Department of Communication And Networking

Protocols and Algorithms for Adaptive Multimedia Systems

Varun Singh

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 73/2015

Protocols and Algorithms for Adaptive Multimedia Systems

Varun Singh

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall S4 of the school on 02 June 2015 at 1300h.

Aalto University
School of Electrical Engineering
Communication and Networking Department
Networking Technology / Tietoverkkotekniikka

Supervising professor
Prof. Jörg Ott (Aalto University)

Preliminary examiners
Dr. Jörg Widmer, Professor, Institute IMDEA Networks, Madrid, Spain
Dr.-Ing. Carsten Griwodz, Professor, University of Oslo, Oslo, Norway

Opponent
Dr. Michael Welzl, Professor, University of Oslo, Oslo, Norway

Aalto University publication series
DOCTORAL DISSERTATIONS 73/2015

ISBN 978-952-60-6220-4 (printed)
ISBN 978-952-60-6221-1 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-6221-1

Unigrafia Oy
Helsinki 2015

Finland

NORDIC ECOLABEL
441 697
Printed matter

A? Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author	Varun Singh
Name of the doctoral dissertation	Protocols and Algorithms for Adaptive Multimedia Systems
Publisher	School of Electrical Engineering
Unit	Department of Communication and Networking
Series	Aalto University publication series DOCTORAL DISSERTATIONS 73/2015
Field of research	Networking Technology / Tietoverkkotekniikka
Manuscript submitted 31 March 2014	**Date of the defence** 2 June 2015
Permission to publish granted (date) 31 July 2014	**Language** English
☐ **Monograph**	☒ **Article dissertation (summary + original articles)**

Abstract

The deployment of WebRTC and telepresence systems is going to start a wide-scale adoption of high quality real-time communication. Delivering high quality video usually corresponds to an increase in required network capacity and also requires an assurance of network stability. A real-time multimedia application that uses the Real-time Transport Protocol (RTP) over UDP needs to implement congestion control since UDP does not implement any such mechanism. This thesis is about enabling congestion control for real-time communication, and deploying it on the public Internet containing a mixture of wired and wireless links.

A congestion control algorithm relies on congestion cues, such as RTT and loss. Hence, in this thesis, we first propose a framework for classifying congestion cues. We classify the congestion cues as a combination of: where they are measured or observed? And, how is the sending endpoint notified? For each there are two options, i.e., the cues are either observed and reported by an in-path or by an off-path source, and, the cue is either reported in-band or out-of-band, which results in four combinations. Hence, the framework provides options to look at congestion cues beyond those reported by the receiver.

We propose a sender-driven, a receiver-driven and a hybrid congestion control algorithm. The hybrid algorithm relies on both the sender and receiver co-operating to perform congestion control. Lastly, we compare the performance of these different algorithms. We also explore the idea of using capacity notifications from middleboxes (e.g., 3G/LTE base stations) along the path as cues for a congestion control algorithm. Further, we look at the interaction between error-resilience mechanisms and show that FEC can be used in a congestion control algorithm for probing for additional capacity.

We propose Multipath RTP (MPRTP), an extension to RTP, which uses multiple paths for either aggregating capacity or for increasing error-resilience. We show that our proposed scheduling algorithm works in diverse scenarios (e.g., 3G and WLAN, 3G and 3G, etc.) with paths with varying latencies.

Lastly, we propose a network coverage map service (NCMS), which aggregates throughput measurements from mobile users consuming multimedia services. The NCMS sends notifications to its subscribers about the upcoming network conditions, which take these notifications into account when performing congestion control.

In order to test and refine the ideas presented in this thesis, we have implemented most of them in proof-of-concept prototypes, and conducted experiments and simulations to validate our assumptions and gain new insights.

Keywords Multimedia, Congestion Control, Multipath RTP, Network Coverage Maps

ISBN (printed) 978-952-60-6220-4	**ISBN (pdf)** 978-952-60-6221-1	
ISSN-L 1799-4934	**ISSN (printed)** 1799-4934	**ISSN (pdf)** 1799-4942
Location of publisher Helsinki	**Location of printing** Helsinki	**Year** 2015
Pages 220	**urn** http://urn.fi/URN:ISBN:978-952-60-6221-1	

Preface

This dissertation is based on several years of research at Aalto University (earlier Helsinki University of Technology) and standardisation at the Internet Engineering Task Force (IETF). My sincere thanks to all the people who helped me with the research. Foremost, I would like to thank Prof. Jörg Ott for encouraging me to pursue the interaction of multimedia and congestion control in more detail. He is an outstanding guide, and an inspiration. His timely interactions kept spirits high especially when faced with disappointments and for his patience to stay the course.

Igor Curcio and Nokia Research Center, presented the practical problems faced by multimedia in mobile networks; his experience with mobile systems was influential in shaping my early ideas related to congestion control. Colin Perkins, his book on RTP is a practitioners guide – covers aspects of the design which are not discussed in the RFCs. Additionally, the numerous discussions I had with him over the last few years have been very beneficial and enhanced me as a researcher. Lars Eggert encouraged me to participate and contribute to the IETF, which exposed me to various deployment and real-world issues, further, his sound advice at various points during the Ph.D. was instrumental in getting to the end.

During the course of my research, I worked with several colleagues on a day-to-day basis. They were: Sharmistha Chatterjee (2010), Saba Ahsan (2011), Marcin Nagy (2012), and Albert Abello (2013). This experience was tremendously beneficial as I learnt several skills beyond performing research activities.

An equal measure of gratitude is due to all the co-authors of the papers that make up the thesis. They are: Jörg Ott, Igor Curcio, Saba Ahsan, Marcin Nagy, Albert Abello, Colin Perkins, Lars Eggert, Jegadish Devadoss, Martin Ellis, Chenghao Liu, and Ye-Kui Wang.

Equivalently, I would like to thank the co-authors of various standardization documents that complement this thesis: Jörg Ott, Lars Eggert, Colin Perkins, Zahed Sarker, Teemu Karkkainen, Xiaoqing Zhu, Ralf Globisch, Thomas Schierl, Mo Zanaty, Rachel Huang, Michael Ramalho, Qin Wu, and Harald Alvestrand.

The pre-examiners for this thesis were Professors Carsten Griwodz and Jörg Widmer. I want to thank them both for their efforts in reviewing the thesis. Their comments and suggestions helped improve the quality of the thesis.

Extremely grateful to the staff at the doctoral school and Department of Communication and Networking for the assistance in all ancillary tasks: Sari Kivelö, Sanna Patana, Heli Liukko, Kati Voutilainen, Anita Bisi.

Special appreciation to my friends: László, Mário, Cathy, Markus, Jakub, Tatu, Judit, José-Luis, Tuomo, and Pavan for the help and support during the years of research. I thank Irena for reinforcing that the benefits outweigh the risk when following your dreams. An extended gratitude to the folks at Aalto Entrepreneurship Society: Kristo, Markus, Tuomo, Krista, Natalie, Charlotta – you rock! To my family for the sacrifice they have patiently made.

Finally, this work has been supported by the European Community under its Seventh Framework Program, the European Institute of Innovation and Technology (EIT) ICT Labs, the Department of Computer Science, the Future Internet Graduate School (FIGS), and the Nokia Foundation.

Espoo, July 31, 2014.

Varun Singh

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I V. Singh, S. McQuistin, M. Ellis, C. Perkins, “Circuit Breakers for Multimedia Congestion Control,” in *Proceedings of IEEE International Packet Video Workshop*, San Jose, USA, Dec, 2013, Pages 1–8.

II V.Singh, J. Ott, I.D.D. Curcio, “Rate adaptation for 3G Conversational Video,” in *Proceedings of IEEE Infocom Workshops*, Rio de Janeiro, Brazil, Apr, 2009, Pages 1–7.

III V. Singh, J. Ott, I.D.D. Curcio, “Rate Adaption for Conversational Video in Heterogeneous Environments,” in *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia (WoWMoM)*, San Francisco, USA, Jun, 2012, Pages 1–7.

IV V. Singh, A. A. Lozano, J. Ott, “Performance Analysis of Receive-Side Real-Time Congestion Control for WebRTC,” in *Proceedings of IEEE International Packet Video Workshop*, San Jose, USA, Dec, 2013, Pages 1–8.

V J. Devadoss, V. Singh, C. Liu, J. Ott, Y-K. Wang, I.D.D. Curcio, “Evaluation of Error Resilience Mechanisms for 3G Conversational Video,” in *Proceedings of IEEE International Symposium on Multimedia (ISM)*, Berkeley, USA, Dec, 2008, Pages 378–383.

VI M. Nagy, V. Singh, J. Ott, L. Eggert, "Rate-control using FEC for Interactive Multimedia Communication," *Proceedings of ACM Multimedia Systems (MMSys)*, Singapore, Singapore, March, 2014, Pages 191–202.

VII V. Singh, S. Ahsan, J. Ott, "MPRTP: Multipath Considerations for Real-time Media," in *Proceedings of ACM Multimedia Systems (MMSys)*, Oslo, Norway, Feb, 2013, Pages 190–201.

VIII V. Singh, J. Ott, I.D.D. Curcio, "Predictive Buffering for Streaming Video in 3G Networks," in *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia (WoWMoM)*, San Francisco, USA, Jun, 2012, Pages 1–10.

Author's Contribution

Publication I: "Circuit Breakers for Multimedia Congestion Control"

The author of this dissertation was the main author of this paper and co-edited the paper with Colin Perkins. He equally contributed to the ideas and concepts discussed in the paper. He also implemented the algorithm, conducted and analysed experiments for the interactive video scenario.

Publication II: "Rate adaptation for 3G Conversational Video"

The author of this dissertation was the main author of the paper. He equally contributed to the ideas and concepts discussed in the paper, and also implemented the algorithm and the overall system, conducted the experiments and analysed the results.

Publication III: "Rate Adaption for Conversational Video in Heterogeneous Environments"

The author of this dissertation was the main author of the paper. He equally contributed to the ideas and concepts discussed in the paper, and implemented the algorithm and the overall system, conducted the experiments and analysed the results.

Publication IV: "Performance Analysis of Receive-Side Real-Time Congestion Control for WebRTC"

The author of this dissertation was the main author of this paper. His contribution consisted of designing the experiments and analysing the results discussed in the paper.

Publication V: "Evaluation of Error Resilience Mechanisms for 3G Conversational Video"

The author of this dissertation was one of the co-authors of the paper. His contribution consisted of providing the 2 out of the 4 ideas (NACK and Slice-size adaptation) discussed in the paper, implementing the corresponding algorithms, conducting the experiments and co-editing the paper with the main author of the paper (Jegadish Devadoss).

Publication VI: "Rate-control using FEC for Interactive Multimedia Communication"

The author of this dissertation was one of the co-authors of the paper. His contribution consisted of discussing the main idea and concept for the paper with the lead author, designing the experiments, implementing the comparative algorithms and providing the corresponding results, and co-editing the paper with the main author of the paper (Marcin Nagy).

Publication VII: "MPRTP: Multipath Considerations for Real-time Media"

The author of this dissertation was the main author of the paper. His contribution consisted of providing the main idea for the paper, providing the configurations for the experiments, performing the statistical analysis, and acting as the lead editor of the paper.

Publication VIII: "Predictive Buffering for Streaming Video in 3G Networks"

The author of this dissertation was the main author of the paper. He equally contributed to the ideas and concepts discussed in the paper, he implemented the algorithm and the overall system, conducted the experiments and analysed the results.

List of Figures

List of Tables

List of Abbreviations

3G	Third Generation
3GPP	3rd Generation Partnership Project (GSM, 3G, IMS, HSPA, LTE)
ABU	Average Bandwidth Utilisation
ACM	Association for Computing Machinery
ADSL	Asymmetric Digital Subscriber Line
ALM	Application Layer Multicast
AMR	Adaptive Multi-Rate
ANDSF	Access Network Discovery and Selection Function
API	Application Programming Interface
AQM	Active Queue Management
AVC	Advanced Video Coding
AVP	Audio-Visual Profile
AVPF	Audio-Visual Profile with Feedback
BW	Bandwidth
CB	Circuit Breaker
CBR	Constant Bit Rate
CC	Congestion Control
CNADU	Conversational NADU
CNAME	Canonical Name
CODEC	Compressor-Decompressor or Coder-Decoder
CSRC	Contributing Source
DASH	Dynamic Adaptive Streaming over HTTP
DCCP	Datagram Congestion Control Protocol
DLSR	Delay Since Last SR
DSCP	Differentiated Services Code Points
DTLS	Datagram TLS
DW	Drift Window
DiffServ	Differentiated Services

ECN	Explicit Congestion Control
EDGE	Enhanced Data Rates for GSM Evolution
ETSI	European Telecommunications Standards Institute
FBRA	FEC-based Rate Adaptation
FEC	Forward Error Correction
FIR	Full Intra Request
FMO	Flexible Macroblock Ordering
FPS	Frames Per Second
FW	Firewall
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HSDPA	High Speed Downlink Packet Access
HSN	Highest Sequence Number
HSPA	High Speed Packet Access
HTML5	Hyper Text Markup Language version 5
HTTP	Hypertext Transfer Protocol
IAT	Inter-Arrival Time
ICE	Interactive Connectivity Establishment
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	Internet Protocol (IPv4, IPv6)
IPTV	Internet Protocol Television
ISP	Internet Service Provider (broadband and/or mobile)
ITU	International Telegraph Union
ITU-T	ITU Telecommunication Standardisation Sector
IntServ	Integrated Services
JPEG	Joint Photographic Experts Group
K-NN	K-Nearest Neighbour
LAN	Local Area Network
LISP	Locator/ID Separation Protocol
LSR	Last SR
LTE	Long Term Evolution
MAC	Media Access Control
MCU	Multipoint Control Unit
MIKEY	Multimedia Internet KEYing
MPEG	Moving Picture Experts Group
MPRTCP	Multipath RTCP

MPRTP	Multipath RTP
MPTCP	Multipath TCP
MSTFP	Multimedia Streaming TCP-Friendly Protocol
MTSI	Media Telephony Service for IMS
MTU	Maximum Transmission Unit
NACK	Negative Acknowledgement
NADA	Network-Assisted Dynamic Adaptation
NADU	Next Application Data Unit
NAT	Network Address Translation
NCMS	Network Coverage Map Service
NTP	Network Time Protocol
NetEm	Network Emulator
O/A	Offer and Answer
OWD	One-Way Delay
P2P	Peer-to-Peer
PCM	Pulse Code Modulation
PCN	Pre-Congestion Notification
PDV	Packet Delay Variation
PEVQ	Perceptual Evaluation of Video Quality
PLI	Packet Loss Indication
PLR	Packet Loss Ratio
PSC	Packet Switched Calls (related to IMS)
PSNR	Peak Signal-to-Noise Ratio
PSS	Packet Switched Streaming (related to IMS)
PT	Payload Type
PtP	Point-to-Point
QoE	Quality of Experience
QoS	Quality of Service
RAP	Rate Adaption Protocol
RED	Random Early Detection
REMB	Receiver Estimated Max Bit rate
RFC	Request for Comments
RGB	Red Green Blue [*colour scheme*]
RLE	Run-Length Encoded
RPS	Reference Picture Selection
RPSI	Reference Picture Selection Indication
RR	(RTCP) Receiver Report
RRTCC	Receiver-side Real-Time Congestion Control

RSVP	Resource Reservation Protocol
RTCP	RTP Control Protocol
RTMFP	Real-Time Media Flow Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
RTT	Round-Trip Time
SBC	Session Border Controller
SCTP	Stream Control Transmission Protocol
SDES	Source Description
SDES	Security Descriptions
SDK	Software Development Kit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLI	Slice Loss Indication
SOHO	Small Office/Home Office
SR	(RTCP) Sender Report
SRTP	Secure RTP
SSA	Slice Size Adaptation
SSRC	Synchronization Source
STUN	Session Traversal Utilities for NAT
SVC	Scalable Video Coding
SW	Skew Window
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
TLS	Transport Layer Security
TMMBN	Temporary Maximum Media Stream Bit Rate Notification
TMMBR	Temporary Maximum Media Stream Bit Rate Request
TURN	Traversal Using Relays around NAT
UDP	User Datagram Protocol
UEP	Unequal Error Protection
ULP	Unequal Level of Protection
VCEG	Video Coding Experts Group (defined: H.26x series of codecs)
VoIP	Voice over IP
VQM	Video Quality Metric
WFQ	Weighted Fair Queuing
WLAN	Wireless LAN
WRED	Weighted Random Early Detection
WebRTC	Web Real-Time Communications

XMPP	Extensible Messaging and Presence Protocol
XR	Extended Reports
YUV	Luma (Y) Chrominance (UV) [*colour scheme*]
ZRTP	Zimmermann RTP

List of Symbols

σ	Standard deviation
L_{avg}	Average value of an array L
$\widetilde{L}$	Median value of an array L
$MAX([L])$	Maximum value of an array L
$MIN([L])$	Minimum value of an array L
$QueueSize_{packets}$	Maximum number of MTU size packets the queue can hold.
$QueueSize_{sec}$	Time to live of a packet in the queue.
$session\_bw$	Session bandwidth
TR_k	Reception timestamp of k^{th} packet
TS_k	RTP timestamp of k^{th} packet

1. Introduction

In recent years, video has emerged as the dominant traffic[1] on the Internet [30, 29], partly due to the success of YouTube and other over-the-top media streaming services (e.g., Netflix, Vimeo, Dailymotion, etc.). Video streaming emerged as the most prominent traffic on the network only after it was easily accessible to Internet users in the web-browser. The initial growth of media streaming is attributed to the Adobe Flash Video plugin, but video streaming became ubiquitous when the video tag was introduced into the HTML5 standard [71] and browsers natively supported rendering media streams. Currently, the same trend is observable for real-time communication; present day web services either use Adobe's Real-time Media Flow Protocol (RTMFP) [140] or their own plugins[2]. As before with media streaming, the community is currently working towards standardising the Web-based Real-Time Communication (WebRTC) stack, which will enable any webservice to provide real-time communication by adding a few lines of code and without requiring the user to install a plugin. Therefore, these forthcoming deployments of WebRTC services are going to kick-start the growth of real-time communication on the Internet.

There are some fundamental differences between media streaming and real-time communication. Media streaming is used in video on demand and IP television (IPTV) services wherein the content is pre-encoded and played back directly from network storage; in streaming the challenge is to be able to simultaneously serve multiple customers (*scaling*) and consistently provide a high-quality multimedia experience. The receiving endpoint in this case is mainly required to avoid pausing the playback midstream, which it does by pre-buffering several seconds of content. A larger pre-buffer not only removes the effect of packet jitter but also helps

[1]In 2012, 51 % of mobile traffic was video.

[2]There are other plugin-based services: Facebook Video based on Skype SDK, Google Talk, Google's Hangout/Helpout services, etc.

in requesting lost packets (via *retransmissions*), therefore using a large playout buffer provides the opportunity to deliver consistent media quality. On the other hand, in real-time communication, the content is not pre-stored and endpoints use small buffers. The small buffering duration preserves interactivity by maintaining as low a delay as possible (limited only by the network delay); hence, endpoints have to modify the sending rate to best match the available capacity and not cause excessive packet delay.

This thesis is about enabling congestion control for unicast real-time communication and deploying it in heterogeneous networking environments containing a mixture of wireless, mobile, and wired links. Nowadays multimedia system run on endpoints that may be connected to the Internet by one or more network interfaces or have more than one IP address, thus enabling the multimedia application to use multiple interfaces to send and/or receive media. We define a congestion control framework and categorise the observed congestion cues, and lastly, propose congestion control algorithms that work in diverse situations. The Real-time Transport Protocol (RTP) [124] is the chosen transport for carrying media in WebRTC [7], and we too use it for our media flows. Further, the congestion control algorithm is built within the design constraints of RTP. This thesis is a bundle of scientific papers that discuss various parts of the framework and this summary puts them in context.

1.1 Multimedia Congestion Control

Interactive real-time media applications use RTP [124] to encapsulate multimedia content. RTP is capable of using Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Datagram Congestion Control Protocol (DCCP) for delivering multimedia. While TCP [107] is extensively used to deliver multimedia via HTTP streaming, it is not very suitable for real-time communication, especially when the path latencies are greater than 100 *ms* [21]; hence, UDP [106] is used to carry real-time multimedia traffic. Since UDP provides no form of congestion control, which is essential for deployment on the Internet, multimedia applications have to implement their own congestion control.

Real-time multimedia communication on the Internet is subject to the unpredictability of the best-effort IP network. The uncertainty is mainly due to packet loss, packet re-ordering, and variable queuing delay. Bufferbloat [61] and drop-tail queues in the router can cause long delays and bursty losses. Video is able to tolerate some amount of loss either by concealing it or using some form of error-resilience technique, but bursty loss causes visual impairments which adversely affect the user's quality of experience [167].

Additionally, in mobile networks the capacity available to a user in a cell varies due to mobility, cell-loading, interference, fading, and handover. Similarly, the presence of cross-traffic in the bottleneck requires the real-time multimedia stream to compete for the available capacity, which may also vary depending on the amount of cross-traffic. Coupling the variability in the available capacity to the intrinsic variability in the captured media (either due to motion or due to voice activity detection), the variability in the size of the frames produced by the video codec (I or P frames), and the responsiveness of the codec to produce the media stream at the requested bit rate makes multimedia congestion control very challenging.

Instead of performing congestion control, the application may reserve capacity for the multimedia traffic. This is often done in IPTV deployments to separate the operators' content from the customer's traffic, thereby guaranteeing good performance for the IPTV media. Similarly, it is possible for other multimedia services to attempt to reserve capacity. There are two ways to do it: IntServ (Integrated Services) [19] and DiffServ (Differentiated Services) [93]. In IntServ, the application requests each QoS-capable[3] router along the end-to-end path to reserve capacity by using the Resource Reservation Protocol (RSVP) [20]. The routers that agree to the reservation, keep tabs on the nature of the expected flow and actively police it. These routers maintain soft-state, which is removed when timeout occurs, but it is difficult to maintain the volume of updates when deployed on the Internet. In contrast, DiffServ was designed not to require setting up the end-to-end reservation beforehand; rather, it relies on the endpoint labelling each packet with a Differentiated Services Code Point (DSCP) [18] before the packet is sent. The labelled packets are then subjected to classification and policing by the intermediate routers [14]. The sending endpoint can only hope that the packets get the appropriate treatment, which cannot be guaranteed once the packet passes from one DiffServ administrative domain to another [46].

In this thesis, we do not rely on the use of DiffServ or RSVP for delivering media traffic. The principal reason is that we would need to implement

[3]Quality of Service.

congestion control anyway if there is insufficient capacity in the service-level agreement for a specific traffic class. Consequently, the multimedia endpoints need to implement congestion control, i.e., the endpoint monitors the media flows for congestion and varies (increases or decreases) the media encoding rate based on the observed congestion cues. Further, we limit the applicability of the congestion control algorithms discussed in this thesis to unicast real-time communication and specifically exclude the use of multicast transmission. The main reason we exclude multicast is because the proposed algorithms rely on feedback from a single endpoint. When multiple endpoints are involved, the congestion control at every source needs a strategy to adapt the transmission rate for each receiver or a sub-group of receivers, which is a study of its own, with extensive studies done in the past [142, 152, 114, 67].

1.2 Research Methodology

This thesis aimed to produce original scientific work that is widely applicable in the Internet community. The Association for Computing Machinery (ACM) defines several cultural styles for conducting scientific research [40], and our methodology falls into the *abstraction* and *design* paradigms. In the *abstraction* paradigm, the researcher iterates through "modelling" or "experimentation" to construct a model and make a prediction, then designs an experiment and collects data, and finally analyses the results. The *design* paradigm is related to engineering and consists of the following steps: state the requirements, write a specification, build and test the system. Our research covers all of these aspects; the results that make up the core of the thesis were implemented as simulations, proof-of-concept prototypes and in test-beds.

In order to make significant impact in the Internet community, researchers not only have to produce noteworthy results to motivate deployment but also solve engineering issues. These engineering solutions are typically described in standards documents, which facilitate interoperability and motivates deployment. In our research, wherever possible, we have contributed to the relevant standards body. To summarise, this thesis is made up of both our research work and our standards work.

1.3 Contributions

The main contributions of this dissertation are:

- A mechanism to implement a rudimentary congestion control (circuit-breaker) that aborts communication when it encounters congestion. By implementing such a mechanism, the endpoints limit the impact of a non-adaptive media flow on other elastic traffic.
- A study on implementing the congestion controller for real-time media communication at the sender, receiver, or both. Additionally, we look at the possibility of reacting to congestion cues sent by the network elements on the media path. We also evaluate the performance of the congestion control currently implemented by the Chrome web-browser.
- Applicability of an error-resilience scheme from a suite of error-resilience mechanisms based on latency and loss rate. Consequently, we also propose using Forward Error Correction (FEC) to perform congestion control instead of just using it for error resilience.
- A mechanism to use multiple interfaces to send and receive real-time multimedia. We also propose a scheduling and an adaptive playout algorithm that takes into account the variability in path characteristics across diverse paths.
- A mechanism to create coverage maps, i.e., associate throughput to a geolocation so that endpoints detect areas of good and poor coverage and adapt their sending rate to best fit the network conditions.

1.4 Summary of the Publications

This thesis consists of an introductory part and eight original publications. In Publication I, we propose a set of circuit-breaker conditions which are applied to non-adaptive media flows. At the moment, these media flows do not implement congestion control and, if deployed on the Internet, are expected to cause congestion. The circuit breaker triggers when the application appears to be causing congestion.

Publication II, Publication III, Publication IV, and Publication VI discuss congestion control for interactive multimedia communication. The congestion control algorithms proposed in Publication II were developed for a mobile environment. We additionally discuss three types of congestion control: *sender-driven*, in which the sender decides the new sending rate; *receiver-driven*, in which the receiver decides the new sending rate; and *network-assisted*, in which the network notifies the endpoints about the available rate (for e.g., by an on-path device, or a 3G base station). Publication III extends the sender-driven algorithm described in Publication II for deployment in heterogeneous environments. Publication IV evaluates the performance of Google's congestion control algorithm proposed for WebRTC and comments on its deployability on the Internet.

Publication V discusses error resilience for interactive multimedia communication in a mobile (3G) environment. In this paper, we experiment with using different types of error resilience schemes, namely, Negative Acknowledgement (NACK) or Packet Loss Indication (PLI), Forward Error Correction (FEC) or Unequal Level of Protection (ULP), adaptive video slice sizes, and Reference Pictures Selection Indication (RPSI). Lastly, it discusses the applicability of these schemes based on observed packet loss ratio and network latency.

In Publication VI, we propose unifying the concept of error resilience and congestion control. This new congestion control algorithm uses FEC to probe for available capacity and is aimed to replace the two separate algorithms currently implemented by existing interactive multimedia applications (e.g., Skype, Google Hangout, FaceTime). We compare the performance of the use of FEC for Congestion Control with the algorithms discussed in Publication II, Publication III, and Publication IV. The paper also discusses the interaction between the multimedia application, the RTP stack and the codec for implementing congestion control.

