

Adaptive Live Streaming Strategies for Multi-homed Environments

by

Sharon Choy

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Sharon Choy 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Prof. Yashar Ganjali
Professor
Department of Computer Science
University of Toronto

Supervisor(s): Prof. Bernard Wong
Associate Professor
David R. Cheriton School of Computer Science
University of Waterloo

Internal Member: Prof. Samer Al-Kiswany
Associate Professor
David R. Cheriton School of Computer Science
University of Waterloo

Internal Member: Prof. Khuzaima Daudjee
Research Professor
David R. Cheriton School of Computer Science
University of Waterloo

Internal-External Member: Prof. Paul Ward
Associate Professor
Electrical and Computer Engineering
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This dissertation includes first-authored and peer-reviewed materials that have appeared in conference proceedings and journals published by the Institute of Electrical and Electronics Engineers (IEEE).

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Waterloo's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

We reuse our published articles in Chapter 3, 4, and 5, and we provide the full citation below:

Choy, Sharon, and Bernard Wong. "Obtaining Accurate Bandwidth Estimations for the Internet of Things." 2023 6th Conference on Cloud and Internet of Things (CIoT). IEEE, 2023. DOI: <https://doi.org/10.1109/CIoT57267.2023.10084909> [29] (Chapter 3)

Choy, Sharon, and Bernard Wong. "Evaluating Machine Learning Techniques for Predicting Link Instability in Wireless Networks to Support Live Video Streaming." 2023 19th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE, 2023. DOI: <https://doi.org/10.1109/WiMob58348.2023.10187772> [28] (Chapter 3)

Choy, Sharon, et al. "Conflux: A Multi-Homed Adaptive Bitrate Protocol for On-Site Live Video Streaming." IEEE Transactions on Networking (2025). DOI: <https://doi.org/10.1109/TON.2025.3577974> [27] (Chapters 4 and 5)

Components of Chapter 3 are patented under US11438265B2 (Packet transmission system and method), United States patent.

Abstract

The use of live video streaming applications over mobile, wireless networks continues to grow [136]. In these applications, a sender (e.g., cameraperson) streams video to a receiver (e.g., video service such as Youtube or a television station), and the receiver disseminates the video to the viewers. Live video streaming over wireless (3G, LTE, 5G) links is challenging since these links often experience fluctuating latency and available bandwidth. Furthermore, as the bandwidth demands of these applications continue to increase, a single, wireless network link may be unable to continuously stream video that satisfies the viewers' Quality of Experience (QoE) requirements. Multi-homing is a potential solution that offers applications additional bandwidth through aggregation and the ability to circumvent congested paths. Although there are a number of commercial multi-homed, adaptive live streaming solutions, their proprietary nature makes it difficult to understand their design trade-offs and evaluate their effectiveness. Furthermore, many multi-homed transport protocols are not designed for latency-sensitive data, do not adapt the video bitrate to changing network conditions, or require significant adjustments if used in a different environment from their original design (e.g., number or type of links). In this thesis, we explore solutions to the challenges of multi-homed, adaptive live video streaming.

Firstly, accurate bandwidth measurements are important because they indicate the maximum video bitrate that the available links can send while avoiding congestion. A packet train is a common measurement technique that sends a series of packets and uses the packets' inter-arrival time to measure the available bandwidth. However, packet train measurements are inaccurate due to kernel interrupts [66, 133, 98]. In this thesis, we propose our PacketBurst bandwidth measurement technique to improve the accuracy of the packet trains. This technique detects packet inter-arrival times that are affected by kernel interrupts and would yield inaccurate bandwidth measurements. Our PacketBurst technique excludes these packets from the bandwidth measurement calculation to improve the accuracy of the packet train. In addition to accurate bandwidth measurements, video streaming protocols can also benefit from future link quality predictions so that they may preemptively reduce the video bitrate or change network paths to avoid video stream disruptions. This thesis evaluates various machine learning techniques for classifying or predicting link instability, which we define as sudden increases in application-level packet loss and latency, or decreases in available bandwidth over a short, pre-determined period of time.

Secondly, using both video bitrate adaptation and multi-homing can greatly improve the live video streaming application's user QoE by avoiding congestion that results in excessive delay or by providing high video bitrates through link aggregation. In this thesis,

we present Conflux: a modular, multi-homed, adaptive video bitrate protocol for live video streaming. Conflux uses a probabilistic link quality model that is used in conjunction with a user-specific utility function to determine the video bitrate and the rate at which to send on each link. By using a simple, yet general, probability-based link quality model, Conflux can easily support different requirements and environments such as the number of links by just maximizing the expected utility. We evaluate Conflux in an emulated network environment where the available bandwidth comes from variety of wireless network traces [97, 118, 145], and the maximum available bandwidth is 38 *Mbps*. Our evaluation shows that Conflux can obtain at least an 18% improvement when there are two available links, and 65% when there are five available links over its non-optimal, multi-homed comparison systems that have oracle-based knowledge of either each link’s available bandwidth or the total aggregate available bandwidth.

