n2 10Mb 20ms DropTail
$ns duplex-link $n3 $n2 10Mb 20ms DropTail
$ns duplex-link $n7 $n2 10Mb 20ms DropTail

#bottleneck link
$ns duplex-link $n2 $n4 1Mb 160ms DropTail

#receiver
$ns duplex-link $n4 $n6 10Mb 20ms DropTail
$ns duplex-link $n4 $n5 10Mb 20ms DropTail
$ns duplex-link $n4 $n8 10Mb 20ms DropTail

$ns queue-limit $n2 $n4 150
$ns queue-limit $n4 $n2 150


#Monitor the queue for the link between node 2 and node 3
$ns duplex-link-op $n2 $n4 queuePos 0.5

#Attaching the agents to the nodes
$ns attach-agent $n1 $rtptran1
$ns attach-agent $n6 $rtptran6
#Creating traffic generators and sink
set rtpgen1 [new Application/RTPTrafficGenerator]
set rtpgen6 [new Application/RTPTrafficGenerator]
$rtpgen1 set iD_ 100
$rtpgen6 set iD_ 600
#$rtpgen1 set ModifyTmmbrRequest_ 1
#Application/RTPTrafficGenerator set ModifyTmmbrRequest_ 1

#Attaching transport agents to the application
$rtpgen1 attach-transport-agent $rtptran1
$rtpgen6 attach-transport-agent $rtptran6

#Attaching applications to the transport agent
$rtptran1 attach-application $rtpgen1
$rtptran6 attach-application $rtpgen6

$rtptran1 generate-rtcp $LINK_BW
$rtptran6 generate-rtcp $LINK_BW


#-----------------------------------------------------------------------
#REAL interface type working: Here the packets are transfered similar
# to the SOCKET interface, but the timing rules and packet exchange
# rules are specified in the Ns2-Codec-Interface document
#-----------------------------------------------------------------------
#interface details --end

set interface "REAL"
set ipaddr "130.233.x.x"
#"80.221.x.x"
set port1 "5000"
set port6 "7000"

$rtpgen1 attach-interface $interface $ipaddr $port1
$rtpgen6 attach-interface $interface $ipaddr $port6



#Set TCP connection

#TCP SRC
set tcp0 [new Agent/TCP]
$tcp0 set class_ 2
$ns attach-agent $n3 $tcp0

#TCP SINK
set sinktcp [new Agent/TCPSink]
$ns attach-agent $n5 $sinktcp
$ns connect $tcp0 $sinktcp
$tcp0 set fid_ 1

#attaching FTP source to TCP SRC
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP

#Set UDP connection
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$ns attach-agent $n7 $udp0


#Create a Null agent (a traffic sink) and attach it to node n3
set sinkudp [new Agent/LossMonitor]
$ns attach-agent $n8 $sinkudp
$ns connect $udp0 $sinkudp
$udp0 set fid_ 2

# Create a CBR traffic source and attach it to UDP
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 attach-agent $udp0
$cbr0 set type_ CBR
$cbr0 set packet_size_ 250
$cbr0 set interval_ 0.01
$cbr0 set random_ true


#Start logging the received bandwidth
$ns at 0.0 "record"
#Start the traffic sources
$ns at 1.0 "$cbr0 start"
$ns at 3.0 "$ftp0 start"

$ns at 80.0 "$cbr0 stop"
$ns at 85.0 "$ftp0 stop"


#Connecting the agents
$ns connect $rtptran1 $rtptran6
$ns connect $rtptran6 $rtptran1
$ns at 0 "$rtpgen1 generate"
$ns at 0 "$rtpgen6 generate"
$ns at $SIMTIME "$rtpgen1 stop"
$ns at $SIMTIME "$rtpgen6 stop"
$ns at $SIMTIME+1 "finish"

puts "CBR: [$cbr0 set packet_size_] [$cbr0 set interval_]"


$ns at $SIMTIME+2 "$ns halt"
#Run the simulation
$ns run

$rtptran1 delete
$rtptran6 delete
$rtpgen1 delete
$rtpgen6 delete

exit 0

#########################################################################
```