In Publication VII, we enable multi-homing for real-time flows and extend the capability of the current RTP system to send media over multiple paths. In this paper, we propose a scheduling algorithm for Multipath RTP (MPRTP) that sends media over paths with widely different path characteristics and also propose a dejitter buffer algorithm that plays out packets smoothly when the path latency between the paths (*skew*) is large. The paper also discusses system- and implementation-related issues involved in deploying MPRTP.

In Publication VIII, we propose a system to enable network-assisted congestion control for mobile clients by building network coverage maps (mainly, measuring throughput). This paper builds on the initial results presented in Publication II, where the middleboxes in the media path assist in congestion control. However, in Publication VIII, mobile clients report their media throughput and geolocation to a third-party service called “network coverage map service”, which collects, stores, and summarises this information for each geolocation. The mobile clients query the coverage map service for available capacity at future geolocations and make appropriate congestion control decisions.

1.5 Structure of the Thesis

This thesis describes techniques to modify the sending rate in response to changing network characteristics in different types of multimedia systems. The work is mainly a summary of scientific papers, but is also supported by an additional body of work. We have co-authored a number of Internet Drafts[4] that complement the scientific results discussed in the thesis. The chapters describing the various parts of the congestion control framework discuss both our scientific and engineering work, while associating it with the relevant related work in the area. The remainder of the thesis is organised as follows.

Chapter 2 provides the necessary background information on the Real-time Transport Protocol (RTP). RTP together with the RTP Control Protocol (RTCP) forms the control loop that adapts media to the reported path characteristics. This chapter is based on the RTP protocol suite [124, 98, 59, 147, 81] and our contributions to it [96, 33, 132, 128, 129, 9].

Chapter 3 provides a high-level overview of our proposed ‘Congestion Cues Framework’, discusses congestion cues, options for reporting intervals, and criteria for evaluating congestion control. We also discuss the circuit breaker (a minimal congestion control) conditions under which a multimedia stream will be terminated. The circuit breaker is applicable to applications that are about to be deployed on the wide Internet, but do not currently implement congestion control and do not want to cause a congestion collapse. This chapter is based on our contributions, which is documented in [135, 160, 131, 127, 102, 120], and Publication I.

Chapter 4 discusses the mechanisms available for congestion control in interactive multimedia. We consider sender-driven, receiver-driven and co-operative congestion control algorithms. The chapter is based on our contributions, which is documented in [43], [126], Publication II, Publication III, and Publication IV.

[4]at the time of writing this thesis, several of these documents are still in the Internet Draft state, but will be published as RFCs shortly.

Chapter 5 discusses the applicability of error-resilience mechanisms for real-time communication. We also discuss using these error-resilience techniques for congestion control. The chapter is based on our contributions, which is documented in [130, 136], Publication V, and Publication VI.

Chapter 6 discusses using multi-homing for real-time media delivery and introduces Multipath RTP (MPRTP). The chapter is mainly based on our contributions, which is documented in [133, 134, 109, 110, 66, 97], and Publication VII.

Chapter 7 discusses network-assisted congestion cues, i.e., from middleboxes in the media path or from a service providing a map of network coverage (collected via active or passive measurements). The chapter is based on our contributions, which is documented in [44], Publication II, and Publication VIII.

Chapter 8 concludes the thesis, we analyse the proposed congestion control framework, and synthesise a unified congestion control algorithm from the proposals discussed in the thesis.

2. RTP: Real-time Transport Protocol

The Real-time Transport Protocol (RTP) [124] is designed for multimedia telephony (voice-over-IP, video conferencing, telepresence systems), multimedia streaming (video on demand, live streaming), and multimedia broadcast. RTP's design is based on the fundamental principles of *application-layer framing* and *integrated layer processing* [39]. To this end, RTP provides the following mechanisms: source and payload type identification, stream synchronisation, packet loss and re-ordering, and media stream monitoring. RTP utilises the RTP Control Protocol (RTCP) to report the performance of the media stream. Figure 2.1 describes the features provided by RTP and RTCP. The media sender transmits encoded media encapsulated in RTP; in addition it also sends RTCP Sender Reports (SR) to facilitate playback synchronisation of different media streams (typically audio and video). The receiver maintains a dejitter buffer to reorder media packets and play them out as per the timing information encoded in the packet. If a packet is missing the receiver attempts to either recover the lost packet or conceal the error. Lastly, the receiving endpoint reports rough or detailed statistics that enables the media sender to adapt its media encoding rate, change to a better codec, or vary the amount of forward error correction.

Figure 2.2 describes the RTP packet header format. The *synchronization source* (SSRC) assists in determining the source endpoint, typically useful when an endpoint sends multiple media streams that need to be synchronised (e.g., audio and video lip-sync). The *RTP timestamp* assists in playing out the received packets at the appropriate instance of time and recomposing the media frame from RTP packets. The *RTP sequence number* assists in identifying the lost packets and re-ordering packets in the case of out-of-order packet arrival. Lastly, RTP uses the *'payload type'* (PT) to describe the encoding of the media data it is carrying. Consequently,

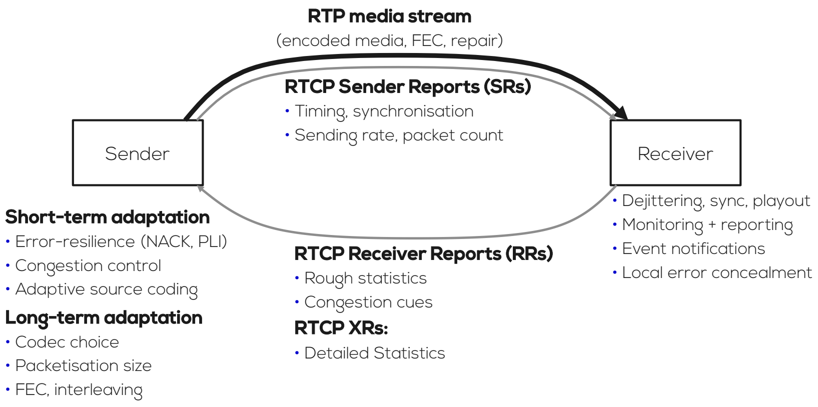


Figure 2.1. RTP and RTCP for adaptive real-time applications.Source: Jörg Ott, "Networked Multimedia Protocols and Systems".

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.2. The RTP packet format that encapsulates the media data.

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
header |V=2|P|    RC   |   PT=SR=200   |             length            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                         SSRC of sender                        |
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
sender |              NTP timestamp, most significant word             |
info   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             NTP timestamp, least significant word             |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                         RTP timestamp                         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                     sender's packet count                     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                      sender's octet count                     |
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
report |                 SSRC_1 (SSRC of first source)                 |
block  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  1    | fraction lost |       cumulative number of packets lost       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |           extended highest sequence number received           |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                      inter-arrival jitter                     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                         last SR (LSR)                         |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                   delay since last SR (DLSR)                  |
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
report |                 SSRC_2 (SSRC of second source)                |
block  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  2    :                               ...                             :
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
       |                  profile-specific extensions                  |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.3. The RTCP packet format for carrying the Sender Report (SR) and the Receiver Report (RR). The SR carries transport statistics and enables stream synchronisation, while the RR carries the receiver transport characteristics.

each codec needs to specify its corresponding payload format.

The receiver measures the incoming streams and reports the coarse-grained transport statistics in an RTCP Receiver Report (RR). The RTCP

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                Payload Format-Specific Header                 |
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               |                                               |
+-+-+-+-+-+-+-+-+                                               +
|                           Media Data                          |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.4. Packet structure of an RTP packet encapsulating the payload-specific header and the associated media data.

RR contains the current loss fraction, jitter, and the highest sequence number received, and it facilitates in calculating the round-trip time (RTT). The sender uses RTCP Sender Reports (SRs) to assist in synchronising the media streams (audio and video) by relating the RTP timestamps of the individual media streams to the wall clock time (NTP) and notifying the receiver about the current packet rate and bit rate. Figure 2.3 shows the RTCP packet header format for a interactive unicast media stream (i.e., both sending and receiving media).

2.1 RTP Payload Formats

The general principle for defining payload formats/types is to identify the encoding of the media packets. These encodings are either codec-specific (e.g., H.264, H.263, H.261, MPEG-2, JPEG, G.711, G.722, AMR, etc.), or generic (e.g., Forward Error Correction (FEC), NACK, multiplexed streams). Typically, a payload document specifies a well-defined packet format for media codecs; it also defines *aggregation rules* for codecs that produce several small frames (e.g., audio) compared to the IP Maximum Transmission Unit (MTU), and *fragmentation rules* for codecs that produce large frames (e.g., I-frames by video codecs). The main reason for fragmenting large frames into smaller packets and not rely on IP fragmentation is that IP fragmented packets are commonly discarded in the network, especially by NATs or firewalls.

2.2 RTP Header Extensions

RTP header extensions carry media-independent information, i.e., data that may be generically applicable to multiple payload formats (e.g., timing information), and needs to be reported more frequently than RTCP reports are emitted. A commonly-cited example is the sending of NACK packets for interactive media, where media flows in both directions and RTP packets are generated every tens of milliseconds. In this case, the RTP header extension can indicate which sequence numbers were correctly received or lost, thereby not completely relying on the RTCP receiver reports to send NACKs or ACKs.

The advantage of using header extensions is that they are backwards compatible, i.e., an endpoint that does not understand them is able ignore them. Some current use-cases for RTP header extensions include reporting the network send timestamp: instead of bursting packets from a large frame on to the network, the sender paces these packets. Another example is equalising a client's audio levels across multiple streams in a media conference. Lastly, RTP header extensions are generic, there is no need to redefine the same extension for each media codec.

2.3 RTCP Reporting Interval

A closed control loop is formed by sending RTP media packets and receiving RTCP feedback packets. The RTCP feedback interval is typically limited to a small fraction of the session bandwidth (*session_bw*) as not to affect the media traffic. The RTCP reporting interval is determined by the number of SSRCs in the session (denoting the session size), and the chosen session bandwidth. The session bandwidth (*session_bw*) is expected to be divided amongst the participants, but oftentimes it is calculated as the sum of the average throughput of the senders expected to be concurrently active. In the case of an audio conference, the session bandwidth would be one sender's bandwidth, but for a video conference, the session bandwidth would vary depending on the number of participants displayed on the user interface. Consequently, the session bandwidth is supplied by the session management layer so that the same value for the RTCP interval is calculated for each participant.

The recommended fraction of the session bandwidth allocated for control traffic is 5 %. For many scenarios, including large conferences, where there are a large number of receivers but a small number of senders, it is recommended that a quarter of the reporting bandwidth (*rtcp_bw*) be shared equally by the senders and the remaining three-quarters by the receivers. The main reason for this allocation ratio is to allow newly-joining participants to quickly receive the CNAME and synchronisation timestamps from the Sender Reports (SRs). For new participants (even if they are just receivers), the RTCP interval is halved to quickly declare their presence. Lastly, the recommended value for a fixed minimum RTCP interval is 5 seconds, while the value for a reduced minimum is $\frac{360}{session\_bw}s$. The fixed minimum RTCP interval of *5 s* is suitable for unidirectional links or for sessions that do not require monitoring of the reception quality statistics (e.g., IPTV), while the reduced minimum RTCP interval is also suitable for participants in a unicast bidirectional multimedia session. The reduced minimum RTCP interval is suitable for sending timely feedback messages to either perform congestion control or error repair; the interval is shorter than *5 s* for session bandwidths greater than *72 kbps*.

If an endpoint detects packet loss or the onset of congestion midway through a reporting interval, the base RTP specification [124] (AVP profile) does not allow the RTCP reports to be sent early and the endpoint has to wait for the next scheduled RTCP report. In this case, the slow control loop causes instability and oscillation in the media bit rate. To overcome this shortcoming, endpoints implement the Extended RTP Profile for RTCP-Based Feedback (AVPF profile) [98], an extension to RTP's default timing rules, to enable rapid feedback. This profile allows the endpoint to adjust the RTCP reporting interval to send the RTCP feedback reports earlier than the next scheduled RTCP report, sometimes even immediately, as long as the reporting interval on average remains the same. Figure 2.5 shows that with AVP profile, the endpoint reports at regular intervals, whereas with AVPF the endpoint it gets the opportunity to send feedback early in every other reporting interval. Along with the possibility of providing timely feedback, the AVPF profile also defines a suite of error-resilience feedback messages, namely, Negative Acknowledgement (NACK), Picture Loss Indication (PLI), Slice Loss Indication (SLI), and the Reference Picture Selection Indication (RPSI).

2.4 RTCP Extended Reports (XRs) for Performance Monitoring

Endpoints use RTCP Extended Reports (XRs) [59, 129] to describe complex metrics that are not exposed by the RTCP Receiver Report (RR). Some examples of XRs relevant to performance monitoring and congestion control are: dejitter buffer metrics [33], Packet Delay Variation (PDV) [34], delay metrics [31], burst-gap discard [32], burst-gap loss [36], Run-Length Encoded (RLE) loss [59], discard RLE [96], the number of discarded pack-

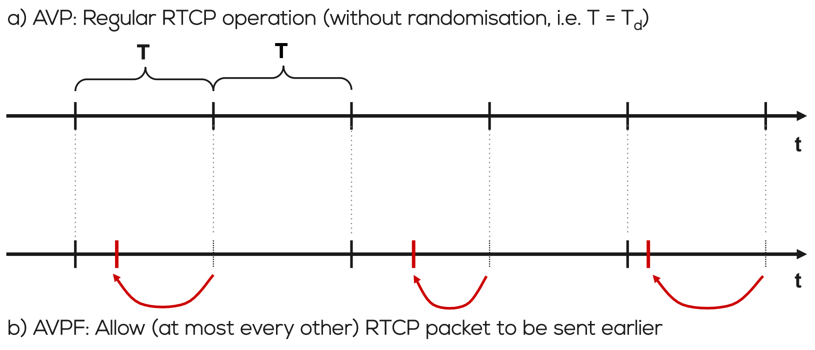

a) AVP: Regular RTCP operation (without randomisation, i.e. T = T_d)
T
T
t
t
b) AVPF: Allow (at most every other) RTCP packet to be sent earlier

Figure 2.5. The RTCP reporting interval as defined in a) AVP, b) AVPF.

ets [38] and bytes [132], post-repair loss count [128], summary statistics [168], Quality of Experience (QoE) [35], and loss concealment [37], etc. RTP allows for new metrics to be defined; the main requirement is to document what is measured, how it is measured and how it is reported to the other endpoints.

2.5 Codec Control Messages

Sometimes an endpoint needs to configure or notify the other endpoint's codec. These messages are broadly classified as *Transport Layer* and *Payload-specific* feedback messages [98, 147]. The transport layer messages are: Temporary Maximum Media Stream Bit Rate Request (TMMBR) and Temporary Maximum Media Stream Bit Rate Notification (TMMBN). The receiving endpoint uses the TMMBR message to configure the maximum encoding bit rate of the media stream, while the sending endpoint uses the TMMBN to inform the receiver of the updated bit rate. Therefore, transport layer feedback messages are intended to transmit general purpose feedback messages, independent of any particular codec or application.

On the other hand, the payload-specific feedback messages carry information specific to a certain payload type and are acted upon by the codec layer. Some examples of these type of messages are: Full Intra Request (FIR), temporal-spatial tradeoff, frame rate, frame size, maximum packet size or packet rate, etc. [149].

2.6 Reduced-Size RTCP Reports

An endpoint sends RTCP feedback as a *compound*, or *minimal*, RTCP packet. A *compound RTCP packet* as defined in [78] contains at least a sender report (SR) or a receiver report (RR) or both, followed by a Source Description (SDES) and any additional XR blocks. A *minimal RTCP packet* is one that contains an SR and/or RR, and is followed by an SDES containing just the canonical name (CNAME)[5]. Hence, every compound RTCP packet is a minimal RTCP packet with additional report blocks. A typical RTCP packet size for interactive multimedia streams is 80 bytes (RTCP=8, SR=20, RR=24, SDES/CNAME =28).

Including any of the additional SDES items or adding XR blocks makes the compound RTCP packet very large. On low bit rate links, these large compound RTCP packets may introduce more delay. Therefore, it may be desirable to logically fragment the report blocks in a compound RTCP packet and send them independently. These fragmented report blocks are called *reduced-size RTCP packets* [81]. Unlike compound RTCP packets, to transmit a reduced-size RTCP packet an endpoint does not need to include the minimal RTCP report. However, when using reduced-size RTCP packets, minimal packets need to be sent once in a while to keep the CNAME-SSRC binding alive.

Reduced-size RTCP reports are beneficial in wireless networks where the packet loss rate increases with the packet size, i.e., larger-sized packets are more susceptible to being dropped compared with smaller-sized packets. Additionally, smaller packets have shorter serialisation time, i.e., the amount of time it takes for the endpoint to put the data packet onto the link is short.

The main reasons for the application to use reduced-size RTCP reports are: 1) to notify the other endpoint of events. Using the signalling channel would incur at least one RTT while implementing it as an RTCP extension would merely incur a one-way delay. 2) to send codec control (e.g., TMMBR) or feedback (e.g., NACK, RPSI) messages. These reduced-size messages are more likely to be transmitted more often and with as little delay as possible, especially since these types of messages are more likely to be sent when link conditions are poor.

[5]The real name (identifier) used to describe the source; it can be in any form desired by the user. Of the SDES items (username, email, phone, geolocation, etc.), it is compulsory to include CNAME in every RTCP packet.

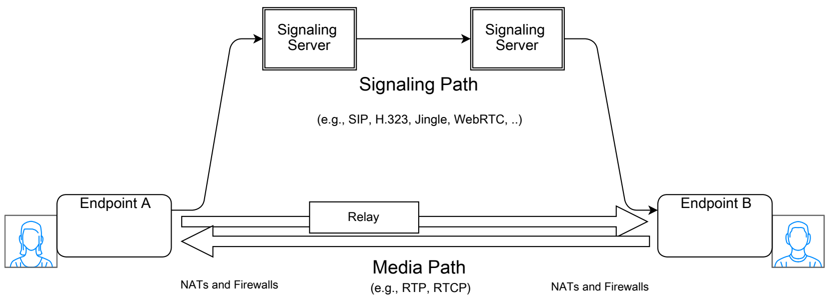


Figure 2.6. The signalling and media paths between two endpoints engaging in a interactive video call.

2.7 Session Setup

There are several ways to set up an interactive or conversational multimedia session, for example by implementing one of the following: H.323 [141], Session Initiation Protocol (SIP) [118], or Jingle [90], an extension to the Extensible Messaging and Presence Protocol (XMPP) [119].

SIP uses the Session Description Protocol (SDP) [69] to describe the endpoint's transport and media capabilities. An SDP description defines a single multimedia session, i.e., an association between a set of participants. It may, however, carry multiple media streams in the session.

The transport details in the SDP are mainly split into two parts: the protocol for delivering media packets (currently, TCP, UDP, or SCTP), and the IP address of the endpoint. The protocol to deliver the media packets is chosen by the application, but identifying the IP address and port of an endpoint is a bit complex. It first requires gathering the endpoint's multiple IP addresses and later exchanging them with the other endpoints to establish connectivity.

Multiple IP addresses arise not only from multiple interfaces but also from the presence of NAT devices in the network, which may change the IP address of the outgoing packets. Since interactive media calls are between endpoints and media streams may eventually traverse a NAT at both ends, in some cases, the only way to deliver media packets between two endpoints would is by using a relay[6] on the public Internet. Hence, the endpoint needs to discover its 2-tuple `[IP address:port]` on the host, which is relatively easy followed by the public address, if behind a NAT [117]. Discovering an endpoint's public address requires contacting a publicly-hosted STUN[7] server and comparing the endpoint's host addresses with the one

[6]Traversal Using Relays around NAT (TURN)
[7]Session Traversal Utilities for NAT (STUN).