Finally, resending delayed data on alternate, non-congested paths and sending redundant data using Forward Error Correction (FEC) can enable a video frame to arrive on time in the presence of deteriorating link quality or link failures. However, determining the degree of redundancy while effectively making use of the available bandwidth so that users can have high QoE is challenging. In this thesis, we introduce extensions to Conflux that support retransmission and FEC. We present our method of calculating the probability of on-time video data arrival for a given redundancy level using Conflux’s probabilistic link quality model. This allows Conflux to select the degree of redundancy and corresponding video bitrate that maximizes the user’s expected utility. Our evaluation shows that using Conflux’s redundancy-specific user utility functions lowers the percentage of incomplete frames 11% to 0.02% in network environments that experience packet loss according to the Gilbert-Elliot loss model.

Acknowledgments

My gratitude goes to my supervisor, Prof. Bernard Wong, for his guidance and mentorship during this PhD and for all that he has taught me. I am thankful for and will always cherish the memories of my graduate studies experience and of Bernard's advice and stories.

I would like to express my deepest gratitude and thanks and acknowledge my committee members: Prof. Yashar Ganjali, Prof. Samer Al-Kiswany, Prof. Khuzaima Daudjee, and Prof. Paul Ward for their helpful and insightful comments and feedback which has made this thesis stronger.

I would like to especially thank Prof. Khuzaima Daudjee for his collaboration for Conflux and Squash. I am especially grateful to him for introducing me to the area of Systems and Networking in CS454 during my undergraduate degree, and for his mentorship and advice throughout my time here at the University of Waterloo.

I am very grateful for the opportunity to collaborate with David Sze. Also, I am very thankful for the pleasure and privilege to work with Jooan Lee on Conflux and Squash.

I would like to express my gratitude to Prof. Paul Ward, Prof. David Taylor, and the members of Shoshin for their comments and feedback on my work during the weekly Shoshin meetings.

Many thanks to Ian O'Neill and Ethel Sithole for teaching me strategies on how to overcome adversity and challenges when completing this thesis.

I would like to thank Savio and Sarah Choi, and the brothers and sisters at Emmanuel Alliance Church of Ottawa for their prayers, care, and encouragement.

I am grateful for the many friendships that I have formed. Thank you to Adam Roegerist for discussions on Machine Learning and William Chan for his prayers and encouragement. I am also grateful for Aaron and Christina Moss, Dimitrios Skrepetos, and Nathan Fish; thank you so much for reading early drafts of this thesis and for all the lunches and memories that we have shared together.

I am greatly blessed by my sisters Katrina Mathurin, Heather Hogan, and Kimberly Flint. I am deeply thankful for all of the practice talks that they have listened to and for all the love that they have shown me over this last quarter century.

I would like to thank my family (Robin, Marie, and Ehren Choy) for their love and support throughout these many years, and to Wai-Fan and Amara Wong for their prayers and encouragement. Finally, words cannot express how thankful and grateful I am for

my husband Jonathan Ross. I am extremely blessed to have his love and support, and I am thankful for all his encouragement, for the practice talks he has listened to, and for traveling with me to conferences. I am grateful for all the care that he has shown me over the years.

Dedication

“Whatever you do, work at it with all your heart, as working for the Lord, not for human masters, since you know that you will receive an inheritance from the Lord as a reward. It is the Lord Christ you are serving.” - Colossians 3:23-24

Table of Contents

List of Figures	xiv
List of Tables	xvii
1 Introduction	1
1.1 Thesis Contributions	4
1.1.1 Link Quality Characterization	5
1.1.2 Video Bitrate Adaptation in Multi-homed Environments	8
1.1.3 An Exploration on Recovery Mechanisms	9
1.2 Outline	11
2 Related Work	12
2.1 Background	12
2.1.1 Live Video Streaming	12
2.1.2 Dynamic Adaptive Streaming over HTTP (DASH) Video	13
2.2 Link Quality Information	14
2.2.1 Bandwidth Estimation	14
2.2.2 Link Quality Characterization	16
2.3 Multi-homed Transport Protocols	17
2.3.1 MPTCP	18
2.3.2 MPQUIC	19

2.3.3	MP-DCCP	20
2.4	Multimedia Transport Protocols	21
2.4.1	Single-homed Multimedia Protocols	21
2.4.2	Multi-homed Multimedia Protocols	21
2.5	Adaptive Bitrate Protocols	23
2.5.1	Single-homed Adaptive Bitrate Protocols for Dynamic Adaptive Streaming over HTTP (DASH) Video	23
2.5.2	Multi-homed Adaptive Bitrate Protocols for DASH Video	29
2.5.3	Single-homed, Live Adaptive Video Bitrate Protocols	31
2.6	Latency-Sensitive, Multi-Homed, Adaptive Video Streaming	32
3	Link Quality Modelling	34
3.1	Packet Burst Bandwidth Estimation	35
3.1.1	Contributions	36
3.1.2	Background	36
3.1.3	Design	38
3.1.4	Evaluation	43
3.1.5	Real Testbed with Emulation	49
3.1.6	Discussion and Summary of PacketBurst	52
3.2	Machine Learning for Predicting Link Instability	52
3.2.1	Contributions	53
3.2.2	Trace Description	53
3.2.3	Classifying Link Instability	57
3.2.4	Evaluating Machine Learning Techniques	62
3.2.5	Discussion	68
3.2.6	Summary	69
3.3	Chapter Summary	70