observed by the STUN server. Lastly, the endpoint discovers the address allocated by the TURN relay [91] and notifies the other endpoints about its transport details. This collection of 2-tuples is known as *candidates*. First, each endpoint sorts its candidates in decreasing order of priority and exchanges these candidate addresses in the SDP with the other endpoint. On receiving the list of candidates, the endpoints probes between each combination of addresses in the two candidate lists; a pair of addresses is called a *candidate pair*. The endpoint chooses the first candidate pair that successfully establishes connectivity (*aggressive nomination*). This process of performing pair-wise *connectivity checks* is called Interactive Connectivity Establishment (ICE) [115, 116] and it relies on the STUN protocol [117] to establish connectivity across a NAT. In case direct connectivity between the two endpoints fails to be established, the individual endpoints use the *Traversal Using Relays around NAT* (TURN) server and the associated protocol [91] to relay media packets between them. Similar to the STUN server, the TURN server is hosted on the public Internet.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
s=
c=IN IP4 192.0.2.3
t=0 0
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
m=audio 45664 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
a=candidate:2 1 UDP 1694498815 195.148.127.98 45664 typ srflx
    raddr 10.0.1.1 rport 8998
```

Figure 2.7. Sample SDP containing the sender's transport and media capabilities.

The second part of SDP carries the media capabilities, together with the transport parameters, that bind the SDP to the Offer/Answer model. In the O/A model, the sending endpoint *offers* to the receiver a set of media capabilities in decreasing order of preference, typically, multiple options of audio/video codecs and ICE candidates. On receiving the sender's capabilities, the receiver compares its media capabilities with the sender's and responds with the one that best fits the receiver's requirements (*answer*). The *offer is rejected* if the receiver is unable to pick any of the options provided by the sender, or if the ICE connectivity checks fail. Hence, the application at the sender needs to pick a minimum number of widely-available audio and video codecs to avoid negotiation failure. If the *offer is accepted*, both endpoints then know the following: 1) which audio and/or

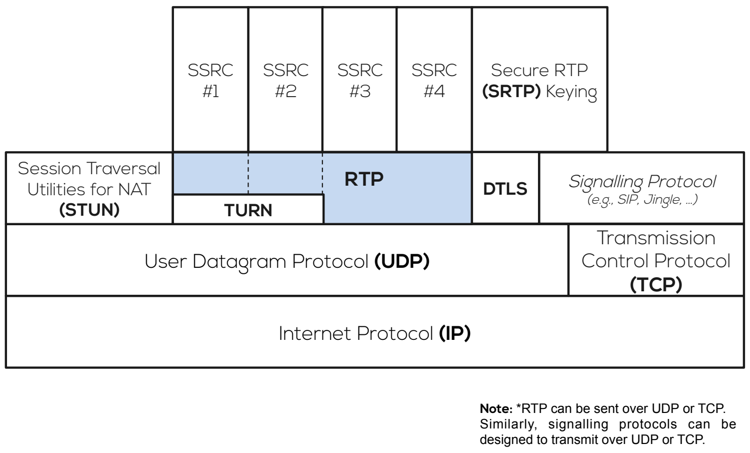

SSRC #1
SSRC #2
SSRC #3
SSRC #4
Secure RTP (SRTP) Keying
Session Traversal Utilities for NAT (STUN)
TURN
RTP
DTLS
Signalling Protocol (e.g., SIP, Jingle, ...)
User Datagram Protocol (UDP)
Transmission Control Protocol (TCP)
Internet Protocol (IP)
Note: *RTP can be sent over UDP or TCP. Similarly, signalling protocols can be designed to transmit over UDP or TCP.

Figure 2.8. Relationship of RTP with STUN, TURN, DTLS and the signalling protocol.

video codecs to use, 2) the payload types of the encoded media streams (possibly even their respective SSRCs), 3) to which IP address and port number to send the media stream, 4) the media session bandwidth, if indicated, and 5) the encryption keys, if encrypting traffic.

Figure 2.8 shows the relationship of the RTP stack with the rest of the IP stack. RTP is usually transmitted over UDP, but under some constrained situations (e.g., restrictive firewalls or NATs), RTP may be encapsulated in TCP [124]. STUN [117] is used by ICE to discover the presence of a NAT device and obtain the mapped (public) IP address. When using a TURN relay server (in the presence of symmetric NATs or when concealing the host address of the caller for privacy reasons), the RTP packets are encapsulated inside the TURN's *ChannelData* message [91]. In Figure 2.8, four media streams (SSRC #1-4) are transmitted by RTP over UDP, but two streams (SSRC #1-2) are relayed through a TURN server. Secure RTP (SRTP) [59] is a security framework that provides confidentiality by encrypting RTP payload (not the RTP headers) and supports source origin authentication. While SRTP is not the only security mechanism for RTP [103], it is widely applicable, especially to voice telephony and group communication. However, the main challenge for SRTP is key management [151], since many options exist (e.g., SRTP over DTLS in WebRTC [51], MIKEY in SIP [11], Security Description in SDP [69], ZRTP [166], etc.).

2.8 Summary

In this chapter we introduced the basic features of RTP and RTCP; we discussed the limits on reporting interval and the numerous RTP and RTCP extensions. The introduction to RTP and RTCP largely provides context and helps understand the design constraints for multimedia congestion control which are discussed in more detail in the forthcoming chapters.

3. Congestion Control Framework for Real-time Communication

In the forthcoming deployment of WebRTC systems, we speculate that high quality[8] video conferencing will see wide-scale adoption. To assure stability of the network (and avoid congestion collapse), these real-time communication systems are required to implement some kind of congestion control for their RTP-based media traffic.

RTP transmits the media data over IP using a variety of transport layer protocols such as UDP, TCP, and Datagram Congestion Control Protocol (DCCP). Consequently, congestion control for RTP-based media flows is implemented either in the application or the media flows are transmitted over congestion-controlled transport (TCP or DCCP). While using a congestion-controlled transport may be safe for the network, it is sub-optimal for the media quality unless the congestion-controlled transport is designed to carry media flows. Unfortunately, TCP is only suitable for interactive multimedia for paths with very low RTT ($<100\,ms$) [21], and DCCP packets have problems with NAT traversal [123] unless DCCP is encapsulated in UDP [105].

This motivates the need for a UDP congestion control algorithm, where the congestion control is implemented between the application and the underlying transport, thereby taking into account both the application's and the transport's requirements or constraints and with appropriate trade-off. In this thesis, we consider congestion control for unicast RTP traffic running over the best-effort IP network.

Endpoints rely on RTCP feedback from the receiver to implement congestion control. Hence, the congestion control should consider the following three aspects in its design: congestion cues to report, block size of each report or the overhead incurred by reporting a cue, and the frequency of these feedback reports. In the following subsections, we describe the in-

[8]normally, high quality corresponds to an increase in required bandwidth.

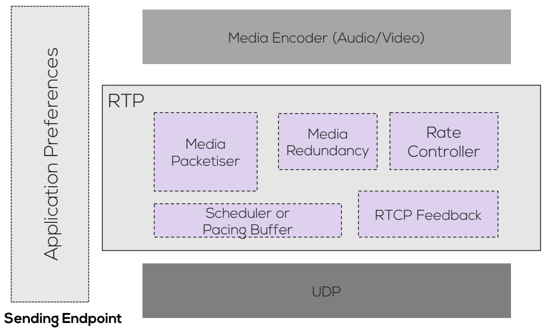
teraction between the application and the congestion control, list common congestion cues, discuss the feedback reporting frequency, the classification of cues, the metrics and criteria to evaluate congestion control proposals, and lastly, we discuss the RTP circuit breaker which stops transmission permanently after observing prolonged congestion.

3.1 Adaptive Multimedia Systems

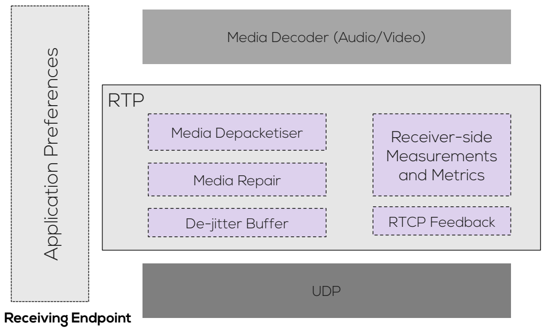
Any real-time communication endpoint is made up of three basic components: codec (encoder and decoder), transport (RTP and UDP) and the application preferences (user and application settings, system policies, capturing and rendering constraints, etc.). The codec encodes and decodes a media stream. A typical application comes with at least one codec each for audio and video. The application may also implement several other media codecs so that it is capable of inter-operating with several different types of endpoints. The transport is mainly made up of RTP, which packetises and depacketises the media and UDP to transmit the media. The application preferences are made up of system polices (which interface to use? which codec to prefer?), codec settings (the preferred or the minimum video resolution, preferred frame rate, etc.). The application preferences may also depend on the outcome of the session establishment, in which the participating endpoints negotiate the codec and network settings.

Figure 3.1 shows a simplified architecture of a sending and receiving endpoint; typically, a real-time multimedia application would contain both these systems and perhaps even two of each for handling audio and video separately. Figure 3.1 (a) depicts the features of the sending-side RTP stack. The *media packetiser* subsystem receives a video frame or a series of audio frames from the codec, which it packetises into RTP. The resulting RTP packets are then passed on to the *media redundancy* subsystem to be cached if a NACK is received or for producing FEC packets. Additionally, the RTP packet size and count is sent to the RTCP subsystem for creating RTCP SR packets. Simultaneously while passing the packet to the media redundancy subsystem, the RTP packets are sent to the *pacing buffer or scheduler* subsystem which transmits the packets in a single burst or trickles them on to the network before the next set of RTP packets are generated by the media packetiser or codec. The sending endpoint also routinely generates and receives RTCP packets.

On receiving an RTCP RR packet, it is sent to the *RTCP feedback* sub-



(a) Sending Endpoint



(b) Receiving Endpoint

Figure 3.1. Typical architecture of a multimedia system. a) sending endpoint, b) receiving endpoint.

system and onwards to the *rate controller* subsystem. The rate controller monitors the congestion cues and based on the congestion control algorithm calculates the new target encoding rate. The codec attempts to compress the media stream to meet the new target encoding rate in a reasonable time frame. Figure 3.1 (b) depicts the features of the receiving-side RTP stack. The received RTP packet is put in a *dejitter buffer* to reassemble out-of-order packets. The dejitter buffer may discard late arriving packets and it also detects packet loss. Furthermore, the packet sequence number, size of the packet, RTP timestamp and the reception timestamp are passed to the *RTCP feedback* subsystem, which routinely generates an RTCP RR packet and may additionally generate requests for retransmitting missing RTP packets. This information is also shared with the *receiver-side measurements and metrics* subsystem, which may generate additional congestion cues to be sent along with the RTCP RR as RTCP extended reports (XRs). If FEC packets or other types of repair packets are received, they are passed on to the *media repair* subsystem which attempts to recover the missing packets in the dejitter buffer. Eventually, the packets in the dejitter buffer are sent to the *media depacketiser* where an audio or video frame is reconstructed and sent to the decoder for playback.

We identify three control loops to implement congestion control [135] based on the interaction between the above components in an multimedia system [160],

1. `Codec-Sender`: The codec adapts its encoding rate based on the feedback from the sender. Unlike elastic traffic, the codec is unable to produce the expected media rate immediately. Therefore, the rate-controller needs to take into account the timeline in which the codec produces the new rate.
2. `Sender-Network`: The sender packetises media frames and sends them over the network to the receiver. The sender may however pace the fragmented frames on to the network instead of sending them in a single burst. It also collects feedback messages from the receiver that may contain congestion cues (i.e., variation in RTT, indication of lost or discarded packets, goodput, jitter, etc.).
3. `Receiver-Sender`: The receiver has a playout buffer of media data waiting to be decoded and rendered, discarding packets that arrive late for playout, and attempting to conceal the missing packets from the observer. The receiver also monitors the media flow for packet losses, variation in jitter, receiver rate, goodput, etc. and reports these to the sender to act upon.

If an endpoint detects congestion rapidly, and the end-to-end path latency is sufficiently low so that this information can be communicated quickly, it is possible to change the encoding rate promptly to meet the variation in the end-to-end path capacity. However, in practice, this is not always possible because **a)** it may take multiple reports or data packets to detect congestion, and **b)** after detecting congestion, it takes at least a one-way delay (OWD) amount of time for the receiver to report it.

3.2 Congestion Cues

Congestion control algorithms rely on cues to detect congestion. These cues are detected either by the sender, receiver, or by an intermediary router. The endpoint observes the congestion along the path and accordingly adapts the sending rate upon receiving the congestion cues. Some common congestion cues are listed below:

- **`Losses`**: occur when intermediate routers drop packets from their queues (*congestion loss*), or due to contention, interference or fading on wireless link (*bit-error loss*). Losses are inferred at the receiving endpoint by gaps in RTP sequence numbers. Typically, a dejitter buffer is used to reorder out of order packets and the fraction packet loss is calculated at the end of each reporting interval.
- **`Discards`**: packets that arrive too late at the receiver to be decoded or played back may be discarded by the receiving endpoint. These late-arriving packets are discarded by the receiver even though they are received because packets with higher *timestamps* have already been passed to the decoder for playback. The fractional loss in the standard RTCP RR does not identify these discarded packets as lost, hence they need to be reported in an RTCP XRs.
- **`Sending rate, receiver rate and goodput`**: are measured at the sender, the receiver and at the decoder, respectively. Typically, the sending rate is the rate at which the media bit stream is generated by the encoder. If packets are lost in the network, the receiver rate is lower than the sending rate. Or if duplicate packets are received, the receiver rate is higher than the sending rate. Lastly, if packets are discarded after arrival or dropped by the decoder, the goodput will be lower than both the sending rate and receiver rate. Hence, goodput represents the actual playback bit rate or the bit rate of the rendered bit stream.
- **`One-way delay (OWD)`**: is a combination of *propagation*, *queuing*, *serialisation* and *processing* delay. Propagation delay is calculated from the ratio of the physical length of the interconnected link and the propagation speed over the specific medium[9]. The serialisation delay is the time taken to send a complete packet on to the communication channel (link) and is a function of the link rate and the packet size. The processing delay is the time taken for the router to determine the next hop or the destination of the packet. Lastly, when multiple packets are received, the router queues them and transmits them one by one. Having large-sized buffers in the router causes *buffer-bloat* [61] and increases the overall one-way delay. However, measuring one-way delay is difficult because the clocks at the endpoints are normally not synchronised; instead, the endpoints rely on RTT measurements for congestion control.
- **`Round trip time (RTT)`**: is the time taken for a packet to go from the sender to the receiver and then back. In RTP, it is calculated with

[9]Usually, propagation speed is a fraction of the speed of light ($0.5c$-$0.8c$).

the collaboration of sending RTCP SRs and receiving RTCP RRs. In interactive multimedia, the media flows in both directions, so the one-way delay (OWD) is approximated as half of the measured RTT. Observing the changes in RTT provides an indication of congestion and smoothing the RTT (averaging over a short interval) protects against over-reacting to the subtle changes in RTT.

- **Packet delay variation and packet inter-arrival time**: packets may arrive at different times due to route changes, or congestion at the bottleneck link causes jitter. Endpoints detect jitter by comparing the send or media generation timestamps with the receiving timestamps. The variation in inter-arrival time may be used to infer congestion.

To pick the right congestion cue, an algorithm developer should consider the following: the type of media stream (audio or video), the expected packet or frame rate, typical MTU size, interdependence of the streams (audio/video sync, multi-view video), whether the congestion controller knows the operating environment (Internet-scale, low-delay local area deployment, heterogeneous environment with a mix of wired and wireless links) and the application requirements (audio preferred over video or vice versa).

Another aspect to consider when picking congestion cues is the the monitoring duration to identify congestion, i.e., either over a *long-term* (order of seconds or minutes) or a *short-term* (order of 100 *ms* or a few seconds). For example, jitter is measured on a per-packet basis, but reported over a longer measurement interval (to filter for noise and transience). In contrast, packet losses, discards, etc. are measured over a shorter interval so that the sender can react to these immediately.

3.3 Congestion Reporting Frequency

Normally, congestion control requires a tight control loop, which means that the receiving endpoint should be able to provide feedback at very short intervals (at least once per RTT). Hence, the design of a congestion control algorithm needs to be aware of the limits on the timing of the feedback. For example, in TCP, the receiver sends an *acknowledgement* packet in response to every packet (or every few packets) it receives, whereas RTCP encourages infrequent feedback and specifies an upper-bound on the fraction of the session media bit rate that the feedback packets can use[10]. [101] discusses three options for the short report intervals,

`Per-packet feedback report`: sends RTCP feedback every time the endpoint receives a packet. For low bit rate media sessions (e.g., audio streams), this would be quite difficult to achieve because the size of the feedback packet would be comparable to the size of the media packet, i.e., the feedback bit rate would larger than the 5 % fraction specified for it. If an endpoint receives packets in a burst or at very short time intervals, the endpoints will not be able to meet the timing requirements for per-packet feedback because the RTCP timing interval calculation has a randomisation factor to avoid synchronising feedback from multiple endpoints.

`Per-frame feedback report`: sends RTCP feedback every time the endpoint receives a complete frame. This is mainly applicable to video where a single video frame would be fragmented into multiple packets because the frame size exceeds MTU size. Typically, an average size of an RTCP packet size in a two-party call is $156\text{-}176$ bytes[11]. For a 30 FPS bidirectional video stream, the $rtcp\_bw \approx 75\,kbps$, which requires the media session bit rate be set to a value higher than $1.5\,Mbps$. Consequently, it would not be possible to perform per-frame for sessions with lower media rates. It should be noted that the requirements for the media session bit rate needs to be re-calculated if the number of participants change, the number of reported blocks change, or the frame rate changes.

`Per-RTT feedback report`: sends RTCP feedback at regular intervals based on the RTT estimate. The requirement for the media session rate would be lower, if the RTT is higher than the frame inter-arrival time. The calculation of the RTCP interval for the per-frame still applies, except that the frame rate is replaced by the RTT estimate.

To summarise, picking longer RTCP feedback intervals requires a lower media session bit rate, hence it increases the possibility of applying the same congestion control to a larger operating area (in terms of session media rates).

[10]The specified feedback rate is 5 % for each multimedia session.

[11]The packet breakdown in bytes is: UDP=16, IPv4=20 or IPv6=40, RTCP=8, SR=20, RR=24, SDES=28, one or more XR blocks is 20 bytes each.

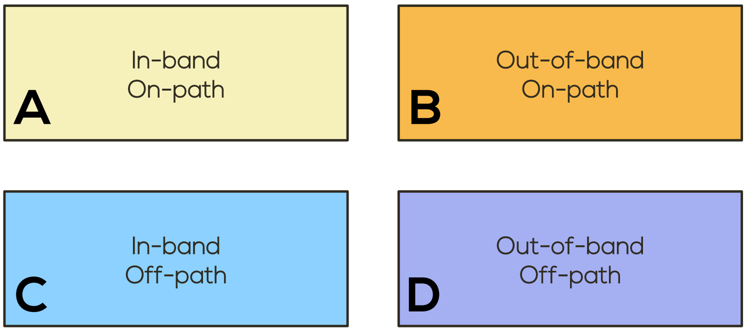


Figure 3.2. Framework for classifying congestion control [127].

3.4 Framework for Classifying Congestion Control

A rate-control or congestion control algorithm relies on congestion cues to pick a new sending rate. These cues are either observed at the receiver or by intermediaries monitoring the flow, or are aggregated by a third-party[12] or a super-peer in an overlay network. Consequently, these observed cues need to be signalled back to the sender which will perform congestion control. We classify these congestion cues as a combination of *where are they measured/observed?*, and *how is the sender notified?* For each, there are two options: On-path and Off-path *sources* and In-band and Out-of-band *signalling* [127]. On-path congestion cues are measured by the receiver or by intermediaries along the path. Off-path congestion cues are reported by devices outside the media path (congestion maps, overlays, etc.). The combination forms four cases which are visualised in Figure 3.2.

A congestion control algorithm needs to pick one or more measurement point (picking multiple adds to the feedback overhead) and then choose a method to signal it to the sending endpoint. The algorithm can choose to report it in-band by encapsulating the cues, either by piggybacking them on the endpoint's own media packets as RTP header extensions (this adds to the header overhead of a media packet) or as RTCP extension blocks (see section 3.3 for details on feedback frequency). Or the congestion control algorithm can choose to signal the cues out-of-band, i.e., re-use the signalling path (e.g., SIP, XMPP) or setup an alternate signalling path (e.g., HTTP or websockets). The following are examples for each category in the framework:

`A) On-path, In-band`: The congestion control algorithm in this case relies on the cues reported in an RTCP feedback from the receiver. For

[12]A system outside the signalling or media path

example, TFRC using information in RTCP RR, TFRC using additional loss reported by ECN markings, Temporary Maximum Media Stream Bit Rate Request (TMMBR), Receiver Estimated Max Bit rate (REMB).

`B) On-path, Out-of-band`: The congestion control algorithm relies on the cues reported in the signalling channel; for example, RTSP implements a *Speed* parameter to vary the transmission rate, or 3G base stations announce the new rate when capacity changes due to cell-loading or handover.

`C) Off-path, In-band`: The congestion control algorithm relies on reports from multiple in-band sources; for example, in Multipath RTP, congestion on one path causes a change in the fractional distribution of traffic on each path.

`D) Off-path, Out-of-band`: The congestion control algorithm relies on third party sources such as receiving bandwidth or congestion notifications from congestion maps, bandwidth lookup services, super-peers and overlays.

3.5 Types of Video Frames in Interactive Multimedia

In this subsection we briefly discuss codec design related to interactive multimedia, for which we limit the discussion to H.264 [76] and VP8 [16] codecs. However it may apply to other modern codecs as well. Typically, the codec design is conceptually divided into a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL). The VCL takes a complete uncompressed frame and outputs slices, which contains one or more macroblock. The NAL encapsulates these slices into packets, which can be transmitted over IP networks.

H.264 [76] has three types of frames: I-, P-, and B-frames. The I-frame is an *intra-coded frame*, which does not dependant on any previous frame and is in effect like a compressed static image. In contrast the other two types of frames hold only parts of the image, are hence smaller in size but depend on other frames to be successfully decoded. A P-frame is a *predictive coded frame*, it encodes only the changes from the previous frame. A B-frame is a *bipredictive coded frame*, it encodes the changes from the previous and next (future) frame. In interactive multimedia, the use of B-frames is avoided because depending on future frames introduces unnecessary delay in decoding them[13]. Therefore, all the algorithms proposed in this thesis

[13]B-frames are extensively used in media streaming applications, as it aids in

use just I- and P-frames during media encoding.

VP8 [16] has two types of frames: intraframes and interframes. Intraframes[14] are compressed without references to any previous frames and can therefore be decoded by an endpoint without receiving any other frames. Essentially, the decoder resets its internal state and stops any previous errors from propagating because of loss of intermediate frames. Interframes[15] are encoded with a reference to previous frames up to the most recent intraframe. A loss or corruption of a single interframe will affect the decoding of all the subsequent dependent interframes i.e., rendering errors will occur, until a new intraframe is received. To overcome this limitation, VP8 introduces a concept of golden frames or alternate reference frames, wherein interframes can refer to not only the most recent interframe but also to previous intraframe. This strategy increases the coding overhead and attempts to avoid retransmitting the lost intraframe, but requires the decoder to keep more intraframes in memory for correctly decoding interframes with several reference frames.

3.6 Congestion Control Evaluation

Real-time interactive communication differs greatly from *elastic* traffic because the sender generates media packets in real-time and expects it to be delivered in hundreds of milliseconds, and the receiver consumes the media packets almost immediately, hence late-arriving packets are useless. Additionally, real-time communication systems are able to tolerate some amount of packet loss and adapt the media rate over a fairly large range. [80] lists a set of requirements for RTP-based interactive multimedia sessions; these requirements form the basis of the guidelines described in [131]. In [121], we define a catalogue of *traffic flows* traversing through a *network topology* with varying *link characteristics* and diverse *queuing* strategies. By picking one feature from each category, we construct scenarios to evaluate the performance of the congestion control. The evaluation scenarios are built using the following components: network topology, link and router characteristics.

The difference between testing in real-world deployments and in simulations is also important to consider, mainly in terms of the accuracy of

rewinding and fast forwarding.

[14]Also known as I-frames or key frames in H.26x codecs

[15]Also known as P-frames or dependent frames in H.26x codecs

RTT measurements which impacts delay-based algorithms (e.g., TFRC). Time-slot-driven simulation systems, such as *ns-2* [125], have accurate timing that is not representative of real-world systems. Testbeds usually use dummynet [23], NetEm [70], or packet traces to emulate the variation in link capacity, latency, intermediate router queue length, and packet losses.

It is also possible to use actual machines placed at geographically distinct locations and send media traffic between them; however, in this case, it is not possible to run a controlled experiment anymore because of the varying amount of cross-traffic generated by other hosts in the network. Usually the last step in the evaluation process involves deploying the congestion control algorithm on several (thousands of) endpoints and showing that it behaves as described without breaking anything on the network (i.e., causing a congestion collapse).

3.6.1 Network flows

In this section, we describe typical test scenarios for evaluating congestion control algorithms. These test scenarios are not supposed to be exhaustive but show the applicability range of the algorithm.

1. `Single media flow on an end-to-end path`: This scenario describes the best case, wherein the network puts each flow identified by its 5-tuple (protocol, source and destination IP address, source and destination port numbers) in its own queue, thus the flow using the proposed congestion control algorithm does not encounter any cross-traffic.
2. `Single media flow competing with multiple similar flows`: In this scenario, the flow using the proposed congestion control algorithm competes with multiple flows using the same congestion control algorithm (i.e., all flows are interactive multimedia).
3. `Single media flow competing with multiple TCP flows` : In this scenario, the flow using the proposed congestion control algorithm competes with TCP flows. These maybe *short* TCP flows representing common web-traffic patterns or *long* TCP flows depicting bulk transfers (e.g., large file downloads).

3.6.2 Link characteristics

The link characteristics can be broken down into the following categories: capacity, latency, and loss. The capacity of a link mainly varies in wireless networks (for example in 3G, LTE, WLAN, etc). In Wireless LAN (WLAN) networks, the capacity varies depending on the density of nodes using the network. The capacity in mobile networks (e.g., 3G, LTE) fluctuates for each user because of fading, interference, mobility, handover, cell loading, etc. The latency of a link is measured as the propagation and serialisation delay. Queuing delay is based on the queue size of the router and hence, is a router characteristic. Latencies between nodes typically vary from a few milliseconds to a few seconds. Commonly used values are: LAN (very low delay, <1 *ms*), low delay (<40 *ms*), trans-continental (>100 *ms*), or satellite links (>500 *ms*). Packet losses are modelled using the Gilbert-Elliott Model [65, 48] or by packet traces [49, 2].

3.6.3 Router characteristics

Apart from managing packet routing, a router also manages congestion; when a packet arrives at a higher rate than it can be processed, the router queues the packet. The routers then use *priority queuing*, *fair queuing*, or *weighted fair queuing (WFQ)* [12] to decide which traffic class to transmit or drop packets from during congestion. When congestion occurs within the same traffic class, the router discards packets using *tail drop*, *Random Early Detection (RED)* [54], or *Weighted Random Early Detection (WRED)*.

We describe the queue sizes as a function of time, i.e., it is the depth of the queue or the amount of time the packet will remain in the queue before it is discarded. However, in practice, the queue size is measured in number of packets. We convert the queue depth (measured in time) to queue length (number of packets, MTU is typically 1500 bytes) using:

$$\text{QueueSize}_{\text{packets}} = \frac{\text{QueueSize}_{\text{sec}} \times \text{Throughput}_{\text{bps}}}{\text{MTU} \times 8}$$

For example, a router with a throughput of 1 *Mbps* and a 1 *s* queue depth would be capable of handling 83 packets (queue length). A 100 *ms* queue depth may represent a short queue, while a 10 *s* queue depth represents a buffer-bloated queue.

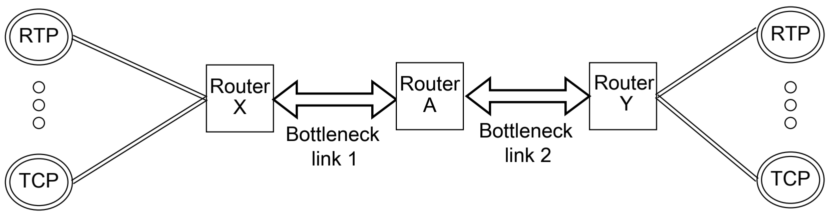


Figure 3.3. Typical network topology for evaluating congestion control consisting of traffic flows (evaluating flow and cross traffic), links and routers.

3.6.4 Network topology

We should run different types of network topologies to evaluate the performance of congestion control. Depending on the amount of cross-traffic, the bottleneck link will move from one node to another in the network. Also, the bottleneck in each direction may be different due to the asymmetry of access links. Additionally, the varying capacity of the access link (e.g., in wireless environments) may be the bottleneck.

Figure 3.3 shows an example of the evaluation setup. This topology is commonly called the *dumb-bell* topology. Another common topology is *parking lot*, which uses multiple bottlenecks instead of a single bottleneck; however, both use common concepts.

To define a scenario, we need to choose the following: the type of cross-traffic (self-similar, short- or long-lived TCP), the characteristics of the cross traffic (e.g., duration), the characteristics of the edge routers (Router X and Y) and the impairments in the network. Lastly, we have to measure and analyse the performance of the multimedia system.

3.6.5 Our Evaluation Setup

Our research made use of the network simulator (*ns-2*) [125] or a testbed made up of real-machines. The individual link characteristics in the testbed were either controlled by NetEm [70] or by dummynet [23]; the intermediate machines that ran NetEm/Dummynet ran kernels re-compiled at 1000Hz for better performance. The endpoints typically ran stock Linux, such as Ubuntu 10.04 or 12.04. In some cases, we used bandwidth and packet loss traces provided by 3GPP [1] or collected them ourselves [24]. Additionally, we ran some tests between machines in Helsinki and the Amazon data centers located in Virginia (US-EAST) and Ireland. The details of the individual test scenarios are discussed in detail in the associated scientific papers.

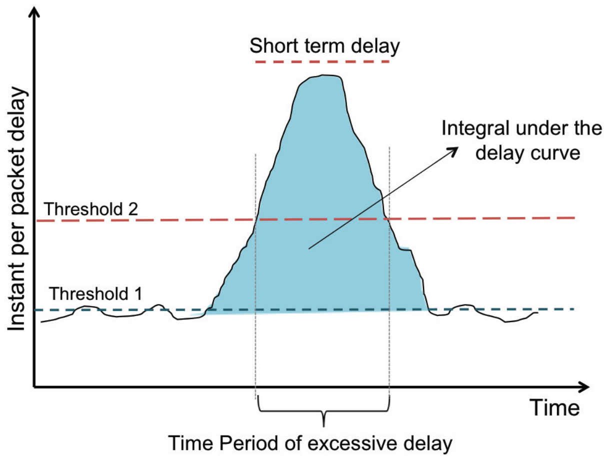


Figure 3.4. Metrics for congestion control.

3.6.6 Metrics for Congestion Control

In this section, we introduce metrics for evaluating congestion control algorithms. Multimedia application 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. The sender's goal is to minimise losses at the receiver while maintaining a stable throughput. Losses are caused by congestion or by bit-errors and are detrimental to the perceived video quality. Although, real-time communication is tolerant to a small amount of losses, bursty loss should be avoided.

Figure 3.4 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:

- `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.
- `Threshold 2` defines the maximum acceptable one-way delay for a packet after which the rendering of the received media packet is no longer meaningful. Packets arriving later than threshold 2 will be discarded.
- The `short-term delay peak` reflects the maximum delay peak encoun-

tered during a congestion control operation. This may be either caused by the appearance of cross-traffic on the bottleneck link or due to congestion resulting in self-inflicted delay.

For losses, we consider two values:

- **`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).
- **`Packets discarded`** at the receiver because their arrival delay violated threshold 2.

Additionally, we measure the instantaneous and average encoder rate, receiver rate and goodput. The instantaneous rate is calculated at 1 second intervals. The Average Bandwidth Utilisation (ABU) is the ratio between the instantaneous goodput (or encoding, receiving bit rate) and the instantaneous channel capacity at 1 second intervals. An ABU larger than 100 % represents over-utilisation and the duration over which it is over 100 % signifies the congestion period.

Peak Signal-to-Noise Ratio (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. 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 behaviour of the human visual system [74]. Another criticism against PSNR is that it does not incorporate time in its calculation. Despite its shortcomings, we use PSNR as a yet another indicator for measuring the performance of congestion control. Recently, a number of new techniques have been proposed for measuring video quality more accurately than PSNR: Video Quality Metric (VQM) [73], and Perceptual Evaluation of Video Quality (PEVQ) [74, 75], but these metrics are not used in this thesis. One reason for not using them is that these metrics have not yet been adapted for interactive real-time media applications and are currently suited for fixed quality media streaming (e.g., in cable television).

3.7 Circuit Breakers for Unicast RTP Sessions

If congestion control is not implemented by multimedia applications, they can cause severe congestion in the network, especially if high data rate media traffic is sent over low-capacity networks. This can not just disrupt the multimedia's quality of experience but also other applications on the network.

We are developing a circuit breaker algorithm that can work with unmodified RTP applications to determine when these non-adaptive multimedia applications are causing excessive network congestion and force them to cease transmission. We envision that the congestion control algorithms for multimedia standardised in the IETF will need to work inside the envelope of this circuit breaker algorithm [131], i.e., a multimedia application implementing congestion control should not trigger the circuit breaker. Consequently, the circuit breakers cannot be too aggressive in terminating media flows because it should allow sufficient time for the congestion control algorithm to monitor and respond to congestion cues.

The circuit breaker algorithm in the short term will serve as a policer, during which time the congestion control algorithm is developed. Developing standard congestion control algorithms for unicast RTP-based interactive multimedia applications is expected to be a multi-year process in the IETF. Therefore, the development of the circuit breaker is on a tight schedule, to be ready for inclusion in the initial roll-out of the WebRTC (Web-based Real-time Communication) framework [79] in web browsers.

3.7.1 Circuit Breaker Design

The RTP circuit breaker algorithm relies on the basic feedback mechanisms defined in the RTP Control Protocol (RTCP) [124]. That is, it solely uses the information available in the RTCP Sender Report (SR) and Receiver Report (RR) packets to detect if the flow is overusing the capacity or causing congestion.

The congestion indicators considered for implementing circuit breakers are: 1) the network *round trip time* (RTT) calculated using timing information in RTCP SR and RR packets, 2) the *jitter* estimated by the receiver over the last reporting interval, and 3) *fractional packet loss* and *cumulative loss* reported by the receiver during the last RTCP interval. These indicators unfortunately only provide a limited insight into the behaviour of the network and cannot be used as strong signals for a circuit breaker.

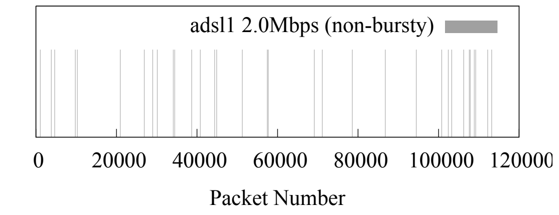
Variation in RTT is used as a congestion indicator in delay-based congestion control algorithms. Additionally, some algorithms use RTT estimates to configure connection timeouts. In RTP/RTCP, the RTT is estimated infrequently because the feedback intervals are rather long, making it difficult to detect the cause in variation of delay. Likewise, variation in jitter can indicate a transient network congestion but does not provide a strong enough signal to implement a circuit breaker. On the other hand, loss is a strong indicator of congestion in networks where packet losses predominantly occur due to queue overflows, and is a less accurate indicator where packet loss occurs due to bit-error corruption (e.g., wireless and mobile links). Therefore, we base the circuit breaker conditions on packet losses.

1. `Media Timeout`: An endpoint is sending media data but when the receiver reports a non-increasing *Highest Sequence Number* (HSN) for two consecutive RTCP intervals, the flow is terminated.
2. `RTCP Timeout`: An endpoint is sending media data but if it receives no RTCP RR for three consecutive RTCP intervals, the flow is terminated.
3. `Congestion`: An endpoint is sending media data and if it receives RTCP RRs indicating fractional packet loss, it calculates the TCP-friendly rate and compares it to the sending rate. If the sending rate exceeds the TCP-friendly rate by a factor of 10 for two consecutive RTCP intervals, the flow is terminated.

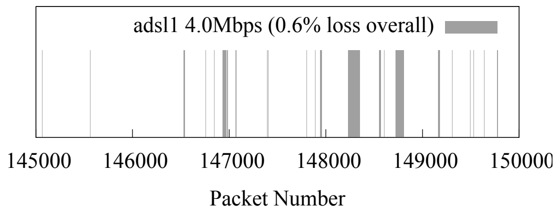
Full details of the RTP circuit breaker algorithm is specified in [102], which is a work in progress and covers various deployment cases such as multiple media sources, impact of shorter-than-standard reporting interval, deployment of Explicit Congestion Notification (ECN), etc.

In Publication I, we apply these circuit breaker conditions to non-adaptive RTP media flows deployed on ADSL and cable modem links. Such flows typically do not implement congestion control at this time, and are likely to cause congestion if deployed on the Internet. We carried out a series of experiments based on real-world traces and on a testbed emulating real-world conditions. Our results show that the proposed RTP circuit breaker performs well, triggering in cases of bursty loss and in sessions that are congesting the links, and does not trigger in low-loss and non-bursty scenarios.

We simulated the RTP circuit breaker performance on 3833 generated



(a) Non-bursty



(b) Bursty

Figure 3.5. Sample non-bursty (a) and bursty (b) packet loss traces. The bursty packet triggers the circuit breaker even though the overall packet loss ratio is 0.6 %.

Loss Pattern	Triggered	Did not trigger
Loss free	0.0 %	100.0 %
Non-bursty loss	0.0 %	100.0 %
Bursty loss	11.9 %	88.1 %

Table 3.1. Sessions triggering the circuit breaker by loss pattern.

RTCP traces corresponding to the measurements collected in *dataset-A* and *dataset-B* of [49]. Of these, 1626 traces have no packet loss, and hence cannot trigger the RTP circuit breaker. The remaining traces each include at least one lost packet. We categorise the remaining traces into two categories: those that have non-bursty packet loss, and those that exhibit bursty loss using the definition of bursty loss from [59] (Figure 3.5 shows representative samples of the non-bursty and bursty packet loss patterns). The data comprises 1344 traces with bursty loss and 863 traces with non-bursty loss.

Table 3.1 shows the fraction of sessions that triggered the RTP circuit breaker for each of the categories of packet loss. The RTP circuit breaker did not trigger for sessions without loss; it also did not trigger for any of the sessions with non-bursty packet loss. However, we observe that the RTP circuit breaker is triggered more often in sessions that contain bursty packet loss.

The circuit breaker conditions trigger mainly due to loss. Figure 3.6 shows that the percentage of sessions triggering the circuit breaker in-

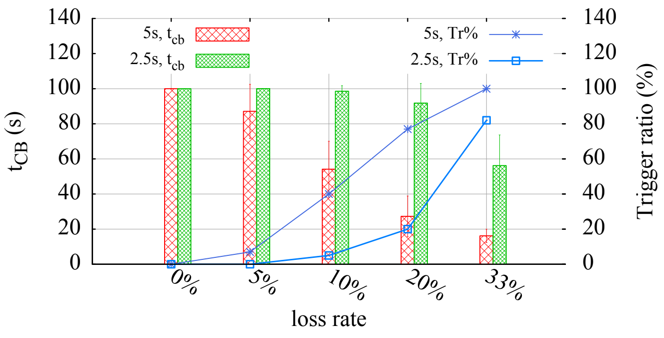


Figure 3.6. Impact of using a shorter RTCP interval on the circuit breaker. Each scenario was run 50 times and the error bars represent the 95 % confidence level.

creases with the increase in loss rate. The figure also shows the impact of the RTCP reporting interval, i.e., by reducing the RTCP interval from 5 s to 2.5 s, fewer sessions are terminated. The endpoints become robust to loss of feedback packets by sending feedback often and we observe a longer time for triggering the circuit breaker (t_{CB}).

3.7.2 Discussion about TCP throughput equations

In Section 3.7.1, we described the circuit breaker triggers when the sending rate exceeds the estimated TCP throughput by a factor of 10. We can estimate the TCP throughput either using Padhye's full TCP throughput equation [99] or using Mathis's simplified TCP throughput equation [92].

In [99], Padhye *et al.* show that the TCP throughput for a long-lived TCP Reno connection can be estimated using the following equation:

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

While Mathis *et al.* in [92] show that under conditions of low packet loss, Padhye's equation can be simplified to:

$$X_{kbps} = \frac{8 \times s}{R \times \sqrt{\frac{p \times 2}{3}})}$$

X is the transmit rate in kbps.

s is the average packet size in bytes.

R is the round trip time in seconds.

p is the loss event rate, $\forall p \in [0.0, \cdots, 1.0]$.

t_{RTO} is the TCP retransmission timeout value in seconds, usually set to $4 \times \mathbf{R}$.

```
b is the number of acknowledged packets in TCP; typically set b = 1.
```

Mathis *et al.* also show that the simplified TCP equation approximates Padhye's full equation with reasonable accuracy [92] .

In Publication I, our experiments on residential networks shows that the simplified TCP throughput equation performs effectively, while using Padhye's full TCP equation triggers the circuit breaker earlier. The data show that the full TCP equation tends to be more sensitive to packet loss. In LTE networks with a combination of a low RTT and high packet loss rate (due to AQM), the circuit breaker using the simplified TCP throughput equation does not trigger at all [120]. However, using the Padhye's full TCP throughput equation in these cases gives better performance [120]. We believe some of this over-sensitivity is due to averaging packet loss events over long RTCP reporting intervals and the fact that the RTT is estimated only once in that interval.

Overall, our preliminary results derived from experiments in a testbed, and simulations based on real-world traces show that the proposed RTP circuit breaker performs as intended, triggering in the case of bursty packet loss and not triggering in the low-loss and non-bursty scenarios.

3.8 Summary

In this chapter, we aimed to classify congestion control cues for real-time communication based on: *where they are measured?* (by on-path or off-path sources) and *how they are reported?* (via in-band or out-of-band signalling). We also describe other fundamental choices needed to implement congestion control: congestion cues, reporting frequency and circuit-breaker conditions. Additionally, we describe a basic evaluation suite for measuring the performance of any proposed multimedia congestion control algorithm; derivatives of these test scenarios are used to discuss the performance of congestion control in this thesis.

We specified the circuit breaker algorithm and discuss the performance of the circuit breaker applied to non-adaptive multimedia traffic. Our results show that it works as intended, i.e., it provides enough time for a congestion control algorithm to respond to congestion cues before triggering the circuit breaker. We also show that the endpoint can slightly vary the sensitivity of the circuit breaker by choosing between the full and simplified TCP throughput equation. In the forthcoming chapters, we discuss our proposals for congestion control in various environments, several of which work within the constraints imposed by the circuit breaker algorithm.

4. Congestion Control for Interactive Multimedia

Figure 3.2 in Chapter 3 shows the structure of the congestion control framework described in this thesis. The framework categorises *on-path* sources and *in-band* signalling for implementing congestion control (corresponds to *Block A* in Figure 3.2), which are discussed in this chapter. This chapter is based on our work on congestion control for interactive multimedia applications, which is documented in Publication II, Publication III, Publication IV, [96], [132], [126], [43] and [135].

In Publication II, we propose a new congestion control algorithm for the mobile (e.g., 3G) environment, to be deployed in the IP Multimedia System (IMS). The main distinction between mobile (e.g., 3G, LTE) and other wireless environments (e.g., 802.11x) is that the media streams are transmitted using the *unacknowledged mode*; the packets corrupted due to bit-errors (e.g., wireless interference) are not re-transmitted. Hence, the packets incur low delay, compared to Wireless LAN where corrupted packets are retransmitted by the link layer. We propose a sender-driven and a receiver-driven congestion control, and evaluate the performance of the proposed congestion control algorithm in a simulated environment (in ns-2) using real-world 3G traces [1, 2]. In Publication III, we extend the approach in Publication II for deployment on the Internet and show that the congestion control scheme is deployable there as well. In [96] and [132], we propose RTCP XR block extensions that indicate the number of bytes discarded and run-length encoding of discarded packets, respectively. These packets are discarded by the receiver because they arrived too early or too late to be played out by the receiver. This information is used as a congestion cue by the sender.

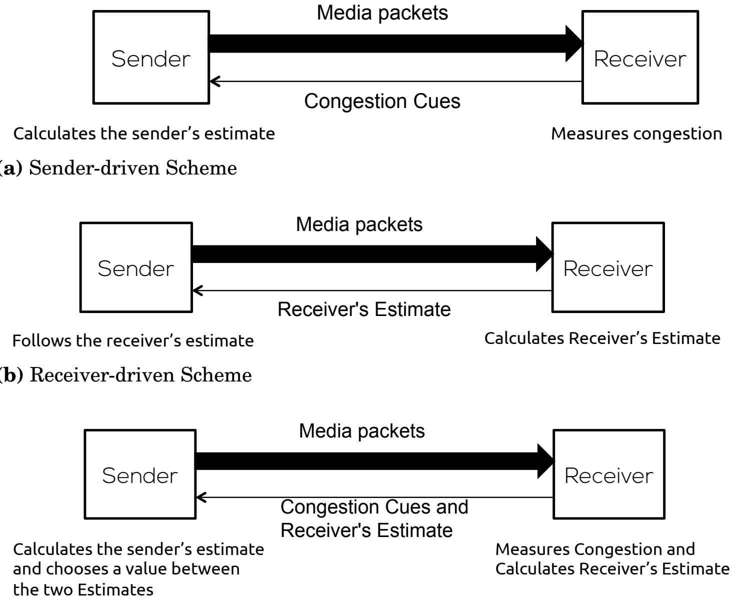
Lastly, in Publication IV we evaluate the performance of a congestion control algorithm proposed by Google for WebRTC. We evaluate the performance in diverse scenarios measuring scalability (*how quickly is the* *congestion control able to utilise the available capacity*), self-fairness and competing against bursty cross-traffic. We evaluate the performance of web browsers implementing the congestion control algorithm in our testbed that emulates the diverse scenarios.

4.1 Schemes of Congestion Control

The congestion control algorithm can be implemented at the sender, at the receiver, or the sender and receiver can operate co-operatively. The *sender-driven* scheme requires that the receiver measures the current network condition and signals the observed congestion cues to the sender, which calculates the sender's estimate and uses it as the new sending rate. In the *receiver-driven* scheme, the receiver calculates the new sending rate (receiver's estimate) based on the observed congestion cues, and signals the new rate to the sender, which, on receiving the new rate, adapts the media bit rate to the received value. The *co-operative* scheme is an extension to the *sender-driven* scheme. In this case, the receiver calculates the receiver's estimated rate and signals it along with the observed congestion cues; the sender at its end calculates its own estimate based on the congestion cues and chooses a new sending rate, typically a value in between the sender's estimate and the receiver's estimate. Figure 4.1 shows the interaction of the sender and receiver for each scheme. The figure merely shows the media flow in one direction; however, it should be noted that the media in the simulation and the emulated testbed actually flow in both directions unless explicitly mentioned. This is mainly done for the convenience of representation and followed throughout the remainder of the thesis.

4.1.1 Sender-driven Congestion Control Schemes

TCP Friendly Rate Control (TFRC) is an equation-based congestion control algorithm implemented at the sender [52] and is also implemented as a profile [56] in the Datagram Congestion Control Protocol (DCCP) [84]. TFRC uses the average packet size, round trip time (RTT), and loss ratio (p) [68] to calculate the new sending rate. Formally, the sending rate in TFRC is calculated as follows:


Media packets
Sender
Receiver
Congestion Cues
Calculates the sender's estimate
Measures congestion
Media packets
Sender
Receiver
Receiver's Estimate
Follows the receiver's estimate
Calculates Receiver's Estimate
Media packets
Sender
Receiver
Congestion Cues and Receiver's Estimate
Calculates the sender's estimate and chooses a value between the two Estimates
Measures Congestion and Calculates Receiver's Estimate

(a) Sender-driven Scheme

(b) Receiver-driven Scheme

(c) Co-operative Scheme

Figure 4.1. Congestion control schemes: a) sender-driven, b) receiver-driven and c) co-operative.

$$TFRC = \frac{8 \times avg\_packet\_size}{R \times \sqrt{\frac{2 \times b \times p}{3}} + t_{RTO} \times \left(3 \times \sqrt{\frac{3 \times b \times p}{8}}\right) \times p \times (1 + 32 \times p^2)}$$

$$where,\ b = 1$$
$$t_{RTO} = 4 \times R$$

TFRC cannot directly be applied to RTP because TFRC requires per-packet feedback, and in RTP, the RTCP feedback is not necessarily sent that often [101]. Therefore, [64] maps the TFRC timing rules defined in [55, 53] to that of the RTP/RTCP feedback loop. It also proposes extensions to the timing rules in the AVPF-profile [98] for very short RTTs ($< 20ms$). [62] and [15] show that TFRC is stable on paths with longer RTTs than those with smaller RTTs, but it too exhibits saw-tooth behaviour [122]. Any algorithm that consistently produces a saw-tooth media rate is not well suited for real-time communication because it generates a poor user-experience [63, 167].

Other sender-driven congestion control algorithms that we explored include the Rate Adaption Protocol (RAP) [111] that uses a windowed approach which exhibits a saw-tooth-type of behaviour. Zhu *et al.* [164] use

Pre-Congestion Notification (PCN), Explicit Congestion Notification (ECN) and loss rate to get an accurate delay estimate for implementing congestion control. In this case, they assume all packets marked by ECN and PCN as lost. Their algorithm as specified now, relies on accurate measurement of one-way delay relying on clock synchronisation. Instead of just relying on RTT and loss for congestion control, Garudadri *et al.* [60] also use the receiver playout buffer to detect the link utilisation, i.e., the variation in the receiver playout buffer occupancy indicates an increase or decrease in congestion. O'Hanlon *et al.* [95] propose using a delay-based estimate when competing with similar traffic and, using a windowed-approach when competing with TCP-type cross traffic, they switch modes by applying a threshold on the observed end-to-end delay. The idea is similar to the one discussed in [22].

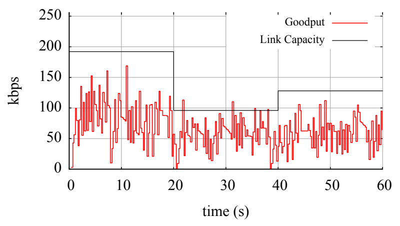
4.1.2 Receiver-driven Congestion Control Schemes

In receiver-driven congestion control, the receiver estimates the rate and notifies the sender about the new sending rate. Temporary Maximum Media Bit-rate Request (TMMBR) is defined as a codec control message in [147]. It is generated by the receiver in a point-to-point video call. The receiver calculates the new estimate (available capacity) based on the average inter-arrival time of RTP packets (*video frames*). When the inter-arrival time of the video frames increases beyond the expected arrival time in an observed period, the receiver senses *over utilisation*. When the frames arrive early, the receiver senses *under utilisation*. The receiver ignores the congestion event if it occur on short timescales and when receiving I-frames. The I-frames are large frames because they are spatially compressed and are not temporally correlated to previous frames. Hence, these I-frames are expected to cause queuing delay. The receiver, on detecting link over or under utilisation, modifies the *receiver's capacity estimate*. The receiver sends the TMMBR message to the sender indicating the maximum sending rate. Currently, interactive multimedia sessions in 3GPP [5] use TMMBR messages to notify the sender of the expected sending rate. In WebRTC [79], TMMBR is expected to be used initially, before RTP congestion control is standardised by the IETF [104]. The expectation is that different WebRTC clients may develop proprietary receiver-driven algorithms and use TMMBR as the standardised mechanism to communicate the capacity estimate to the sender, which will blindly follow it.

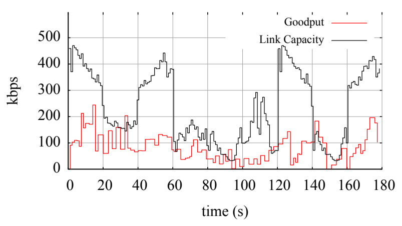
4.1.3 Co-operative Congestion Control Schemes

The Next Application Data Unit (NADU) [41, 42] is designed for rate adaptation for video streaming in 3GPP [6]. A NADU receiver measures the playout delay (as a measure of buffer occupancy in time) and signals it to the sender along with the next sequence number to be played out. Conversational NADU (C-NADU) is an extension of NADU for congestion control for interactive multimedia and is described in Publication II and Publication III. In C-NADU, the receiver also calculates the *receiver's capacity estimate* by measuring the frame inter-arrival time and signals that along with the NADU report. If the video frame arrives at the expected time, the receiver assumes no ongoing congestion, and if it arrives later than the expected time, the frame is considered late and the receiver diagnoses congestion. If the frame is delayed and misses its playout time, it is discarded and in this case the receiver estimates congestion. Based on the above cases, the receiver estimates the current capacity and signals it to the sender. At the sender, the C-NADU controller calculates the TCP-friendly rate, measures the variation in RTT (75 and 90 percentile values) and calculates the fraction of video frames that missed their playout deadline. Based on these congestion cues and the receiver estimate, the sender chooses a new sending rate.

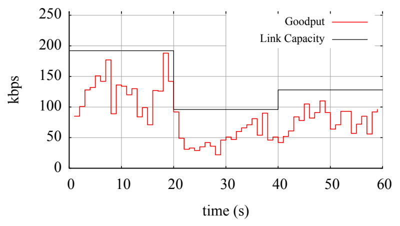
Receiver-side Real-Time Congestion Control (RRTCC) is described in [8] and is proposed as one of the solution candidates for WebRTC by Google. Like C-NADU, RRTCC also has a receiver- and sender-side component. The receiver-side measures the under or over utilisation by monitoring the timestamp jitter of the incoming frames. The arrival times are modelled as a white Gaussian process. When the mean is 0 there is no congestion; the mean is expected to increase when there is ongoing congestion and expected to decrease when the congestion abates. Based on this expectation, the receiver calculates the capacity estimate and signals it to the sender. The sender calculates its estimate based on TFRC, and finally, chooses the new sending rate as a value between the TFRC rate calculated by the sender and the receiver's estimate. Full details of the algorithm proposed by Google are documented in an Internet-Draft [8].



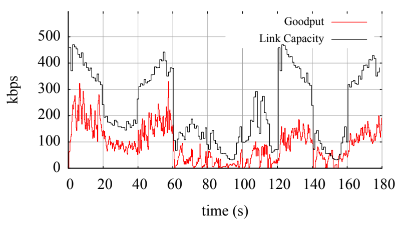
(a) TFRC



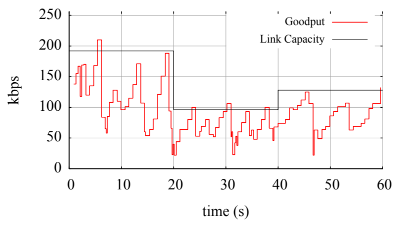
(b) TFRC



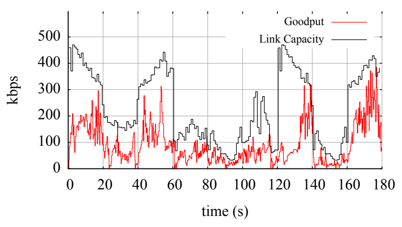
(c) TMMBR



(d) TMMBR



(e) C-NADU



(f) C-NADU

Figure 4.2. Performance of TFRC, TMMBR and C-NADU in a slow time-varying link (a, c, e) and 3G network (b, d, f).

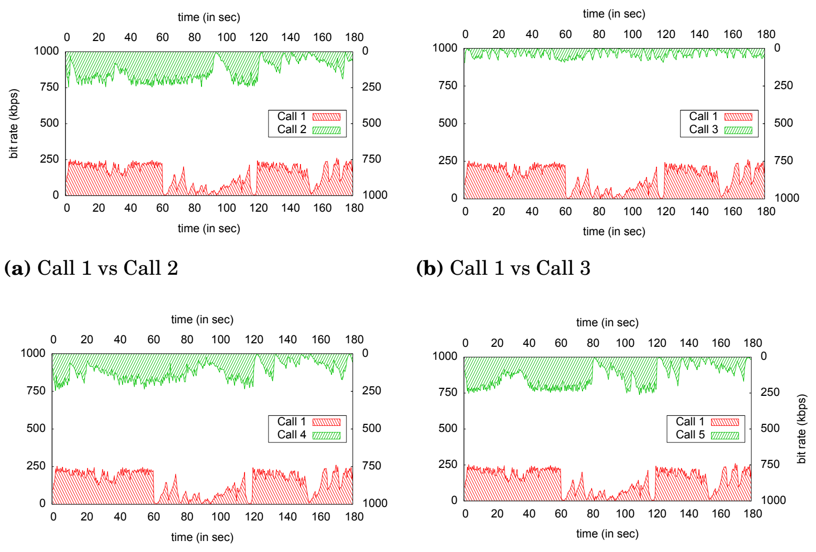
	$Goodput_{avg}$ (kbps)	PLR (%)	$PSNR_{avg}$ (dB)
TFRC	84.1	6.9 %	29.3
TMMBR	89.8	3.7 %	30.5
C-NADU	92	2.2 %	31.9

Table 4.1. Comparing TFRC, TMMBR, C-NADU for calls over mobile nodes (180 *s* simulations using 3G traces).

4.2 Performance Analysis of TFRC, TMMBR, C-NADU, and RRTCC

This section briefly discusses the performance of each congestion control algorithm. The detailed analysis can be found in the respective papers.

Our results in Publication II show that TFRC produces a saw-tooth sending rate, similar to the performance in [122]. When the media stream is the only flow on the end-to-end path, we also observe that the average bandwidth utilisation (ABU) is between 30-40 %, i.e., TFRC under utilises the link and the loss ratio is about 6 %, which results in a lower media quality (approximated by measuring PSNR) compared with the other two



(a) Call 1 vs Call 2 **(b)** Call 1 vs Call 3

(c) Call 1 vs Call 4 **(d)** Call 1 vs Call 5

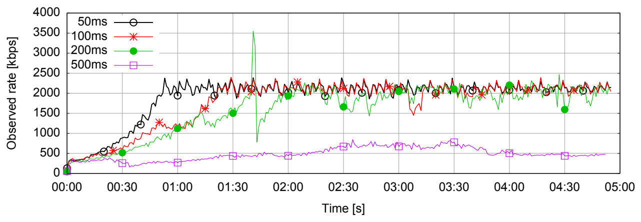
Figure 4.3. Performance of five C-NADU calls competing for capacity on a shared bottleneck in a heterogeneous network. Each call needs to quickly adapt to changes in 3G link capacity and fairly share the bottleneck link.

	3G Capacity Pattern	$Goodput_{avg}$ (kbps)	PLR (%)	$PSNR_{avg}$ (dB) (σ)	ABU (%)
Call 1	Excellent-Poor-Elevator	140.10	2.15 %	31.4 (0.39)	70.1 %
Call 2	Good-Good-Poor	133.55	1.61 %	31.9 (0.62)	66.8 %
Call 3	Poor-Poor-Poor	35.18	1.55 %	22.2 (1.13)	17.59 %
Call 4	Fair-Fair-Poor	114.96	2.75 %	31.1 (0.75)	57.5 %
Call 5	Excellent-Elevator-Poor	130.23	2.25 %	31.3 (0.13)	65.1 %

Table 4.2. C-NADU: Five calls in a heterogeneous network with end-to-end latency between 60-120 *ms* and 0.5 % link-layer losses.

schemes (see Table 4.1). TMMBR-based congestion control utilises the link better than TFRC (ABU between 50-70 %) and produces a lower loss ratio (≈3 %). Lastly, C-NADU has comparable bandwidth utilisation (ABU=55-60 %) and loss ratio (≈2 %) to TMMBR. Figure 4.2 shows the performance of TFRC, TMMBR and C-NADU over two types of bottleneck links, a slow time-varying link and a 3G link.

In Publication III, we show that C-NADU is self-fair with other C-NADU flows in both wired and wireless environments [126], and in Publication VI we show that it competes fairly with TCP cross-traffic, for both the long- and short-TCP flows. Figure 4.3 show five video calls in which each sender uses an independent 3G link into a common bottleneck to the receivers. The 3G links are based on radio link traces and have different capacities. Hence, at some instances of time, the 3G link is the constraint and at other times it is the shared bottleneck link. Table 4.2 shows that four calls have



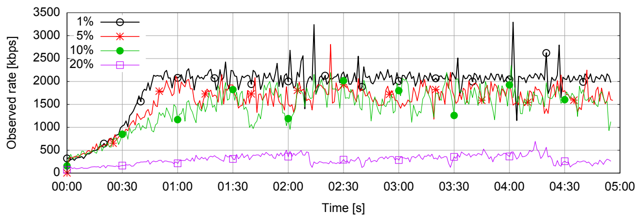


Figure 4.4. The plots show the performance of RRTCC on a link with varying delay and fractional loss rate. We observe that by the sending rate decreases with increasing link latency or bit-error loss.

	$Goodput_{avg}$ (kbps)	Residual Loss (%)	PLR (%)
0 %	1949.7 ± 233.62	0.011	0.011
5 %	1568.74 ± 178.52	0.23	9.77
10 %	1140.82 ± 161.92	0.49	19.02
20 %	314.4 ± 61.98	2.43	36.01

Table 4.3. RRTCC: Metrics for a bottleneck with different packet loss rates.

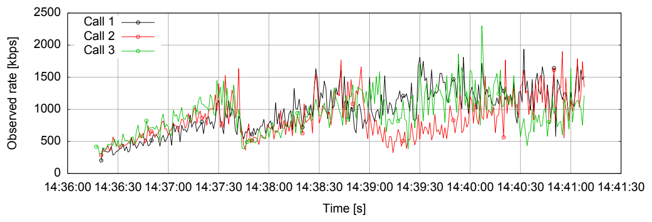
	$Goodput_{avg}$ (kbps)	RTT (ms)	Residual Loss (%)	PLR (%)
3 calls	809.07 ± 202.38	31.48 ± 24.93	0.21	0.23
3 calls (time shifted)	1154.32 ± 250.54	35.15 ± 27.88	0.08	0.91

Table 4.4. RRTCC competing with similar cross-traffic on the bottleneck link.

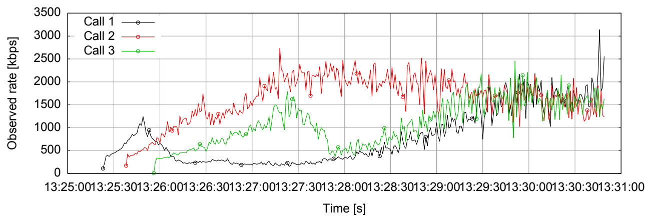
comparable performance (see PSNR and goodput) and one call suffers due to poor connectivity (the 3G link has insufficient capacity which affects the quality).

In Publication IV, we evaluate the performance of RRTCC in several scenarios: by itself on a bottleneck link, competing with other RRTCC flows and competing with TCP cross-traffic. Figure 4.4 shows an example plot of the performance of RRTCC. In Figure 4.4(a) when increasing the bottleneck link latency reduces the sending rate of RRTCC. Similarly, Figure 4.4(b) shows that when increasing the loss rate also affects the sending rate. However, Table 4.3 shows that even though the link has a high loss rate, the residual loss rate is low (even when the loss was 20 %), mainly due to the use of NACKs, PLI and FEC.

Next, we emulate three calls sharing a common bottleneck. In this case,



(a) Start together



(b) Start 30 *s* apart

Figure 4.5. The plots show the variation in receiver rate of three RRTCC flows, a) starting together, b) starting 30 *s* apart. The total duration of the call is 5 mins (300 *s*).

the individual media rates do not reach their individual maximum rate of 2 *Mbps*. Figure 4.5(a) shows that the three calls ramp up at about the same rate, reach a peak and drop their rate simultaneously. The sending rates synchronise, even though the flows originate from different endpoints using independent RTP stacks.

Lastly, instead of the three calls starting together, each call starts at 30 *s* intervals. We observe that while the media rate per call on average is higher, the first call has a disadvantage. In all the cases, the first media flow temporarily starves when a new flow appears, ramping up again after a few minutes. Figure 4.5(b) shows the instantaneous rates of each of the calls. The first call temporarily starves when new flows appear because, when it starts, it is the only flow on the bottleneck and does not encounter any queues; it observes a certain RTT and uses that as the baseline. When the second flow appears, the second flow already observes queues from the existing stream and competes with it, while the initial flow observes an increase in queues and reduces the sending rate to avoid congestion.

To summarise, we observe that the performance of TFRC is bursty, which may lead to poor user-experience, whilst TMMBR, C-NADU and RRTCC have a more stable throughput. Lastly, TMMBR appears to be conservative with very low packet loss, while RRTCC appears to be aggressive with a lot more packet loss. C-NADU appears to be in between the two schemes, with higher throughput than TMMBR and much lower packet loss compared to RRTCC.

4.3 Summary

In this chapter, we describe congestion control implemented using congestion cues from in-band sources and signalled in path (in RTCP). We further categorise the congestion control algorithms based on *where they are implemented*: sender-driven scheme (e.g., TFRC), receiver-driven scheme (e.g., TMMBR), or co-operative scheme (combination of sender and receiver, e.g., C-NADU, RRTCC) and compare the performance of algorithms in each scheme. Our initial experiments show that that C-NADU or TMMBR would be preferred, but requires further investigation with more complex aggregate flows (mainly short or bursty TCP flows).

In the next chapter, we discuss the interaction between the error-resilience algorithm and the congestion control algorithm and evaluate if FEC can be used as a probing mechanism by a multimedia congestion control algorithm.

5. Interaction of Error-Resilience and Congestion Control

Error-resilience is typically a topic discussed orthogonally to congestion control, mainly because error-resilience caters to handling packet loss while congestion control caters to the amount of information sent over the network. This chapter is based on our work on unifying error-resilience and congestion control, which is documented in Publication V, Publication VI, [136], and [130].

In Publication V, we evaluate the performance of the various error-resilience schemes available for use in interactive multimedia communication (mainly applicable to H.264). These are: using Negative Acknowledgement (NACK) or Packet Loss Indication (PLI), Forward Error Correction (FEC) or Unequal Level of Protection (ULP), adaptive slice sizes, and Reference Pictures Selection Indication (RPSI). We evaluate the performance of the proposed mechanisms in diverse scenarios in a simulated environment (in ns-2) using real-world 3G loss patterns [2]. Lastly, based on our observations, we define the applicability of the various error-resilience with respect to end-to-end latency and packet loss.

In Publication VI, we propose using FEC not only for error-resilience but also for congestion control, i.e., instead of probing for available capacity by increasing the sending rate of the media stream, the endpoint sends redundant packets. If a packet gets lost and the added FEC packet arrives in time, the receiving endpoint would recover the lost packet. However, if the packet is not lost, by introducing the FEC packet the sender not only discovers that there is additional available capacity, but also has a sense of the (minimum) magnitude of the available capacity. We compare our proposal with our previous work in Publication II and Publication III, and RRTCC [8]. We evaluate the performance of the mechanisms in diverse scenarios implemented in a simulation environment (in ns-2) and in our testbed.

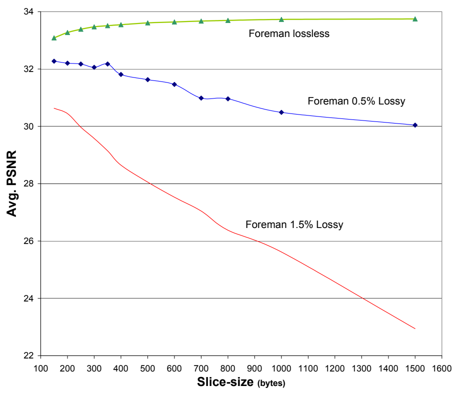


Figure 5.1. Effect of different slice sizes on PSNR for links experiencing different amounts of bit-error corruption.

5.1 Error-resilience Schemes

H.264 [76] uses various error-control methods [85, 144, 143, 146] to overcome loss due to corruption (e.g., in wireless) and non-bursty packet loss (e.g., due to congestion). These methods are classified into three categories: source coding, channel coding, and joint source and channel methods. Source coding refers to the methods implemented by the video codec. Channel coding refers to the methods implemented by the networking layer. Joint source-channel refers to methods that combine source-coding and network-coding mechanisms.

The H.264/AVC codec has several features that support error-resilient mechanisms for video communication that correspond to the above categorisation [146]. At the codec level, the following are available: adaptive slice size, Reference Picture Selection (RPS), and Flexible Macroblock Ordering (FMO) [85, 148]. Similarly, at the channel level, the following are available: Selective Retransmission (NACK), and PLI. An example of joint source-channel is UEP with FEC [143], in which the sender attempts to selectively protect important parts of the bitstream or encodes redundant frames differently than other frames [145].

The performance of the available error-resilience mechanisms vary with the observed end-to-end latency, link loss, bandwidth constraints and operating environment (3G/LTE to 3G/LTE, 3G/LTE to WLAN, or wireless to fixed, etc.). No single error repair mechanism fits all operating environments. A solution that works for an observed packet loss ratio of less than 2 %, may not scale well for paths with higher packet loss or higher latency.

- `Retransmissions`: In RTP, retransmissions are either payload-specific or at the transport-specific. Transport-specific loss contains packet loss information (Generic NACK), while payload-specific loss contains Slice Loss Information (SLI), Picture Loss Information (PLI) or Reference Picture Selection indication (RPS). Typically, the receiver detects a loss and sends it as soon as the RTCP reporting interval allows feedback.
- `Adapting Slice Sizes`: the encoder adapts the size of the picture slice based on the link characteristics; when the channel is lossless there can be one picture per slice (can be larger than MTU, but an endpoint may limit the slice to MTU size) and when high losses are reported, the slices are reduced in size (up to 100 bytes). Larger slice sizes improve encoding efficiency, but are more vulnerable to frame losses. Figure 5.1 shows the variation of average PSNR with respect to different slice sizes in varying loss scenarios. We observe that there is a direct correlation between packet loss, slice size and media quality.
- `Reference Picture Selection`: Instead of retransmitting a lost packet, the encoder finds the list of correctly received pictures (or frames) by the decoder. Hence, for subsequent encodings, the encoder uses a different decoded picture for inter-prediction reference. This method stops the temporal error propagation caused by an earlier packet loss. In the RPS message, the decoder reports the list of pictures (or frames) that were correctly received or lost. Hence, the encoder is able to retrieve the lost picture data. The mode of operation can be decided depending on the observed packet loss rate.
- `Unequal Error Protection`: When the link latency is high, retransmissions cannot be used because the retransmitted packets arrive too late to be played back. In these cases, the sender proactively uses a portion of the available capacity to send redundant packets (typically, FEC), hoping to recover any lost packet before decoding. Hence, by using UEP, the media senders try to strike a balance by protecting only a chosen set of the media packets. In Publication V, we protect the reference frames and not the non-reference frames with UEP.

In Publication V, we stream three different YUV sequences [156] over a 3G link simulated in *ns-2* [125] with varying amount of link layer packet loss (0.5 %, 1.0 % and 1.5 %) [2]. Our experiments show that using NACK, the endpoint is able to correct 15-30 % of the lost packets for an end-to-end path latency of 60 ms, which meets the preferred packet latency

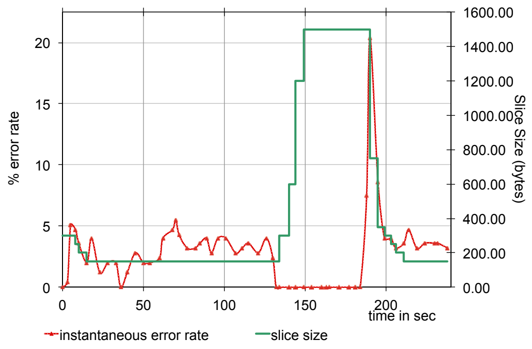
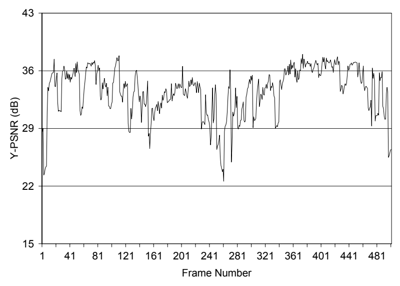


Figure 5.2. The plot shows the variation in slice sizes with varying error rates.

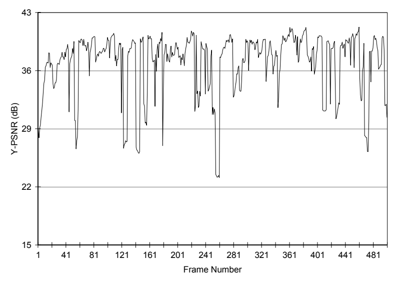
of 150 *ms* specified by ETSI [50]. The number of recovered packets will increase with shorter RTCP reporting intervals; in this paper, we use a 1 *s* reporting interval, with early and immediate reporting as defined in [98]. Our experiments show that NACK is an effective mechanism for low delay scenarios. Similarly, protecting the media flows with UEP incurred a 15-25 % overhead, and about 21-24 % of the lost packets were recovered.

Senders use a moving average of the last three observed fraction loss rates reported in the RTCP RR to vary the slice size. The slice size is doubled for every period with average loss below 1.0 % until it reaches the MTU size (1500 bytes). Slice sizes remain constant for loss rates between 1 % to 2.5 % to provide stability to the receiving system. However, in high loss scenarios, if the slice size is larger than 400 bytes, it is halved, and for sizes below 400 bytes, it is reduced in steps of 50 bytes to a minimum of 150 bytes. Figure 5.2 shows variation of slice-size with the instantaneous loss rate and the average loss rate. Hence, by adapting the slice sizes, the sender is not repairing the stream (or replacing the missing packets), rather, it is attempting to constrain the area of errors in the picture. If an RTP packet containing a small-sized slice is lost, a smaller area of the decoded picture is affected, alternatively, when a packet containing a large-sized slice is lost, a large part of the decoded picture or the complete picture is affected.

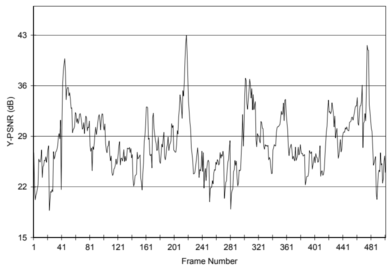
Lastly, using the RPS error-resilience mechanism, the receiver indicates the decoding correctness when losses occur, i.e., which pictures or slices have been decoded correctly or incorrectly. In our experiments, the receiver sends RTCP RR every 250 *ms*, which increases the reporting overhead to 2 % of the session bandwidth. In Figure 5.3 (a), (b) and (c), we observe that the PSNR drops down when a lost frame is referenced and the PSNR increases immediately after the encoder chooses a correct reference picture.



(a) Foreman



(b) News



(c) Football

Figure 5.3. The plots show the PSNR variation when using RPSI to stop decoding error propagation for (a) Foreman, (b) News, and (c) Football sequences.

	Y-PSNR, 0.5 % PLR		
	Foreman	Football	News
NACK	32.1456	28.0331	35.3867
RPSI	33.68	28.05	37.37
UEP	28.33	26.86	34.47
Adaptive slices	32.30	28.4	37.25

Table 5.1. Comparing the performance of different error-resilience schemes on three different types of YUV sequences [156]. The link loss rate is 0.5 % at each 3G link.

When the one-way latency is 100 *ms* and the frame rate is set to 15 frames per second, we observe that the error propagation is stopped by RPS in about 4 to 7 picture frames.

Table 5.1 presents the average PSNR of the different error-resilience schemes for links with 0.5 % fractional loss. We observe that the RPS performs better than UEP and NACK and is comparable to adaptive slices. The UEP under performs mainly because the media stream is encoded at a lower rate to make room for FEC, compared to the media streams of the other error-resilience mechanisms.

In Publication V, we model the error-resilience mechanisms as a function of observed packet loss and end-to-end delay (or latency), and Figure 5.4 summarises the applicability of the error-resilience mechanisms based on our experiments. NACK is useful when loss rates are low and the end-to-

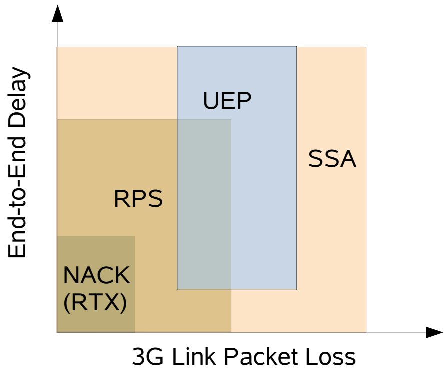


Figure 5.4. Applicability of the error-resilience schemes in a heterogeneous environment containing both wireless and wired links.

end delay is also low. Adaptive slice size (SSA) is applicable to the whole operational region because it attempts to scale the packet size based on the observed loss rate, but does not help in packet repair. RPS works better on links with bursty packet loss, where NACK would not be effective. By correctly choosing a new reference picture, the sender is more effective in repairing the error and it works when network latency is higher. UEP/FEC schemes are mainly useful when the sender and receiver cannot effectively co-operate to repair the stream and the fractional loss is within the FEC's chosen protection range.

5.2 Using FEC for Congestion Control

For many years, interactive multimedia flows have used FEC for error protection [143, 144], i.e., the application trades off additional sending rate for redundant packets to reduce the effect of losses. In Publication V, we show that 21-24 % of the lost packets are recovered using a static amount of FEC. In Publication VI, we investigate the use of FEC packets not only for error-resilience but also as a probing mechanism for congestion control. The FEC packets are sent in a repair flow separate from the media flow that carries the source symbols. Figure 5.5 shows the sender-side FEC framework and the interaction between the FEC module, FEC scheme, and the RTP stack. The rate-controller controls shares the state of the congestion control (e.g., increasing, decreasing, or keeping the media rate) with the FEC module. Additionally, the FEC module keeps track of the losses, discards and post-repair losses reported by the receiving endpoint in RTCP to calculate the new FEC interval. The FEC code subsystem

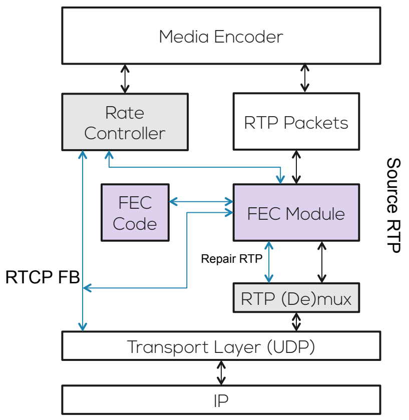
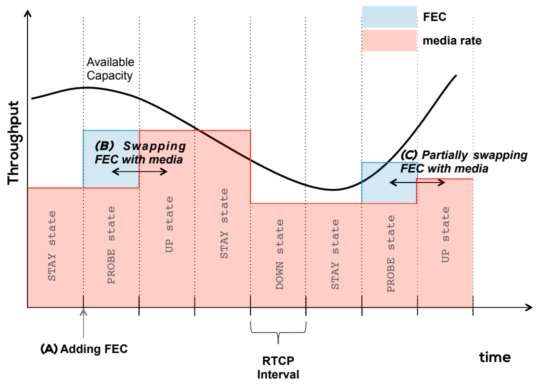


Figure 5.5. Interaction between FEC and RTP.

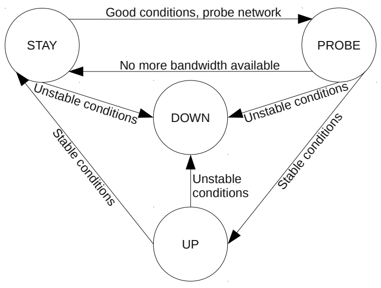
implements the FEC scheme that creates the FEC packets from the RTP packets generated by the sender in the FEC interval. The RTP (De)mux subsystem on receiving the FEC and RTP packets schedules them for transmission.

In Publication VI, the sender generates the FEC packets using parity codes from a set of source media packets encapsulated in RTP. The parity codes are a set of systematic codes in which a number of repair packets are generated from a set of source media packets. There are several FEC schemes available, such as, 1-d and 2-d parity scheme, Reed-Solomon (RS), etc. In the non-interleaved parity scheme an XOR packet is generated for every N source packets. Consequently, this FEC scheme is able to recover from a single random packet loss in N packets. Alternatively, if we apply the XOR operation on source packets that are N sequence numbers apart from each other, the resulting N repair packets are referred to as interleaved FEC packets, which can protect against burst loss as long as the chosen length of "N" RTP packets is longer than the burst packet loss length. To protect against both single random loss event and burst losses, the endpoint ought to use a combination of non-interleaved and interleaved parity FEC. This scheme (2-d parity scheme) is able to cope with burst loss of at least N packet in every B block of packets (multiple of N packets). Typically, the repair packets are encapsulated in an RTP payload format, for example using the packet format described in [87] or [136].

Zhu *et al.*[163, 162] propose using ULP for both congestion control and error-resilience. Firstly, they estimate the available rate using a variant of TFRC, called Multimedia Streaming TCP-Friendly Protocol (MSTFP) [161]. Secondly, they take packet loss and historical sending rate to smooth out



(a) Concept

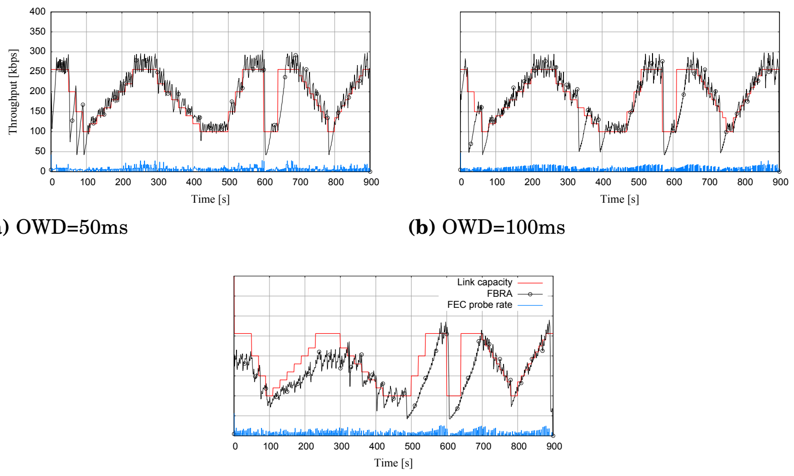


(b) State-machine

Figure 5.6. a) Interaction of FEC and congestion control, b) FEC-Based Rate Adaptation (FBRA) state machine.

the encoding rate. Lastly, they apply FEC while performing congestion control and their results show a significant increase in media quality. MSTFP, on the other hand, does not use RTP/RTCP and acknowledges each packet for calculating the TFRC estimate.

Our concept in Publication VI is similar to that described in [163] but applied to interactive multimedia. The concept is as follows: the sender chooses a high FEC rate to aggressively probe for available capacity and conversely chooses a low FEC rate to conservatively probe for available capacity. While probing, if a packet is lost and the FEC packet arrives in time for decoding, the receiver is be able to recover the lost packet; if no packet is lost, the sender is able to increase the media encoding rate by swapping out the FEC. Figure 5.6(a) shows the endpoint adding FEC and then swapping it for additional media. This method can be especially useful when the sending rate is close to the bottleneck link rate: by choosing an appropriate FEC rate, the endpoint is able to probe for available capacity without affecting the user-experience because the media encoding rate remains constant.



(a) OWD=50ms **(b)** OWD=100ms

(c) OWD = 240ms

Figure 5.7. Performance of a single RTP flow using FBRA in a varying link capacity scenario with different bottleneck latencies. The plots also show the FEC probing rate. We observe that the FEC rate is low when the FBRA rate drops and FEC rate is high when the FBRA is ramping up.

Figure 5.6(b) illustrates the state machine of a congestion-control algorithm incorporating FEC. The state machine includes 4 states: **STAY**, **PROBE**, **UP**, and **DOWN**. The congestion control monitors the congestion cues (such as, RTT, packet loss or discard rate [38, 96, 132], jitter, frame inter-arrival delay variation, post-repair [17, 128] etc.) to stay in the current state, or transit to another. The state machine specifies only a generic description of path conditions for the transition between states and leaves the interpretation to the underlying congestion control algorithm.

After enabling FEC for an interval, one of the following three conditions may occur: 1) No more bandwidth is available and the sender keeps the current sending rate (**STAY state**), 2) Stable conditions are detected and the sender increases the sending rate and disables FEC (**UP state**), and 3) Unstable conditions are detected, and the sender reduces the sending rate **DOWN state**. If FEC is disabled for two consecutive intervals and no change in path characteristics is observed, FEC is enabled (**PROBE state**) [130].

In Publication VI, we compare the performance of FEC-Based Rate Adaptation (FBRA), RRTCC and C-NADU. The bottleneck link capacity varies between 100 *kbps* and 256 *kbps*. In the scenario, we evaluate the reactivity and convergence of the rate-control algorithm to the available end-to-end capacity for different path latencies. Our simulations in *ns-2* [125] shows that, for the 50 *ms* and 100 *ms* bottleneck link latency, the goodput achieved by FBRA is comparable to RRTCC but with comparatively lower loss rates

Delay	Metric	FBRA	RRTCC	C-NADU
		avg. $\pm\sigma$	avg. $\pm\sigma$	avg. $\pm\sigma$
50ms	Goodput [kbps]	179.13 ± 2.26	181.8 ± 3.11	165.42 ± 3.87
	Loss rate [%]	1.23 ± 0.28	4.27 ± 0.78	0.34 ± 0.11
	No. of lost frames	441.43 ± 82.37	1842 ± 25.4	93.67 ± 29.64
100ms	Goodput [kbps]	172.83 ± 2.74	172.48 ± 6.6	163.84 ± 3.11
	Loss rate [%]	1.72 ± 0.37	3.09 ± 0.85	0.17 ± 0.09
	No. of lost frames	562.83 ± 103.44	740 ± 42.82	46.4 ± 22.94
240ms	Goodput [kbps]	144.89 ± 8.35	169.22 ± 5.68	153.52 ± 6.81
	Loss rate [%]	2.82 ± 0.89	2.98 ± 0.55	0.19 ± 0.07
	No. of lost frames	789.93 ± 223.55	705.67 ± 41.33	53.23 ± 21.41

Table 5.2. Overall metrics for an RTP flow on a variable capacity link. Results are the average of 30 runs.

Delay	Metric	Call 1	Call 2
		avg. $\pm\sigma$	avg. $\pm\sigma$
50ms	Goodput [kbps]	375.39 ± 88.25	348.77 ± 83.64
	Loss rate [%]	1.21 ± 0.19	1.39 ± 0.69
	FEC rate [kbps]	12.24 ± 1.64	11.78 ± 1.39
	No. of FEC protected lost frames	15.3 ± 3.44	14.6 ± 2.73
	No. of recovered frames	7.4 ± 2.83	6.9 ± 3.33
	PSNR [dB]	38.08 ± 2.1	37.7 ± 1.53
100ms	Goodput [kbps]	295.33 ± 48.27	351.1 ± 63.4
	Loss rate [%]	3.15 ± 0.93	2.33 ± 0.87
	FEC rate [kbps]	10.7 ± 0.68	11.69 ± 1.52
	No. of FEC protected lost frames	3.0 ± 2.1	4.1 ± 3.14
	No. of recovered frames	7.3 ± 3.03	4.5 ± 3.07
	PSNR [dB]	35.64 ± 1.17	37.32 ± 1.65

Table 5.3. Two FBRA flows on a bottleneck link in an emulated testbed. Results are the average of 10 runs.

($\approx$1.5 %, see Table 5.2). Figures 5.7(a)–(b) show that the FBRA can quite quickly bounce-back after undershooting. The figure also shows that the FEC probing rate increases when the FBRA ramps up and the FEC rate is low or disabled when the FBRA undershoots. However, FBRA is primarily a delay-based control algorithm, and for the 240 *ms* bottleneck delay, it observes that the packets are arriving very close to the application-defined maximum delay ($\approx$400 ms) and is therefore, conservative in its probing for available bandwidth (this is observed by the low FEC rate in Figure 5.7(c)). The FEC rate is about 10 % of the media rate at about 15-20 *kbps* and the packet loss recovery is around 30 %. Compared to FBRA, RRTCC tries to achieve high throughput at the cost of incurring packet losses (this behaviour is also observed in 4.3), while C-NADU trades off packet loss for throughput. In our experiments, FBRA's performance is placed between the two trade-offs: better throughput than C-NADU and lower loss ratio than RRTCC.

In Publication VI, we also measure the performance of two FBRA calls competing on a bottleneck link (1 *Mbps*) in a testbed. The difference in the goodput of the calls in the two latency scenarios (50 *ms* and 100 *ms*) is about 30-50 *kbps*, but the PSNR of these calls is similar (see Table 5.3). Therefore, as far as we can tell from PSNR, the small rate variations have

Metric	**50ms delay**	**100ms delay**
	avg. $\pm\sigma$	avg. $\pm\sigma$
Goodput [kbps]	302.24 ± 87.07	280.97 ± 92.15
Loss rate [%]	4.24 ± 0.89	4.1 ± 0.58
FEC rate [kbps]	13.6 ± 2.15	12.58 ± 2.18
No. of lost frames	154.6 ± 16.56	170.9 ± 12.38
No. of FEC protected lost frames	38.0 ± 6.08	23.7 ± 8.99
No. of recovered frames	7.4 ± 2.83	6.9 ± 3.33
PSNR [dB]	35.62 ± 1.49	34.7 ± 2.26
TCP throughput [kbps]	612.22 ± 48.45	575.11 ± 45.67

Table 5.4. An RTP flow sharing a bottleneck link with short TCP flows in an emulated testbed. Results are average of 10 runs.

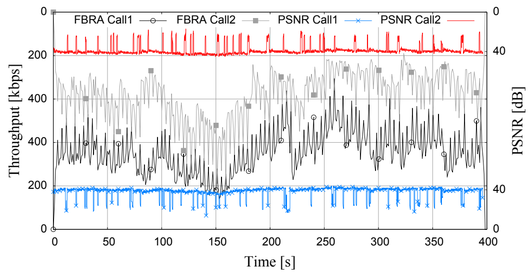
little bearing on the quality of the call. The FEC recovery is low due to the small amount of packet loss, and most of the losses occur when FEC was disabled. Additionally, 90 % of the times FEC was enabled resulted in an increase in media rate. Low FEC overhead also implies that the FBRA remains longer in the *STAY* state, thus avoiding abrupt changes to the encoding rate, which is detrimental for user experience [167].

When competing with short TCP flows, the FBRA achieves an average goodput of 302 *kbps* and 280 *kbps* in the 50 *ms* and 100 *ms* latency scenario, respectively (see Table 5.4). Additionally, about 85 % of the times FEC was enabled, resulted in an increase in media rate. The TCP flow achieves a throughput of around 600 *kbps* on average. The PSNR in both scenarios are very similar ($\approx$ 35dB), but the PSNR is lower for the 100 *ms* delay scenario because the average goodput is also a bit lower in this case (see Figure 5.8).

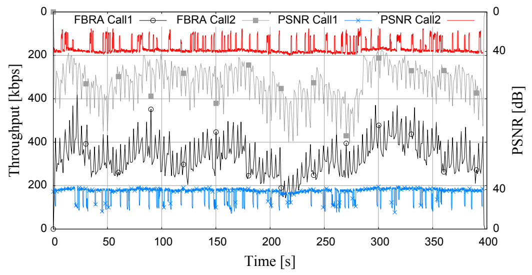
5.3 Summary

In this chapter, we address two problems: applicability of error-resilience schemes and the use of FEC to probe for available capacity in a multimedia congestion control algorithm. In the first case, we show that NACK, RPS, UEP/FEC and adaptive slice-size can be used for different levels of latencies and observed packet loss ratio. We also observed that sending smaller packets in mobile networks performed better than sending MTU-sized ($\approx$ 1500 bytes) packets. Alternatively, the MTU sized packets were better suited for wired or fixed networks where bit errors occurred less often.

In the second case, we propose a congestion control scheme where instead of increasing the rate when network conditions seem stable, the sender introduces FEC for one RTCP interval. If the FEC and the media packets are received successfully, sender rate increases by the amount of the FEC rate. The trade-off is that we get a smoother ramp up and, if a packet gets



(a) 50ms



(b) 100ms

Figure 5.8. The plots show the goodput of two RTP calls sharing a common bottleneck. To illustrate amount of empty link capacity and how two flows push one another, we plot one of them on the reverse axis. The end-to-end path capacity is 1 *Mbps* in both delay scenarios and delays are 50 *ms* and *100ms*. The plot also shows the PSNR variation for the two calls (on the minor Y-axis).

lost, it may be recovered by the FEC packet. The sender also implements a variable FEC interval, i.e., it varies the number of packets for which FEC is generated. Hence, if the sender thinks that it is under utilising the link by a large margin, it introduces a shorter FEC interval (up to 33 % redundancy) and therefore ramps up quickly. Consequently, when the sender thinks that it is closer to the bottleneck link capacity, it introduces a longer FEC interval (up to 8 % redundancy) and therefore is conservative in probing for available capacity. Our experiments show that by using adaptive FEC for probing, the endpoints are able to recover 15-25 % of the lost packets, and, ≈90 % of the time, using FEC subsequently results in an increase in the media rate. These results are comparable to our earlier experiments using a fixed FEC interval for error-resilience, where we were able to recover 20-24 % of the lost packets. We believe using FEC for congestion control in interactive multimedia has not been explored in depth by the community, partly because interactive multimedia flows have very tight delay constraints and FEC may not arrive in time for recovering the packet.

This chapter concludes the discussion of congestion control based on *on-path* sources and *in-band* signalling. In the next two chapters, we will discuss the use of congestion cues from off-path sources to perform congestion control.

6. Mobility, Offloading, Multihoming, and Overlays

Figure 3.2 in Chapter 3 shows the structure of the congestion control framework described in this thesis. The framework categorises *off-path* sources and *in-band* signalling for implementing congestion control (corresponds to *Block C* in Figure 3.2), which are discussed in this chapter. This chapter is based on our work on Multipath RTP (MPRTP), which is documented in Publication VII, in [133], [134], [110], [109], [66], and [97].

In Publication VII, we propose design goals to implement a multipath protocol for multimedia, protocol details, a scheduling algorithm to send media packets over multiple paths, and a dejitter buffer implementation to play out packets smoothly even when the path skew is high. We evaluate the performance of the proposed mechanisms in diverse scenarios in our testbed. Lastly, we discuss the system consideration for deployment.

In [133], we describe the requirements, functional blocks and protocol formats to extend RTP for enabling multipath capabilities. However, [133] does not define a scheduling algorithm and therefore allows for multiple proposals. In [134], we describe SDP and RTSP extensions required to set up MPRTP sessions and also ICE extensions to advertise MPRTP interfaces and perform NAT traversal. [109] is an extension to ICE candidate checks and measures RTT and loss while testing for connectivity. Thus, providing early measurement metrics for each path before MPRTP sends any media on those paths.

In [66], we propose using a network topology with multiple distribution trees to distribute media streams in a very large video conference call with few active speakers and many passive participants listening in. The multiple distribution trees carry separate MPRTP subflows and participants are members of multiple distribution trees, but actively forward media flows to one of the distribution trees. Therefore, a node is an overlay node in a few (e.g., one) distribution trees and a leaf node in the rest. In the paper,

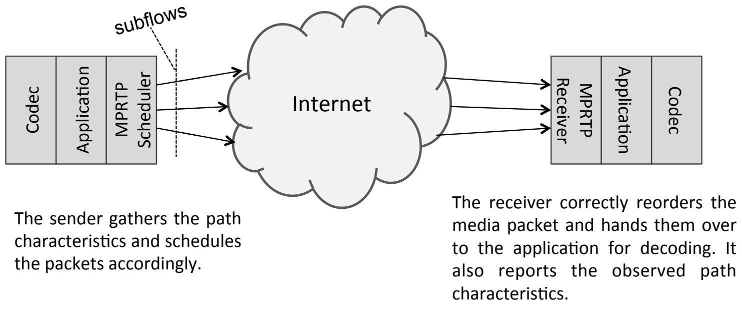

subflows
Codec
Application
MPRTP Scheduler
Internet
MPRTP Receiver
Application
Codec
The sender gathers the path characteristics and schedules the packets accordingly.
The receiver correctly reorders the media packet and hands them over to the application for decoding. It also reports the observed path characteristics.

Figure 6.1. System Overview: A sender uses multiple paths to stream media to a receiver. The receiver uses a dejitter buffer to reorder packets and sends per-path characteristics to the sender that distributes the packets based on the reported values.

we use a centralised focus to manage joining, leaving and inserting a participant in the appropriate position in the distribution tree. [97] is a work in progress and proposes protocol extensions to perform tree constructions in a distribution tree without the need for a centralised conferencing focus.

6.1 Multipath RTP (MPRTP)

The Internet backbone has evolved over the past decades into a mesh of service providers with manifold peerings that are generally capable of offering a number of (independent) paths between two nodes. Networks often use multiple attachment points for resilience purposes, such as data enterprise networks or data centers, and even routers for SOHO networks support multiple access networks [13, 28]. Additionally, many hosts today feature multiple network interfaces (e.g., WLAN and 3G on mobile devices). This may yield the opportunity for two endpoints to communicate via multiple paths. While exploiting multipath characteristics [154] has been explored for TCP (e.g., MPTCP [155, 58]), the requirements for real-time traffic differs notably and TCP can at best serve real-time communication within tight delay constraints of the network [21]. In the multipath case, the (MPTCP) scheduling algorithms do not consider real-time bounds when spreading data segments across different paths, and diverse paths may lead to worst-case delay and thus even longer buffering time.

We propose Multipath RTP (MPRTP) as a backwards-compatible extension to RTP [124]. It is documented in [133] and defines the basic mechanisms to operate across multiple parallel paths. Figure 6.1 shows a macroscopic system overview of MPRTP. The primary use-case for MPRTP is transporting media flows between multi-homed endpoints. Such end-

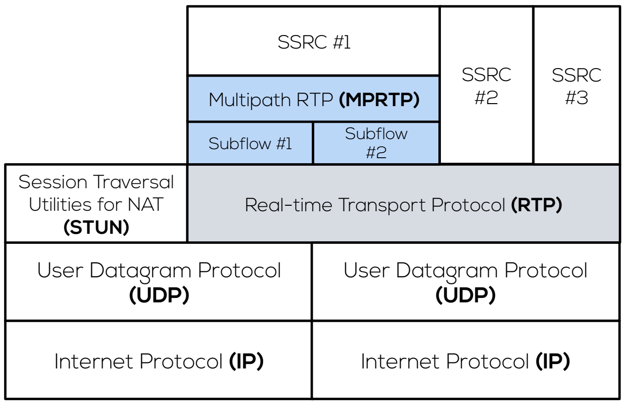

SSRC #1
Multipath RTP (MPRTP)
Subflow #1
Subflow #2
SSRC #2
SSRC #3
Session Traversal Utilities for NAT (STUN)
Real-time Transport Protocol (RTP)
User Datagram Protocol (UDP)
User Datagram Protocol (UDP)
Internet Protocol (IP)
Internet Protocol (IP)

Figure 6.2. The RTP and MPRTP stack working alongside each other. SSRC #1 uses MPRTP while SSRC #2 and SSRC #3 uses single path RTP.

points could be residential IPTV or telepresence devices that connect to the Internet through two different Internet service providers (ISPs), or mobile devices that connect to the Internet through 3G and WLAN interfaces. By allowing RTP to use multiple paths for transmission, the following gains can be achieved:

1. **`Higher quality`**: Pooling the resource capacity of multiple Internet paths allows higher bit-rate and higher quality codecs to be used. From the application perspective, the overall available capacity between the two endpoints increases.
2. **`Load balancing`**: Transmitting an RTP stream over multiple paths reduces the capacity usage on a single path, which in turn reduces the impact of the media stream on other traffic on that path. Also, seamlessly offloading a flow from one path to another allows for some gains such as reduced energy consumption, reduced access costs, or reduced network latency.
3. **`Fault tolerance`**: Using multiple paths in conjunction with redundancy mechanisms (FEC, re-transmissions, etc.), outages on one path have less impact on the overall perceived quality of the stream. This can also enable seamless handover in the case of mobility, i.e., moving from one network to another.

Figure 6.2 compares the network stack of a single path and a multipath-capable endpoints. SSRC #2 and SSRC #3 use a single path, while SSRC #1 uses multiple paths (with two subflows for the two interfaces). Subflow #1 and #2 are expected to flow over IP address #1 and #2, respectively. To discover its available interfaces, the multimedia application either uses the ICE procedures (hence, STUN) or implements a similar lightweight interface discovery process.

The design goals for MPRTP from our perspective are: an MPRTP-enabled system that makes use of multiple paths and adapts to their relative capacity changes by redistributing the load. As different paths will likely exhibit different RTTs, mechanisms must be developed to overcome the resulting skew. Furthermore, the choice of suitable transmission paths should reflect the demands of the application. From a protocol perspective, RTP must be extended to perform these functions, yet maintain backwards compatibility.

Specifically for multimedia, Liang *et al.* [89] show that transmitting redundant voice traffic over multiple paths performs better than an FEC-protected single stream. Chesterfield *et al.* [27] show that sending media over one 3G interface and UEP packets over a separate 3G interface can compensate for losses on the first path. Chebrolu *et al.* [26] propose capacity aggregation for multimedia applications by computing the earliest delivery time for each packet. They further propose to drop less important frames (e.g., B-frames) if the available capacity is smaller than the current encoding rate [25]. Jurca *et al.* [82] propose a frame-aware scheduling algorithm that sends key frames and other important media packets over less lossy paths, and this approach is similar to the one proposed in this paper. However, they also propose sending future packets over high-latency paths by reading ahead in the media stream. While this is an interesting concept, it would require larger buffers and keeping more state at the sender (typically, RTSP servers) to read ahead the stored media stream, which would not work for interactive multimedia and live media streams where the application cannot read ahead (into future packets).

Our proposed scheduling algorithm in Publication VII calculates the per path rate based on the following: **a)** the characterisation of the path based on the observed network behaviour, **b)** the choice of performant paths from the available paths for active transmission, **c)** packet scheduling rules that use the constraints applied by the multimedia application. We do not use B-frames and do not discard any packets at the sender. Furthermore, we try to maintain optimal playout by choosing paths that meet the latency constraints (<500 *ms*) and we try to maintain a very short dejitter buffer (hundreds of *ms*), so that the scheduling algorithm can be extended to include interactive applications.

6.1.1 Multipath Scheduling and Adaptive Playout

The scheduling algorithm at startup assigns equal fractional distributions and the per-path distribution changes depending on the observed path characteristics. Hence, the MPRTP sender calculates the estimated receiver rate for each path based on the Subflow Receiver Reports [133]. Next, the sender characterises the paths based on the observed packet discards and losses. From these paths, the sender chooses a set of *active paths* from the available paths. Lastly, it calculates the per-path fractional distribution.

A path that reports discards and losses in a single or consecutive intervals is considered *mildly congested*. If this behaviour is observed over three successive intervals, it is considered *congested*. Furthermore, if a path reports only losses and no discards in successive intervals, it is considered *lossy*. A path without losses or discards is considered *non-congested*.

A multipath sender chooses the paths that meet the media rate and latency requirements. Next, it groups the paths based on the path latencies–capacity is additive for paths with similar latencies [154]. Subsequently, it sorts the path groups in decreasing order of $\frac{capacity}{latency}$, so that groups of paths with high capacity and low latency are preferred. The endpoint chooses the set of paths from the groups that meet the encoded media's requirements and marks these paths as *'active'*; the rest are marked *'passive'*. The *'passive'* paths are used when the chosen paths begin to fail. Depending on the amount of packet loss (due to bit-errors), it may affect the quality of experience. Therefore, an MPRTP sender should avoid scheduling packets on paths with losses. The scheduler observes the following rules:

- If the next scheduled frame is an I-frame then the corresponding RTP packets are assigned to the path with the highest $\frac{capacity}{latency}$, path capacity and lowest loss rate.
- On receiving a NACK, transmit the requested packet on the path with the highest $\frac{capacity}{latency}$, least RTT and lowest loss rate.
- The *mildly congested* and *congested* paths get a smaller fractional distribution, in an attempt to reduce congestion on those paths.

To compensate for the difference in path latencies, the receiver calculates: 1) Packet skew based on the path jitter, 2) Path Skew, based on the media value of packet skew on each path, and 3) Playout Delay, based on the per Path Skew. First, the endpoint calculates the packet skew of each packet received on a path by subtracting the difference between reception timestamps (TR) and RTP timestamps (TS), $Packet\ Skew = (TR_j - TR_i) - (TS_j - TS_i)$, where 'i' and 'j' are consecutive packets received on a path.

For each path, the receiver maintains a Drift Window (DW), which is a sliding window of 2 seconds of media packets or 100 packets, whichever is lower. We chose a relatively small window size to prevent the receiver from under-flowing by changing the playout very late. Every time the endpoint receives a packet on a path, it calculates the drift and inserts it into the window. The receiver then sorts the window and chooses the median ($\widetilde{DW}$) value for calculating the path skew:

$$Path\,Skew_{now} = 0.01 \times \widetilde{DW} + 0.99 \times Path\,Skew_{prev}$$

The path skew values are then fed into the regular playout delay calculation [57, 100] to yield the playout delay applicable for multipath operation:

$$Playout\,Delay_{now} = \frac{MAX([SW]) + 124 \times Playout\,Delay_{prev}}{125}$$

We use the MAX value of the Skew Window (SW) instead of the median because we want to include the high latency path as soon as possible.

Depending on the fractional traffic distribution and RTT per path, our experiments show that our proposed method performs better in adapting the playout quickly. It takes some $3\,s$ to adapt the playout, while the method implemented in [72, 57, 100] takes 15-20s. Also, our algorithm converges more quickly than the receiver can report the RTT to the sender; the typical RTCP interval is $5\ \pm\ 2.5$s.

6.1.2 Comparing MPRTP to single path RTP

In Publication VII, we show that the performance of an MPRTP endpoint does not deteriorate compared with the performance of a flow that use just a single interface. In our experiment in the testbed, we use the results from a single-path media flow as the benchmark for comparing the performance of MPRTP. Table 6.1 shows that the performance of endpoints implementing MPRTP compared with a single path is not adversely affected. When none of the paths exhibit any losses, the performance of MPRTP was exactly the same, except that MPRTP induces an overhead because it uses an additional extension for identifying, monitoring and reporting subflows.

	$PSNR_{avg}$	σ_{PSNR}	PLR
1-Path (no loss)	48.427	0.00	0.00
2-Path (no loss)	48.427	0.00	0.00
3-Path (no loss)	48.427	0.00	0.00
Variable losses per path			
1-Path (0.5 % loss)	40.887	0.506	0.49
1-Path (1 % loss)	36.172	0.705	1.01
2-Path (0-0.5 %)	43.4	1.9	0.24
3-Path (0-1.0 %)	40.5	0.49	0.48
Variable RTT per path			
2-Path	48.303	0.278	0.004
3-Path	48.164	0.32	0.0121

Table 6.1. Comparing performance of using a single path with using multiple paths.

In our experiments in Publication VII, the RTP overhead for a $1\,Mbps$ media flow is an additional $1.275\,kbps$ and the Multipath RTCP (MPRTCP) accounted for ≈70 % of the total RTCP bandwidth (≈$0.25\,kbps$). When we introduce losses, the PSNR drops for the single path; however, for the multipath case, the PSNR is significantly higher because the paths do not necessarily exhibit losses at the same instance in time. Hence, the MPRTP scheduling algorithm is able to redistribute the capacity, preferring the path with lower loss rate. Additionally, when the paths have dissimilar RTTs (up to $150\,ms$ of skew across paths), yet again the receiver is able to play out packets across all paths and performs (comparing PSNR) at par with the single path. The scheduling algorithm and the adaptive dejitter buffer to play out packets across different path skews is discussed in detail in Publication VII.

6.2 Call Establishment and NAT Traversal

When an endpoint wants to use multiple paths, offload traffic onto another path (or interface), or move between networks, it requires the endpoint to either change its IP address or use multiple IP addresses at the same time. Typically, an endpoint changing its IP address breaks some of the higher level protocols (e.g., TCP, RTP), unless the higher level protocol is designed to be oblivious to the changes in IP address (e.g., SCTP [139] or MPTCP [58]).

Typically, performing interface advertisement is tightly coupled with NAT and firewall traversal, which would be needed for each interface anyway. Endpoints implement NAT and firewall traversal using Interactive Connectivity Establishment (ICE) [115] procedures, which enable the endpoints to ascertain connectivity between themselves by performing connectivity checks before transmitting media. The endpoint usually advertises the multiple interfaces in SDP, which usually couples the interface advertisement to the offer/answer mechanism. The offer/answer mechanism is excessive in this case, because a declarative mechanism would suffice. The endpoint mainly wants to notify the other endpoints of its interfaces. Likewise, when multiple interfaces become available at the other endpoint, it would notify its peers.

To summarise, in [133], we define an *in-band* mechanism to advertise interfaces in RTCP. The endpoint is able to update its existing interfaces or advertise new ones, whenever the RTCP interval expires. Advertising in-band is mainly useful when the endpoints are not deployed behind NATs or the ICE agent works together with the MPRTP stack [153]. In [134], we define the *out-of band* mechanism in SDP. The endpoint in this case performs the first round of offer/answer exactly as it would do for a multimedia session using a single path, but indicating that supports MPRTP and containing multiple *ICE candidates*. Later, when the connectivity checks for more than one path are successful, each endpoint advertises its MPRTP interfaces. Irrespective of the presence of a NAT, in Publication VII we show that advertising the multiple interfaces *in-band* leads to a establishing the call (with MPRTP capabilities) more quickly than when advertising the same interfaces *out-of-band*.

In practice, however, it would depend on the specific application to decide which method it prefers. Applications may prefer the *in-band* mechanism for real-time communication where low latency is expected mainly because it follows a declarative model as opposed to the offer/answer model. Alternatively, applications may prefer to use the *out-of-band* mechanism when they have to use ICE and SDP anyway.

6.3 Offloading and Multihoming

In Publication VII, we focus on spreading a constant bit rate (CBR) media stream across multiple paths, for which we present a scheduling algorithm that allocates traffic based on path characteristics. We use an adaptive dejitter buffer at the receiver so that the endpoint can play back media packets from paths with diverse characteristics. In our experiments, the application configures the scheduling algorithm for a maximum end-to-end

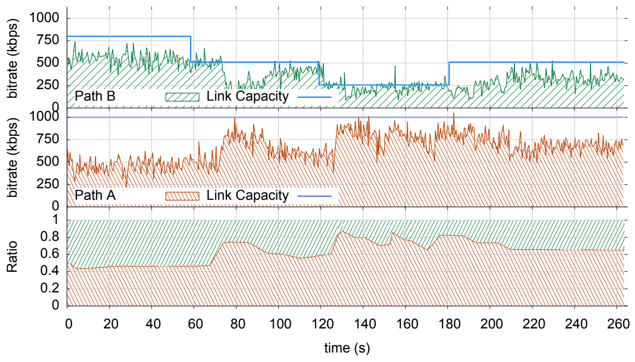


Figure 6.3. MPRTP offloading media from a path with changing capacity to another path with stable capacity and vice-versa.

latency of 500 ms and a maximum path skew of 200 ms. However, our work is orthogonal to rate adaptation–which would just change the aggregate media rate to spread across each subflow.

Offloading: In this scenario, the end-to-end capacity on one path is variable, demonstrating the sensitivity of the scheduling algorithm to the changes in network capacity, which may be caused by *cross-traffic*. In Figure 6.3, the Path B's capacity varies while Path A's capacity remains constant. The figure also shows the instantaneous bandwidth utilisation for each MPRTP subflow. In this case where congestion is observed on Path B, the scheduling algorithm reallocates the media on to the other paths (see the points where the link rate drops).

Multihoming: Figure 6.4 shows the bandwidth utilisation of a WLAN and 3G path and the overall bandwidth distribution between the paths. The bandwidth is more evenly shared except when the 3G path is constrained and the scheduling algorithm offloads the remaining media on to the WLAN path. However, the algorithm does not quickly reallocate the bandwidth it took away from the link to avoid bandwidth oscillations. This is a useful feature for the scheduling algorithm because it can then use the passive or idle paths for fallback. Moreover, the 3G path encounters packet losses more often than on the WLAN path, which causes the scheduling algorithm to prefer sending more media over the WLAN path. Despite using two lossy paths, the PSNR of the media stream (see Table 6.2) in this scenario is close to that of the original stream.

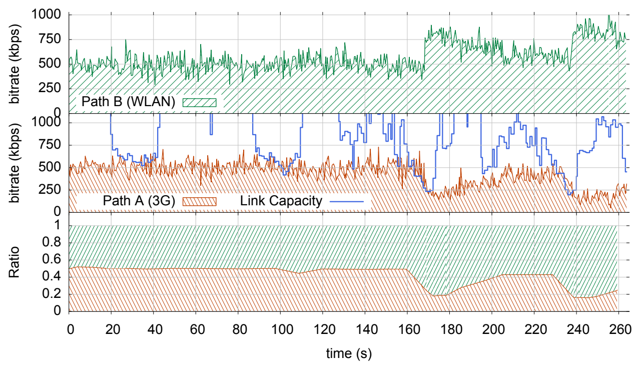


Figure 6.4. Multihomed endpoint load-balancing a media flow over WLAN and 3G paths.

Path Characteristic	$PSNR_{avg}$	σ_{PSNR}	PLR
Offloading	42.93	2.23	0.772
Multihoming	46.7173	0.21	0.33

Table 6.2. Performance of multipath scheduling when offloading (from a constrained path) and multihoming (with WLAN and 3G paths).

6.4 Summary

In this chapter, we propose using the multiple interfaces of an endpoint (e.g., mobile device, tablet, SOHO gateways) for increasing throughput and robustness. This corresponds to using congestion cues from out-of-band sources (different paths) and signalled in path (in RTCP). We design a protocol extension (MPRTP) to RTP that is backwards compatible and exploits multipath capabilities. We implement a scheduling algorithm that takes application requirements and current path characteristics into consideration to send packets over multiple paths. At the receiver, we implement a per-path and aggregate dejitter buffer, which attempts to playout packets smoothly even when the path skew is high. Our experiments show that the performance of MPRTP is not degraded compared to single path RTP, so that it is safe to deploy. It enables load distribution and capacity aggregation, which enables features like mobility, offloading, and multihoming.

In this thesis, we did not apply MPRTP for interactive real-time communication; however, we do use a fairly small playout buffer size on the order of hundreds of milliseconds, which is more apt for live streaming. The scheduling algorithm could be made to work with interactive real-time multimedia by coupling the scheduling algorithm with a congestion control algorithm. The congestion control algorithm would decide the aggregate path capacity and the scheduling algorithm proposed in this thesis would send the appropriate number of packets on each of the paths.

Energy usage is another aspect that was not considered explicitly in the evaluation and it is possible that using multiple interfaces may drain the mobile device's battery more quickly. A way to include the energy implications is to add energy awareness to the system policy; the multimedia application can then control the access to the particular interface depending on the battery level. This is similar to exposing any capacity or download restriction a user may have from their mobile operator. However, any further optimisation would require a deeper study of the energy consumption by a particular type of network interface (e.g., WLAN or 3G/LTE), and is left as a possible direction for future work.

7. Network-assisted Congestion Control

Figure 3.2 in Chapter 3 shows the structure of the congestion control framework described in this thesis. The framework categorises *on-path* and *off-path* sources and *out-of-band* signalling for implementing congestion control (corresponds to *Block B and D* of Figure 3.2). This chapter is based on our work on network-assisted congestion control, which is documented in Publication II, Publication VIII and [44].

In a 3G network, mobility, cell loading, handover and other factors can affect the throughput available to each user and the varying network capacity affects the video quality [47]. Deployments of GPRS, 3G and LTE show that there are still geographical areas where capacity or coverage is constrained [45, 138]. These constrained geographical areas may occur due to fading and interference from large building structures or closed inaccessible areas (e.g., tunnels, boats on lakes or in the archipelago, rural areas).

In Publication II, when the available link capacity changes at a base station, it notifies the endpoints connected to it about the current capacity. Based on these notifications, the endpoint adapts the media encoding rate. Hence, this paper covers the *on-path* sources and *out-of-band* signalling of the framework defined in Chapter 3. We evaluate the performance of employing network assistance for congestion control of interactive multimedia in a simulated environment using real-world 3G traces.

In Publication VIII, we explore the use of coverage maps for congestion control. The map server collects throughput information from the mobile clients, which also add geo-location information along with the throughput information. This assists the map server to build a bandwidth and coverage map. The server may be queried by mobile clients to predict coverage outage. This paper covers the *off-path* sources (e.g., coverage map) that signal congestion cues *out-of-band* to the endpoints. The paper

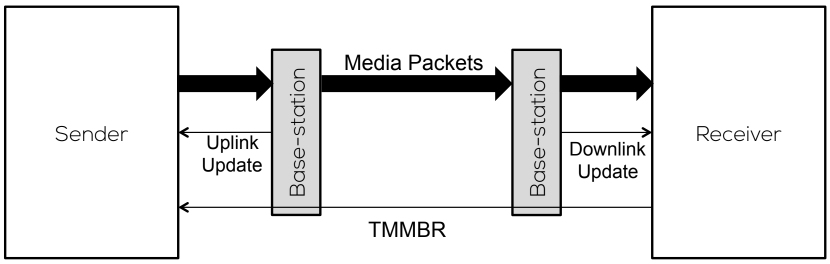


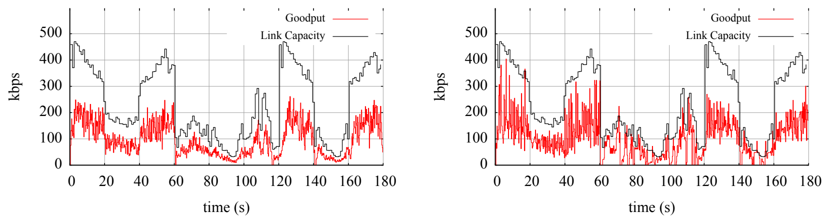
Figure 7.1. Endpoints receiving capacity indication from the middleboxes (TMMBR) for implementing congestion control.

also discusses the protocol and implementation aspects of such a service. Lastly, we evaluate the performance of using a coverage map for congestion control by collecting 3G traces and emulating it in our testbed.

7.1 On-path Congestion Cues

In some network deployments, routers along the media path are capable of detecting congestion before the queue overflows, typically, using active queue management (e.g., RED). A router marks a packet, indicating that the packet experienced congestion and that the router would soon drop packets for this flow [108]. The receiver on receiving this indication keeps a counter for the number of ECN-marked IP packets and signals it to the sender. For performing congestion control, the sender typically treats the count of ECN-marked packets as lost packets [150]. For example, the sender uses the sum of the reported loss events and the reported ECN events as the p (loss) value in the TFRC equation. Network-Assisted Dynamic Adaptation (NADA) [164] proposes a delay-based congestion control wherein the receiver measures congestion by aggregating the packet loss count, reported ECN markings, and one-way delay measurements into a single cue. The sender calculates the new rate based on the variation in the delay compared to earlier measurements and the priority of the multimedia stream [165].

In wireless networks, such as 3G/LTE, the last hop is typically the bottleneck because the core network is well provisioned. In Publication II, we implement a network-assisted congestion control scheme wherein base-stations (along the media path) notify the mobile terminals about the available link capacity over the wireless interface. In TMMBR-A, the base-stations notify both the sending and receiving endpoints about the available capacity, i.e., the sender is notified about the uplink capacity



(a) TMMBR-A **(b)** TMMBR-B

Figure 7.2. Performance of bandwidth indications by the base stations. a) TMMBR-A: both terminals are assisted, b) TMMBR-B: only the receiver is assisted.

and the receiver is notified about the downlink capacity. If an endpoint sends and receives data, it is notified asynchronously about the uplink and downlink capacity. Figure 7.1 shows a schematic representation of the interaction between the middleboxes and the endpoints. In TMMBR-B, the receiving endpoint is solely notified about the downlink capacity. Both TMMBR-A and TMMBR-B employ a co-operative congestion control scheme, wherein the receiver sends a TMMBR request to the sender containing the current downlink capacity. In TMMBR-A, the sender also receives a notification about the uplink capacity from the base-station; hence, comparing the request from the receiver and the notification from the base-station, the sender chooses the minimum of the two values as the new target media bit rate. In TMMBR-B, the sender calculates the *sender's estimate* (similar to the one described in 4.1.3) and chooses the minimum of the sender's estimate and the receiver's bit rate request.

Figure 7.2 shows the sample performance of TMMBR-A and TMMBR-B, where both endpoints are sending media over a 3G network. TMMBR-A, due to its knowledge about the network conditions at the uplink and the downlink, provides an average throughput of 180 *kbps* and 0 % loss, while TMMBR-B provides a comparable throughput of 178 *kbps* but with a 2 % loss ratio.

7.2 Off-path Congestion Cues

Modern mobile networks have been designed to carry multimedia streams with QoS traffic classes [3], but deployments of GPRS, 3G and HSDPA networks show that there are still geographical areas where best-effort traffic classes are used [45, 113]. To overcome these challenges, in Publication VIII, we implement a bandwidth coverage map that collects connectivity information from the users (*crowd-sourcing*) and calculates the available

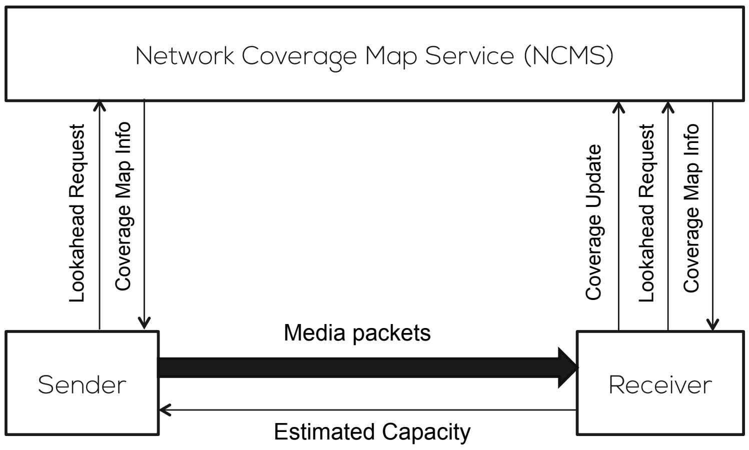


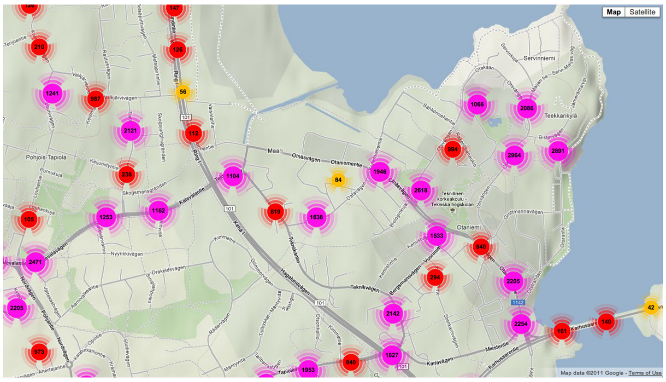
Figure 7.3. Endpoints using a Network Coverage Map Service to send coverage updates, query for upcoming congestion and receive coverage updates.

capacity at the reported geographical locations.

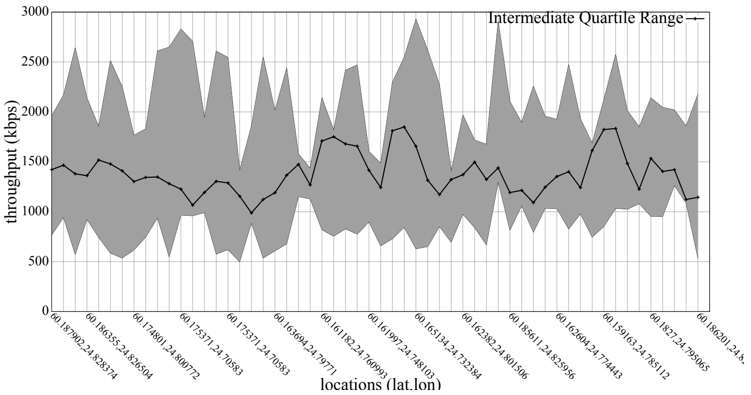
Service Maps are presented in [86] and the measurement-based approach is proposed in [10]. GPS-based congestion control is introduced in [157], and evaluated in different scenarios ([158], [159]), but they take signal strength as an influential factor for rate-control and show that predicting based on signal strength alone is insufficient. Yet, their results indicate that past information can be used to predict future network characteristics.

Riiser *et al.* [113] proposes a similar architecture (*bandwidth lookup service*) to the one in Publication VIII, but uses different types of averaging algorithms to predict future network characteristics in Dynamic Adaptive Streaming over HTTP (DASH). While the averaging algorithm is not a focus of this thesis, we use K-means [83] and K-nearest neighbour (K-NN) [77] algorithms to form regions with similar capacity. Details of the averaging algorithms employed by our coverage map server are discussed in [24]. A contrary approach is employed by *Netradar.org* [138] which divides the Earth's geographic map into 100m x 100m squares. All measurements in a specific area are averaged, and the maximum, minimum and mean values for each operator in that area is reported.

Riiser *et al.* [112] proposes fetching the bandwidth along a travel route in steps of 100 meters, which we find limiting. Instead, we propose multiple methods for discovering areas with poor connectivity (e.g., known travel route, area look ahead, geo-fencing or subscribing to coverage holes.) and show that not only looking up future bandwidth but also when to vary the sending rate affects the usefulness of the service. A preliminary analysis using simulations of our system is done in [45].



(a) Map area lookahead



(b) Known travel route Lookahead

Figure 7.4. a) A map of the available capacity around Otaniemi (university area). b) Example of the average and the Inter Quartile Range ($IQR = \pm 1\sigma$) throughput along a known route. The data in both representations were captured over several days.

Siris *et al.* [137] use bandwidth maps to offload data from WLAN to 3G and vice-versa, choosing the best interface to transfer streaming video, but does not use the interfaces simultaneously like MPRTP. Similarly, 3GPP's Access Network Discovery and Selection Function (ANDSF) standard [4] assists mobile endpoints to discover other access networks (e.g., WLAN) that may be used instead of the 3G/LTE access network. In this case, the operator needs to maintain a service map containing geolocations of WLAN access points.

In Publication VIII, we present a mechanism enabling an endpoint (usually a receiver) to proactively react to upcoming capacity limitations in wireless access networks. We enable this by implementing three steps, Figure 7.3 shows the interaction between the endpoints and the Network Coverage Map Server (NCMS). First, each endpoint receiving a media stream monitors the throughput and its geolocation and reports it to the NCMS. Figure 7.4 shows the throughput around the university area in

Espoo. Second, each endpoint is capable of querying the NCMS for upcoming congestion (known as *lookahead*). Using one of the following methods: known travel route, area lookahead or by subscribing to areas with poor coverage, typically using geo-fencing[16]. Figure 7.4 shows a graphical representation of the average throughput for an *area lookahead* and a *known travel route*. Third, for each lookahead requests, the NCMS responds with an coverage map info, i.e., the expected throughput at every geolocation along the route or requested area.

The receiver uses these hints provided by the NCMS to implement congestion control and sends an *estimated capacity vector* (a data structure containing a time-series of available throughput, thus preserving user's location privacy) to the sender, which can alter the sending rate based on the received information. The sender can use one of the following techniques to pick the sending rate: 1) adapt the encoding rate, 2) perform a rate-switch, or, 3) pre-buffer. Changing the encoding rate is only possible in interactive real-time media, because the RTP sender co-operates with the encoder to modify the encoding rate. Rate-switching is possible for streaming content encoded at different bit rates. It is also possible to use rate-switching for interactive real-time media involving Scalable Video Codecs (SVC) or simulcast media, wherein a media stream is encoded at different bit rates and the application switches between these different media streams. The performance of the congestion control in this case depends on the granularity of the chosen bit rates for the individual media streams. Lastly, varying the amount of pre-buffered video provides a more consistent experience to the user, because the technique attempts to maintain multimedia playback at a constant media encoding rate. This is only applicable to stored content since the multimedia application typically reads ahead in the media stream and sends more data much before before it detects poor coverage.

In Publication VIII, video is sent from a server to mobile clients. The user in the experiments mainly commuted around the city of Helsinki and Espoo using public transport. The NCMS collected the measured throughput for each client; thereafter, we conducted simulated experiments of users travelling at different vehicular speeds through the Helsinki region. In total, the NCMS collected over 400,000 updates over a month of operation (about 40-50 bus trips). The NCMS received more than 10,000 updates for 6 geographical areas (*Otaniemi*, Helsinki City Center) while on average

[16]Areas where the expected channel capacity is lower than the media bit rate

Method	SR_{avg}	$PSNR_{avg}$	σ_{PSNR}	PLR
No adaptation	865	27.48	4.55	6.6
Omniscient	929	43.12	1.9	0.33
Rate-switching	881	42.75	2.21	0.0
Late scheduling	1014	48.43	0.18	0.0

Table 7.1. A bus ride with good 3G coverage.

each geographical location had around 100 updates.

In Publication VIII, we describe a scenario where a user moves smoothly through a set of locations along a route and the receiver performs *lookahead* queries to fetch coverage information for surrounding areas. In this scenario, there are gaps in coverage but overall the throughput at most locations is above the required media rate. We analyse the performance of four different algorithms: 1) the *omniscient algorithm*, in which the endpoint is aware of the exact end-to-end channel capacity and is expected to perform the perfect rate-adaptation. 2) *no adaptation*, in which the endpoints perform no congestion control and the stream is transmitted at a constant bit rate. In the periods when the link capacity falls below the media rate, the receiver will observe an increase in losses/discards, which may result in frequent pausing of the video. 3) *Rate-switching*, in which the endpoints perform short *lookaheads*, which enables the endpoints to detect outages a few moments before the user enters a location with poor connectivity. In this case, the receiver sends a TMMBR message (with lowest available bit rate in the area) before entering the area and another one after the exiting the area (to reset the bit rate). The sender reacts by switching the media stream to the closest available media rate. 4) *Late scheduling*, in which the receiving endpoint searches for areas with poor coverage much before the user arrives at those locations, so that it can pre-buffer the content equivalent to the duration of the disruption. To reduce the impact of changing routes, the streaming client uses the congestion control algorithm proposed in Publication VIII.

Table 7.1 compares the results for the different schemes and Figure 7.5 shows the sending/receiver rate variation to channel throughput for one out of 10, simulation runs. The *omniscient algorithm* provides the best rate-adaptation but since it is unable to pre-buffer, the PSNR ($PSNR_{avg}$=43) is lower than that of late scheduling ($PSNR_{avg}$=48). Alternatively, performing *no adaptation* causes frequent pausing when the capacity is lower than the media rate, additionally, packet loss also affects the media quality and is reflected in the low PSNR ($PSNR_{avg}$=27). *Rate-switching* has

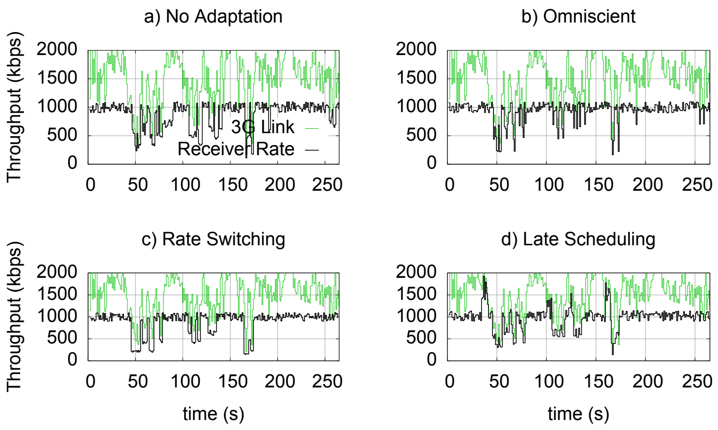


Figure 7.5. The plot shows the variation of sending rate to 3G link capacity based on a) no adaptation b) Omniscient c) Rate-switching d) Late scheduling when there are few coverage holes and good connectivity between them.

comparable results to the *omniscient* case because it avoids congestion by switching to bit rates lower than the minimum reported throughput. There are no losses and the average PSNR is 42.75, which is comparable to the *omniscient* case. *Late scheduling* outperforms the rest of the schemes in terms of media quality (measured in terms of average PSNR, $PSNR_{avg}$=48.43) mainly because it attempts to stream media at a single quality level; it relies heavily on pre-buffering, constantly updating the size of the pre-buffer to compensate for the longest disruption it finds along the travel-route or in the vicinity of the user. Another reason is that, in our measurements, we found low density of poor coverage areas and good connectivity in areas in between, allowing the client to pre-buffer content. However, when it is not possible to pre-buffer, the client will perform a rate-switch. Finally, we expect the performance of the congestion control to be at least that of rate-switching because the sender will be able to vary the sending rate preemptively without probing.

We find that the information provided by the NCMS is suitable for both predictive rate-switching and pre-buffering and helped in avoiding almost all packet losses in the scenarios we investigated, noticeably increasing video quality. This system works for media streaming and can be adapted for interactive media communication, wherein the sender employs a co-operative congestion control scheme, i.e., the sender receives the *estimated capacity vector* from the receiver and *coverage map information* for itself from the NCMS; it then picks the minimum of the two rates and implements any additional rate recommended by the congestion control (one of the algorithms discussed in Chapter 4).

7.3 Summary

In this chapter, we have described two congestion control mechanisms that uses out-of-band signalling, but use on-path and off-path sources, respectively. First, the on-path middleboxes (e.g., 3G or LTE base-stations) may notify the subscribed endpoints about bit rate changes using standard RTP extensions (e.g., TMMBR). The subscribed endpoints can use these bit rate (change) indications to modify the encoding rate in interactive multimedia applications or change the transmission rate in the case of video on demand services. Second, we introduce an off-path source, which is a crowd-sourced 3G coverage map that collects throughput and geolocation information from users, and it notifies the subscribed endpoints about the available capacity in their respective neighbourhoods. We propose a system to collect and query coverage maps. The endpoints query the coverage server using out-of-band signalling to discover areas of poor coverage. Similarly, based on these notifications the endpoints vary their encoding or transmission rate.

We show that notification from middleboxes such as base stations would help endpoints perform better congestion control, for example, when an LTE cell in a mobile network is experiencing extreme load[17]. Furthermore, we also show that using congestion maps also aids in performing congestion control. A congestion map is a measurement service that aggregates throughput information from multiple users and sends notification of areas with poor coverage to its subscribers. We expect these notifications to be used as congestion cues and not as a replacement to in-band, on-path congestion control. The main reason for restricting the use of such a service to notify congestion cues is that such a service is vulnerable to data pollution, i.e., the clients may report incorrect measurements, not necessarily intentionally but because of programming errors. Another reason is that, in the reported notifications may depend on the way the aggregation is performed and how quickly the aggregation converges to the prevailing network conditions in a region. A fast convergence may be susceptible to misreporting, leading to false positives and causing the user experience to fluctuate when in reality there would be no reason for it to fluctuate. A slow convergence would ignore minor spikes in load and endpoints would experience a poor quality of experience for a brief period until the values converge. A slow convergence may also lead to inefficiency

[17]Extreme load occurs when an LTE cell has about 10x more users than when it is busy.

because once an aggregation value in a region drops, it would stay there unless the endpoints probe for additional capacity.

While we show that NCMS can be used for streaming both the live and stored multimedia content, it can be adapted to work with interactive multimedia. However, we believe that the NCMS notifications would perform best when working alongside an existing congestion control algorithm.

8. Conclusions

In this thesis, we have described our proposed classification of congestion control cues (framework) in Chapter 3. When discussing the different parts of the framework, we highlighted the existing related work and our contributions in those areas. In terms of innovativeness, the congestion control framework provides options to look at congestion cues beyond those reported by the receiver in an RTCP Receiver Report. The framework allows the congestion control algorithm to use multiple paths for either aggregating capacity or for increasing error-resilience, or to use capacity notifications from middleboxes along the path, or to build a coverage map that provides congestion notification as a third-party services. Each of these techniques apply to one of the four areas described in the framework; however, the definitions of each area are generalised enough to allow the application of a broad range of techniques, not just those proposed in this thesis.

We defined a new class of co-operative congestion control algorithms and believe that these algorithms will replace the purely sender-driven mechanisms currently in use. The reason is that the co-operative algorithm lets the receiver not only measure the end-to-end capacity but assess the quality of experience (rendering quality) to decide its rate estimate. As with the sender-driven approach, the sender estimates the congestion based on the reported cues and its current sending rate, but the sender can then factor in the receiver's estimate before finally choosing a new target bit rate. We already see an uptake of this general idea, with Google's proposal for multimedia congestion control in WebRTC [8].

Another observation that we made during our research was that many multimedia systems implement the error-resilience and congestion control algorithms separately. We believe the community has not explored the use of FEC for congestion control in depth, partly because interactive

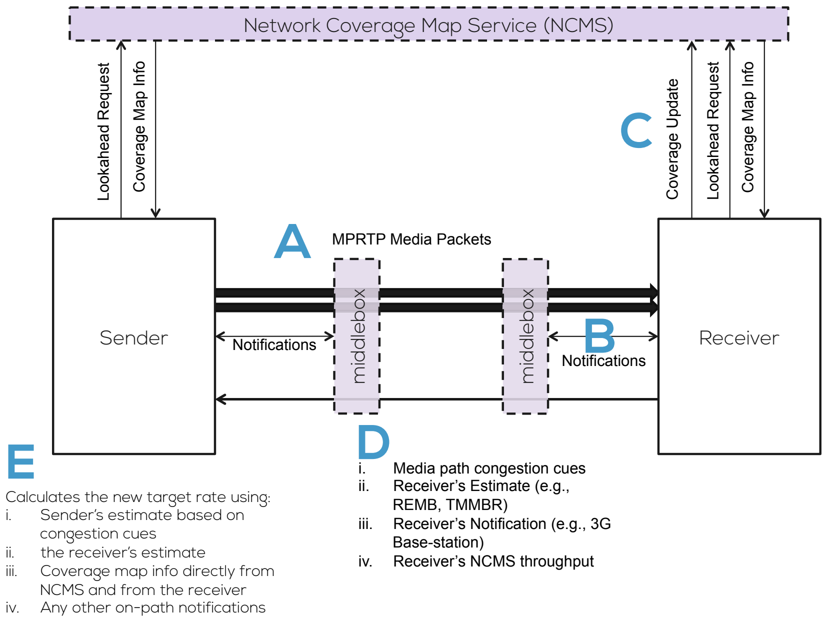

Network Coverage Map Service (NCMS)
Lookahead Request
Coverage Map Info
C
Coverage Update
Lookahead Request
Coverage Map Info
A
MPRTP Media Packets
Sender
Notifications
middlebox
middlebox
B
Notifications
Receiver
E
D
i. Media path congestion cues
ii. Receiver's Estimate (e.g., REMB, TMMBR)
iii. Receiver's Notification (e.g., 3G Base-station)
iv. Receiver's NCMS throughput
Calculates the new target rate using:
i. Sender's estimate based on congestion cues
ii. the receiver's estimate
iii. Coverage map info directly from NCMS and from the receiver
iv. Any other on-path notifications

Figure 8.1. Interworking of all the ideas presented in this thesis.

multimedia flows have very tight delay constraints and FEC may not arrive in time for recovering the lost packet. The results outlined in this thesis show that FEC can be used for congestion control and perform suitably when no bursty loss occurs.

8.1 Synthesis

While the techniques proposed in this thesis were only shown to work independently, nothing prevents them from interworking. Figure 8.1 depicts an example for a comprehensive architecture wherein an endpoint will be able to use cues from all the above sources to perform multimedia congestion control and packet scheduling. Endpoints will always use the multipath extensions for RTP (MPRTP), even when using a single path; this will allow the opportunity to offload or aggregate capacity when new interfaces (or paths) appear (Block A of Figure 8.1).

Since the circuit breaker algorithm relies on basic congestion cues (RTT and loss) and periodic reception of RTP and RTCP, the circuit breaker sets the boundary condition under which all future multimedia application will operate. The congestion control will not just rely on the cues reported in an RTCP RR/XRs, but gather hints from additional sources. For instance, mobile base stations can help provide information whenever capacity allocation changes due to mobility, an increase in active users, or handovers (Block B of Figure 8.1). Enabling Explicit Congestion Notification (ECN) in the network and getting ECN-marked packets is another way of enabling collaboration between the network and the endpoints. Lastly, using network coverage maps to get information about prevailing network conditions would also assist in congestion control (Block C of Figure 8.1). The main concern in this case is to ascertain the trustworthiness and reliability of the indicated measurement values. Since the additional congestion cues are just hints and part of a larger set of cues, an endpoint may ignore cues that seem erroneous or provide false indication than the other cues.

These additional hints are also essential at session startup, because currently a media application typically starts sending media at a preconfigured value (either set very low or set to the maximum media rate). Receiving notifications from a NCMS server at the beginning of the session may help the endpoint pick a better start rate.

Block D of Figure 8.1 shows that the receiver sends an RTCP RR and XR containing typical congestion cues. It also sends any estimate made by a receiver-side congestion control algorithm (e.g., using TMMBR, REMB). Furthermore, it adds any throughput notifications it may have received from an on-path middlebox (RTCP XR containing ECN markings or capacity changes indicated by a base station) or from a NCMS server. The sender decides the new target bit rate based on the congestion cues reported by the receiver, and the notifications it has also been receiving from an on-path middlebox or from the NCMS (Block E of Figure 8.1).

Finally, depending on the underlying codec implementation, the new target rate may result in: a change in the encoding rate of an audio and/or video stream, a change in the number of layers produced by a Scalable Video Codec (SVC), packetization time (ptime) of an audio stream, or a change in the number of simulcast multimedia streams (typically video streams).

8.2 Future Directions

Another aspect of MPRTP that is not discussed in detail in this thesis is the role of MPRTP in overlay networks, or for processing media in data centers. Very large conferences, with hundreds of participants can be arranged in complex topologies containing combinations of cascaded meshes and trees. Such very large media conferences (e.g., massive open online courses, seminars or conferences), usually have a low peer churn because all participants arrive and leave roughly at the same time. Also, the active participants produce the media flows, while the dormant/passive participants consume it and occasionally chime in with questions and comments. Instead of broadcasting to all the participants, which may create load on a centralised server and require scaling, we can exploit the asymmetric relationship between the participants by using them as overlays. This would require participants to forward at least as much as they receive, if not more, and using multiple paths eases this requirement [94, 88, 66].

In the near-term, we see the following elements of emerging. First, the emergence of WebRTC requires standardised congestion control. Chapters 4 and 5 make some proposals that may fit this purpose, but it requires more extensive evaluations to be deployment-ready. Second, using FEC for congestion control needs to be generalised to work with any other multimedia congestion control algorithm, to enhance the applicability of the congestion control. Third, multipath scheduling needs to be reconsidered for interactive multimedia communication. One way of achieving this would require implementing a coupled congestion control that synthesises all the subflow cues to arrive at an overall rate that would fit the combined paths and then let the current scheduling algorithm allocate the bits per subflow. Fourth, the scalability and robustness of the network coverage map service needs to studied; furthermore, the location-throughput matching algorithm needs to be studied in more detail to respond to diverse reporting from various mobile devices, etc. Lastly, all these techniques need to be combined for implementing a unified congestion control for interactive multimedia .

Bibliography

[1] 3GPP R1-081955. LTE Link Level Throughput Data for SA4 Evaluation Framework., May 2008.

[2] 3GPP S4-050560. Software Simulator for MBMS Streaming over UTRAN and GERAN, September 2005.

[3] 3GPP TS 23.107. Quality of Service (QoS) concept and architecture, December 2009.

[4] 3GPP TS 24.312. Access Network Discovery and Selection Function (ANDSF) Management Object (MO), June 2012.

[5] 3GPP TS 26.114. IP Multimedia Subsystem (IMS): Multimedia telephony; media handling and interaction, December 2009.

[6] 3GPP TS 26.234. Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs, December 2009.

[7] H. Alvestrand. Transports for RTCWEB, 2014. IETF Internet Draft.

[8] H. Alvestrand, S. Holmer, and H. Lundin. A Google Congestion Control Algorithm for Real-Time Communication. https://tools.ietf.org/html/draft-alvestrand-rtcweb-congestion, 2013. IETF Internet Draft.

[9] H. Alvestrand and V. Singh. Identifiers for WebRTC's Statistics API. http://dev.w3.org/2011/webrtc-stats/editor/webrtc-stats.html, 2014.

[10] O. Aravinda, K. Acharya, A. Sharma, E.M. Belding, K.C. Almeroth, and P. Papagiannaki. Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach. In *Proc. of IEEE SECON*, 2008.

[11] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing, August 2004. RFC3830.

[12] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for DiffServ Service Classes, August 2006. RFC4594.

[13] F. Baker. Exploring the multi-router SOHO network. https://tools.ietf.org/html/draft-baker-fun-multi-router, 2011. IETF Internet Draft.

[14] F. Baker, J. Polk, and M. Dolly. A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic, May 2010. RFC5865.

[15] H.V. Balan, L. Eggert, S. Niccolini, and M. Brunner. An Experimental Evaluation of Voice Quality Over the Datagram Congestion Control Protocol. In *Proc. of IEEE INFOCOM*, 2007.

[16] J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins, and Y. Xu. VP8 Data Format and Decoding Guide, November 2011. RFC6386.

[17] A. Begen, D. Hsu, and M. Lague. Post-Repair Loss RLE Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs), February 2010. RFC5725.

[18] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, December 1998. RFC2475.

[19] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview, June 1994. RFC1633.

[20] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, September 1997. RFC2205.

[21] E. Brosh, S.A. Baset, D. Rubenstein, and H. Schulzrinne. The delay-friendliness of tcp. In *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '08, pages 49–60, New York, NY, USA, 2008. ACM.

[22] L. Budzisz, R. Stanojević, A. Schlote, F. Baker, and R. Shorten. On the fair coexistence of loss-and delay-based tcp. *IEEE/ACM Transactions on Networking (TON)*, 19(6):1811–1824, 2011.

[23] M. Carbone and L. Rizzo. Dummynet revisited. *ACM SIGCOMM CCR*, Jan 2010.

[24] S. Chaterjee. Crowd-sourcing Mobile Internet Access Statistics to Improve Video Streaming, 1 2011.

[25] K. Chebrolu and R.R. Rao. Selective frame discard for interactive video. In *Proc. of IEEE ICC*, volume 7, june 2004.

[26] K. Chebrolu and R.R. Rao. Bandwidth aggregation for real-time applications in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 5(4), april 2006.

[27] J. Chesterfield, R. Chakravorty, I. Pratt, S. Banerjee, and P. Rodriguez. Exploiting diversity to enhance multimedia streaming over cellular links. In *Proc. of IEEE INFOCOM*, 2005.

[28] T. Chown, J. Arkko, A. Brandt, O. Troan, and J. Weil. Home Networking Architecture for IPv6. `https://tools.ietf.org/html/draft-ietf-homenet-arch`, 2012. IETF Internet Draft.

[29] Cisco. Dawn of the Zettabyte Era. `http://share.cisco.com/dawn-of-the-zettabyte-era.html`, 2012.

[30] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html`, 2013.

[31] A. Clark, K. Gross, and Q. Wu. RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting, January 2013. RFC6843.

[32] A. Clark, R. Huang, and Q. Wu. RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting, September 2013. RFC7003.

[33] A. Clark, V. Singh, and Q. Wu. RTP Control Protocol (RTCP) Extended Report (XR) Block for De-Jitter Buffer Metric Reporting, September 2013. RFC7005.

[34] A. Clark and Q. Wu. RTP Control Protocol (RTCP) Extended Report (XR) Block for Packet Delay Variation Metric Reporting, November 2012. RFC6798.

[35] A. Clark, Q. Wu, R. Schott, and G. Zorn. RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Mean Opinion Score (MOS) Metric Reporting, June 2014. RFC7266.

[36] A. Clark, S. Zhang, J. Zhao, and Q. Wu. RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Loss Metric Reporting, May 2013. RFC6958.

[37] A. Clark, G. Zorn, C. Bi, and Q. Wu. RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Concealment Metrics Reporting on Audio Applications, July 2014. RFC7294.

[38] A. Clark, G. Zorn, and Q. Wu. RTP Control Protocol (RTCP) Extended Report (XR) Block for Discard Count Metric Reporting, September 2013. RFC7002.

[39] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOMM*, Philadelphia, PA, USA, September 1990. ACM.

[40] D.E. Comer, D. Gries, M.C. Mulder, A. Tucker, A.J. Turner, P.R. Young, and P.J. Denning. Computing as a discipline. *Commun. ACM*, 32(1):9–23, January 1989.

[41] I.D.D. 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*, pages 66–71, 13-16 June 2005.

[42] I.D.D. Curcio and D. Leon. Evolution of 3gpp streaming for improving qos over mobile networks. *ICIP 2005: IEEE International Conference on Image Processing, 2005.*, 3:III–692–5, 11-14 Sept. 2005.

[43] I.D.D. Curcio, V. Singh, and J. Ott. Method and Apparatuses for Facilitating Determination of Receiver Buffer. `https://www.google.com/patents/US20120170469`, December 2010. US 12982190 (Patent Pending).

[44] I.D.D. Curcio, V. Singh, and K.M.V. Vinod. Method and Apparatus for Providing a Geo-Predictive Streaming Service. `https://www.google.com/patents/US8391896`, July 2010. US 8391896 B2 (Patent Granted).

[45] I.D.D. Curcio, V. Vadakital, and M. Hannuksela. Geo-predictive Real-time Media Delivery in Mobile Environment. In *Proc. of Mobile video delivery*, oct 2010.

[46] S. Dhesikan, C. Jennings, D. Druta, P. Jones, and J. Polk. DSCP and other packet markings for RTCWeb QoS, 2013. IETF Internet Draft.

[47] A. Diaz, P. Merino, L. Panizo, and A.M. Recio. Evaluating video streaming over gprs/umts networks: A practical case. In *Proc. of IEEE VTC*, pages 624–628, 2007.

[48] E. Elliott. *Estimates of error rates for codes on burst-noise channels*. Bell Telephone Laboratories, 1963.

[49] M. Ellis, C. Perkins, and D. Pezaros. End-to-end and network-internal measurements of real-time traffic to residential users. In *Proc. ACM MMSys*, San Jose, CA, USA, Feb. 2011.

[50] ETSI TR 102 643. Human Factors (HF); Quality of Experience (QoE) requirements for real-time communication services, November 2009.

[51] J. Fischl, H. Tschofenig, and E. Rescorla. Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS), May 2010. RFC5763.

[52] 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*, pages 43–56, New York, NY, USA, 2000. ACM.

[53] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification, September 2008. RFC5348.

[54] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *Networking, IEEE/ACM Transactions on*, 1(4):397–413, 1993.

[55] S. Floyd and E. Kohler. TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant, April 2007. RFC4828.

[56] S. Floyd, E. Kohler, and J. Padhye. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC), March 2006. RFC4342.

[57] D. Fober, Y. Orlarey, and S. Letz. Real time clock skew estimation over network delays, 2005.

[58] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses, January 2013. RFC6824.

[59] T. Friedman, R. Caceres, and A. Clark. RTP Control Protocol Extended Reports (RTCP XR), November 2003. RFC3611.

[60] H. Garudadri, H. Chung, N. Srinivasamurthy, and P. Sagetong. Rate Adaptation for Video Telephony in 3G Networks. In *Proc. of IEEE Workshop on Packet Video*, 2007.

[61] J. Gettys and K. Nichols. Bufferbloat: dark buffers in the Internet. *in Proc. of Communications of the ACM*, 55:57–65, Jan 2012.

[62] L. Gharai and T. Lehman. Experiences with high definition interactive video conferencing. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 433–436, 2006.

[63] L. Gharai and C. Perkins. Implementing Congestion Control in the Real World. In *Proc. of ICME '02*, volume 1, pages 397 – 400 vol.1, 2002.

[64] L. Gharai and C. Perkins. RTP with TCP Friendly Rate Control. `https://tools.ietf.org/html/draft-gharai-avtcore-rtp-tfrc`, 2011. IETF Internet Draft.

[65] Edgar N Gilbert et al. Capacity of a burst-noise channel. *Bell Syst. Tech. J*, 39(9):1253–1265, 1960.

[66] R. Globisch, V. Singh, T. Wiegand, and T. Schierl. Architecture for Asymmetric P2P Group Communication. In *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm)*, page 4. ACM, 2011.

[67] S Jamaloddin Golestani and Krishan K Sabnani. Fundamental observations on multicast congestion control in the internet. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 990–1000. IEEE, 1999.

[68] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification, January 2003. RFC3448.

[69] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol, July 2006. RFC4566.

[70] S. Hemminger. Network Emulation with NetEm. Proc. of the Linux Conference Australia, 2005.

[71] I. Hickson and D. Hyatt. HTML5: A vocabulary and associated APIs for HTML and XHTML. http://www.w3.org/TR/html5/, Sept 2010.

[72] O. Hodson, C. Perkins, and V. Hardman. Skew detection and compensation for internet audio applications. In *IEEE International Conference on Multimedia and Expo, 2000. ICME 2000.*, volume 3, pages 1687–1690 vol.3, 2000.

[73] ITU-T J.144. Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference . Fetched April 2010, 2004. Pre-published.

[74] ITU-T J.247. Objective perceptual multimedia video quality measurement in the presence of a full reference, 2008. Pre-published.

[75] ITU-T J.341. Objective multimedia video quality measurement of HDTV for digital cable television in the presence of a full reference signal. Fetched April 2010, 2011. Pre-published.

[76] ITU-T Rec. H.264. Advanced video coding for generic audiovisual services, June 2006.

[77] G.S. Iwerks, H. Samet, and K. Smith. Continuous K-nearest neighbor queries for continuously moving points with updates. In *Proc. of VLDB*, 2003.

[78] J. Jason, L. Rafalow, and E. Vyncke. IPsec Configuration Policy Information Model, August 2003. RFC3585.

[79] C. Jennings, T. Hardie, and M. Westerlund. Real-time communications for the web. *IEEE Communications Magazine*, 51(4):20–26, April 2013.

[80] R. Jesup. Congestion Control Requirements For RMCAT. `https://tools.ietf.org/html/draft-ietf-rmcat-cc-requirements`, 2013. IETF Internet Draft.

[81] I. Johansson and M. Westerlund. Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences, April 2009. RFC5506.

[82] D. Jurca and P. Frossard. Video packet selection and scheduling for multipath streaming. *IEEE Transactions on Multimedia*, april 2007.

[83] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. A local search approximation algorithm for k-means clustering. In *Proc. of SCG*, 2002.

[84] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP), March 2006. RFC4340.

[85] S. Kumar, L. Xu, M.K. Mandal, and S. Panchanathan. Error resiliency schemes in h.264/avc standard. *Journal of Visual Communication Image Representation*, 17:425–450, 2006 2006.

[86] D. Kutscher and J. Ott. Service Maps for Heterogeneous Network Environments. In *Proc. of MDM.*, May 2006.

[87] A. Li. RTP Payload Format for Generic Forward Error Correction, December 2007. RFC5109.

[88] J. Li, W. Lei, and H. Si. A Comparison of Multimedia Conferencing Frameworks. *Journal of Networks*, 5(6):740–747, June 2010.

[89] Y.J. Liang, E.G. Steinbach, and B. Girod. Multi-Stream Voice over IP Using Packet Path Diversity. In *Proc. of MMSP*, 2001.

[90] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, and J. Hildebrand. XEP-0166: Jingle, 2009.

[91] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN), April 2010. RFC5766.

[92] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997.

[93] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, December 1998. RFC2474.

[94] J. Noh, A. Mavlankar, P. Baccichet, and B. Girod. Reducing end-to-end transmission delay in p2p streaming systems using multiple trees with moderate outdegree. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 473–476. IEEE, 2008.

[95] P. O'Hanlon and K. Carlberg. Congestion Control Algorithm for Lower Latency and Lower Loss Media Transport. `https://tools.ietf.org/html/draft-ohanlon-rmcat-dflow`, 2013. IETF Internet Draft.

[96] J. Ott, V. Singh, and I. Curcio. RTP Control Protocol (RTCP) Extended Report (XR) for RLE of Discarded Packets, January 2014. RFC7097.

[97] J. Ott, V. Singh, J. Devadoss, and I.D.D. Curcio. RTP Control Protocol (RTCP) in Overlay Multicast. `https://tools.ietf.org/html/draft-ott-avtcore-rtcp-overlay-multicast`, 2012. IETF Internet Draft.

[98] 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), July 2006. RFC4585.

[99] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. In *ACM SIGCOMM Computer Communication Review*, volume 28/4, pages 303–314. ACM, 1998.

[100] C. Perkins. *RTP: audio and video for the internet*. Addison-Wesley Professional, first edition, 2003.

[101] C. Perkins. On the Use of RTP Control Protocol (RTCP) Feedback for Unicast Multimedia Congestion Control. `https://tools.ietf.org/html/draft-perkins-rmcat-rtp-cc-feedback`, 2013. IETF Internet Draft.

[102] C. Perkins and V. Singh. Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions. `https://tools.ietf.org/html/draft-ietf-avtcore-rtp-circuit-breakers`, 2013. IETF Internet Draft.

[103] C. Perkins and M. Westerlund. Securing the RTP Framework: Why RTP Does Not Mandate a Single Media Security Solution, April 2014. RFC7202.

[104] C. Perkins, M. Westerlund, and J. Ott. Web Real-Time Communication (WebRTC): Media Transport and Use of RTP. `https://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage`, 2013. IETF Internet Draft.

[105] T. Phelan, G. Fairhurst, and C. Perkins. DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal, November 2012. RFC6773.

[106] J. Postel. User Datagram Protocol, August 1980. RFC0768.

[107] J. Postel. Transmission Control Protocol, September 1981. RFC0793.

[108] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP, September 2001. RFC3168.

[109] T. Reddy, D. Wing, P. Martinsen, and V. Singh. Discovery of path characteristics using STUN. `https://tools.ietf.org/html/draft-reddy-tram-stun-path-data`, 2014. IETF Internet Draft.

[110] T. Reddy, D. Wing, B. VerSteeg, R. Penno, and V. Singh. Improving ICE Interface Selection Using Port Control Protocol (PCP) Flow Extension. `https://tools.ietf.org/html/draft-reddy-mmusic-ice-best-interface-pcp`, 2013. IETF Internet Draft.

[111] R. Rejaie, M. Handley, and D. Estrin. RAP: An End-To-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proc. of INFOCOM*, Mar 1999.

[112] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Trans. Multimedia Comput. Commun. Appl.*, 8(3):24:1–24:19, August 2012.

[113] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Bitrate and video quality planning for mobile streaming scenarios using a gps-based bandwidth lookup service. In *Proc. of ICME*, pages 1–6, july 2011.

[114] Luigi Rizzo. pgmcc: a tcp-friendly single-rate multicast congestion control scheme. In *ACM SIGCOMM Computer Communication Review*, volume 30, pages 17–28. ACM, 2000.

[115] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, April 2010. RFC5245.

[116] J. Rosenberg, A. Keranen, B. B. Lowekamp, and A. B. Roach. TCP Candidates with Interactive Connectivity Establishment (ICE), March 2012. RFC6544.

[117] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN), October 2008. RFC5389.

[118] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, June 2002. RFC3261.

[119] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core, March 2011. RFC6120.

[120] Z. Sarker, V. Singh, and C. Perkins. Circuit Breakers for Multimedia Congestion Control. In *Proc. of IEEE Workshop at INFOCOM*, page 6. IEEE, April 2014.

[121] Z. Sarker, V. Singh, X. Zhu, and M. Ramalho. Test Cases for Evaluating RMCAT Proposals. `https://tools.ietf.org/html/draft-ietf-rmcat-eval-test`, 2014. IETF Internet Draft.

[122] A. Saurin. Congestion control for video-conferencing applications. Master's thesis, University of Glasgow, December 2006.

[123] M. Schier and M. Welzl. Using DCCP: Issues and improvements. In *Proc. IEEE ICNP*, 2012.

[124] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, July 2003. RFC3550.

[125] Network Simulator. ns-2, March 2007. `http://www.isi.edu/nsnam/ns/`.

[126] V. Singh. Rate-control for Conversational H.264 Video Communication in Heterogeneous Networks, May 2010.

[127] V. Singh. Rate-control for rtp-based multimedia applications. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–4, June 2011.

[128] V. Singh and R. Huang. RTP Control Protocol (RTCP) Extended Report (XR) for Post-Repair Loss Count Metrics. `https://tools.ietf.org/html/draft-ietf-xrblock-rtcp-xr-post-repair-loss-count`, 2013. IETF Internet Draft.

[129] V. Singh, R. Huang, R. Even, D. Romascanu, and L. Deng. Considerations for Selecting RTCP Extended Report (XR) Metrics for the RTCWEB Statistics API. `https://tools.ietf.org/html/draft-ietf-xrblock-rtcweb-rtcp-xr-metrics`, 2013. IETF Internet Draft.

[130] V. Singh, M. Nagy, J. Ott, and L. Eggert. Congestion Control Using FEC for Conversational Media. `https://tools.ietf.org/html/draft-singh-rmcat-adaptive-fec`, 2014. IETF Internet Draft.

[131] V. Singh and J. Ott. Evaluating Congestion Control for Interactive Real-time Media. `https://tools.ietf.org/html/draft-ietf-rmcat-eval-criteria`, 2013. IETF Internet Draft.

[132] V. Singh, J. Ott, and I. Curcio. RTP Control Protocol (RTCP) Extended Report (XR) Block for the Bytes Discarded Metric, May 2014. RFC7243.

[133] V. Singh, J. Ott, T. Karkkainen, S. Ahsan, and L. Eggert. Multipath RTP (MPRTP). `https://tools.ietf.org/html/draft-ietf-avtcore-mprtp`, 2014. IETF Internet Draft.

[134] V. Singh, J. Ott, T. Karkkainen, R. Globisch, and T. Schierl. Multipath RTP (MPRTP) attribute in Session Description Protocol. `https://tools.ietf.org/html/draft-singh-mmusic-mprtp-sdp-extension`, 2013. IETF Internet Draft.

[135] V. Singh, J. Ott, and C. Perkins. Congestion Control for Interactive Media: Control Loops & APIs. In *IAB/IRTF Workshop on Congestion Control for Interactive Real-Time Communication*, July 2012.

[136] V. Singh, M. Zanaty, and A. Begen. RTP Payload Format for Flexible FEC. `https://tools.ietf.org/html/draft-ietf-payload-flexible-fec-scheme`, 2014. IETF Internet Draft.

[137] Vasilios A Siris, Maria Anagnostopoulou, and Dimitris Dimopoulos. Improving mobile video streaming with mobility prediction and prefetching in integrated cellular-wifi networks. *arXiv preprint arXiv:1310.6171*, 2013.

[138] S. Sonntag, J. Manner, and L. Schulte. Netradar - measuring the wireless world. In *Modeling Optimization in Mobile, Ad Hoc Wireless Networks (WiOpt), 2013 11th International Symposium on*, pages 29–34, 2013.

[139] R. Stewart. Stream Control Transmission Protocol, September 2007. RFC4960.

[140] M. Thornburgh. Adobe's Secure Real-Time Media Flow Protocol, November 2013. RFC7016.

[141] International Telecommunication Union. Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service, 2003. http://www.itu.int/.

[142] Lorenzo Vicisano, Jon Crowcroft, and Luigi Rizzo. Tcp-like congestion control for layered multicast data transfer. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 996–1003. IEEE, 1998.

[143] Y. Wang, S. Wenger, J. Wen, and A.K. Katsaggelos. Error resilient video coding techniques. *Signal Processing Magazine, IEEE*, 17(4):61–82, Jul 2000.

[144] Y. Wang and Q-F. Zhu. Error control and concealment for video communication: a review. *Proceedings of the IEEE*, 86(5):974–997, May 1998.

[145] Y-K. Wang, M.M. Hannuksela, and M. Gabbouj. Error resilient video coding using unequally protected key pictures. *2003 International Workshop on Very Low Bitrate Video (VLBV 2003)*, pages 290–297, Sept. 2003. Madrid, Spain.

[146] S. Wenger. H.264/avc over ip. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):645–656, 2003.

[147] S. Wenger, U. Chandra, M. Westerlund, and B. Burman. Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF), February 2008. RFC5104.

[148] S. Wenger and J. Ott. A joint source/channel coding: Approach for low-delay video. *submitted for review*, 2007.

[149] M. Westerlund, B. Burman, and L. Hamm. Codec Operation Point RTCP Extension. `https://tools.ietf.org/html/draft-westerlund-avtext-codec-operation-point`, 2012. IETF Internet Draft.

[150] M. Westerlund, I. Johansson, C. Perkins, P. O'Hanlon, and K. Carlberg. Explicit Congestion Notification (ECN) for RTP over UDP, August 2012. RFC6679.

[151] M. Westerlund and C. Perkins. Options for Securing RTP Sessions, April 2014. RFC7201.

[152] Jörg Widmer and Mark Handley. *Extending equation-based congestion control to multicast applications*, volume 31. ACM, 2001.

[153] D. Wing, T. Reddy, P. Patil, and P. Martinsen. Mobility with ICE (MICE). `https://tools.ietf.org/html/draft-wing-mmusic-ice-mobility`, 2013. IETF Internet Draft.

[154] D. Wischik, M. Handley, and M.B. Braun. The resource pooling principle. *SIGCOMM CCR*, 38, September 2008.

[155] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Proc. USENIX NSDI*, 2011.

[156] xiph.org. YUV Sequences, 2005. http://media.xiph.org/video/derf/.

[157] J. Yao, S. Kanhere, and M. Hassan. An empirical study of bandwidth predictability in mobile computing. In *Proc. of ACM WiNTECH*, 2008.

[158] J. Yao, S. Kanhere, and M. Hassan. Geo-intelligent Traffic Scheduling For Multi-Homed On-Board Networks. In *Proc. of ACM MobiArch*, 2009.

[159] J. Yao, S. Kanhere, and M. Hassan. Quality improvement of mobile video using geo-intelligent rate adaptation. In *Proc. of IEEE WCNC*, 2010.

[160] M. Zanaty, V. Singh, S. Nandakumar, and Z. Sarker. RTP Application Interaction with Congestion Control. `https://tools.ietf.org/html/draft-ietf-rmcat-app-interaction`, 2014. IETF Internet Draft.

[161] Q. Zhang, W. Zhu, and Y-Q. Zhang. Network-adaptive rate control with tcp-friendly protocol for multiple video objects. In *Proc. of ICME*, 2000.

[162] Q. Zhang, W. Zhu, and Y-Q. Zhang. Network-adaptive rate control and unequal loss protection with tcp-friendly protocol for scalable video over internet. *in Proc. of The Journal of VLSI Signal Processing*, 34:67–81, 2003.

[163] W. Zhu and Q. Zhang. Network-Adaptive Rate Control With Unequal Loss Protection For Scalable Video Over Internet. *in Proc. of Circuits and Systems*, 2001.

[164] X. Zhu and R. Pan. NADA: A Unified Congestion Control Scheme for Real-Time Media. `https://tools.ietf.org/html/draft-zhu-rmcat-nada`, 2013. IETF Internet Draft.

[165] X. Zhu and R. Pan. NADA: A Unified Congestion Control Scheme for Low-Latency Live Video. In *Proc. of IEEE Workshop on Packet Video*, 2013.

[166] P. Zimmermann, A. Johnston, and J. Callas. ZRTP: Media Path Key Agreement for Unicast Secure RTP, April 2011. RFC6189.

[167] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz. Subjective impression of variations in layer encoded videos. In *Quality of Service—IWQoS 2003*, pages 137–154. Springer, 2003.

[168] G. Zorn, R. Schott, Q. Wu, and R. Huang. RTP Control Protocol (RTCP) Extended Report (XR) Blocks for Summary Statistics Metrics Reporting, September 2013. RFC7004.

Aalto-DD 73/2015

Video is everywhere, it is the dominant traffic on the Internet! Disruptive Analysis Inc. believes that there are going to 2 billion active users in the world using video communication by 2019. This is based on the success of the ongoing standardization of the Web Real-time Communication (WebRTC) API and protocols by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). For WebRTC to be successful, it needs to be available to users on various devices (laptops, smartphones, tablets, set-top boxes), in diverse locations (office, home, while traveling at high speeds, in crowded areas). This requires the multimedia application (e.g., Hangout, Facetime, Skype, WeChat, ...) to adapt the media quality based on the application requirements, device capabilities, and prevailing network conditions. This dissertation designs, develops, and evaluates algorithms that tune the media characteristics to meet the above constraints.

9 789526 062204

ISBN 978-952-60-6220-4 (printed)
ISBN 978-952-60-6221-1 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Communication And Networking
www.aalto.fi