4	Conflux: A Multi-homed Adaptive Bitrate Protocol for On-Site Live Video Streaming	71
4.1	Unique Challenges Found in On-Site Live Video Streaming	71
4.2	Conflux Protocol	73
4.2.1	Conflux’s Control Plane	73
4.2.2	Conflux’s Data Plane	81
4.3	Methodology	82
4.3.1	Traces	83
4.3.2	Metrics	84
4.3.3	Videos	84
4.3.4	Video Streaming Settings	85
4.3.5	Comparison Systems	86
4.4	Evaluation	88
4.4.1	Conflux vs. Comparison Systems	89
4.4.2	Conflux’s User Preferences	92
4.5	Conflux’s Interaction with Encoders	95
4.6	Chapter Summary	97
5	Recovery Mechanisms in Conflux	99
5.1	Conflux Extensions to Support Recovery	100
5.1.1	FEC-Enabled Configuration Finder	101
5.1.2	Blocksize Selection	103
5.1.3	Packet Retransmission	104
5.1.4	User Preferences	106
5.1.5	Extensions to Feedback Messages to Handle Packet Loss	107
5.2	Methodology for Evaluating Recovery Mechanisms	107
5.2.1	Traces	108
5.2.2	Metrics	109

5.2.3	Videos and Settings	109
5.2.4	User Preferences	109
5.3	Evaluation	110
5.3.1	Impact of Retransmission	111
5.3.2	Recovery Mechanisms in LTE Workloads	114
5.3.3	Environments with Application-Level Packet Loss	117
5.4	Chapter Summary	120
6	Conclusion	121
6.1	Contributions	121
6.2	Future Work	123
6.2.1	Directions in Link Quality Characterization	123
6.2.2	Development of User Preferences	124
6.2.3	Frame Awareness	124
6.2.4	Handling Rate Mismatch in the Encoder	124
6.2.5	Explicit Considerations for Retransmission and Packet Loss	125
6.3	Concluding Remarks	125
	References	127

List of Figures

3.1	Three packets (a) are sent on short succession and must be queued before traversing the bottleneck link (b). Packets are spaced out as they arrive at the receiver (c).	37
3.2	Packet Coalescing Example. The timestamps that are provided to the application are the service times (time from when the interrupt is generated to when the packet is actually serviced).	38
3.3	Bandwidth estimation using packet bursts	42
3.4	Packet Train Length - Bandwidth Estimates	44
3.5	Packet Train Length - Bandwidth Estimation Percent Error	45
3.6	Available Bandwidth Impact on Bandwidth Estimates	46
3.7	Available Bandwidth Impact on Bandwidth Estimation Percent Error	47
3.8	Testbed Setup	50
3.9	CDF of Measurement Intervals (ms)	54
3.10	Instantaneous Received Bandwidth (Recv BW)	55
3.11	Analysis on Packet Delivery Percentage	55
3.12	Analysis on Mean Latency	56
3.13	Analysis on Jitter	56
3.14	Analysis on RSSI	57
3.15	Example creation from raw traces.	59
3.16	Average and StdDev of F1 and Accuracy score for different ML techniques for different Link Instability definitions	63
3.17	False Positive Rate v. Classification Rate for DT, KNN, and SVM	63

3.18	False Positive Rate v. Classification Rate for SGD, LR, NB, and NN	65
4.1	Overview of Conflux. Solid arrow depicts the flow of video data. Shaded arrow depicts the control information.	73
4.2	Overview of Conflux’s Control Plane	74
4.3	Overview of Conflux’s Data Plane.	81
4.4	Video Bitrate v. Reference video VMAF (No Loss)	85
4.5	VMAF vs. %-tage Aggregate Available Bandwidth	87
4.6	CDFs of video quality metrics for Walk video. Line towards the right is better.	89
4.7	CDFs of video quality metrics for Walk video. Line towards the right is better.	91
4.8	VMAF of Different User Preferences. Four-link Mei et al. environment	92
4.9	% Incomplete Frames of Different User Preferences. Four-link Mei et al. environment	93
4.10	VMAF of Different User Preferences. Five links with periodic increases in latency and drops in available bandwidth.	94
4.11	% Incomplete Frames of Different User Preferences. Five links with periodic increases in latency and drops in available bandwidth.	94
4.12	% encoder error (average and 95% confidence interval) of each subsequent frame after target bitrate is reduced as indicated in the subcaption. Frame # X indicates the X^{th} frame since the change in target bitrate. Positive % encoder error indicates that the produced frame is greater than expected.	95
4.13	VMAF comparison of Conflux with is VBR Reduction Mechanism vs. Conflux with VBR Reduction Disabled.	96
4.14	% encoder error (average and 95% confidence interval) of each subsequent frame after target bitrate increases from the amounts as indicated in the subcaption. Frame # X indicates the X^{th} frame since the change in target bitrate. Negative % encoder error indicates that the produced frame is smaller than expected.	97
5.1	CDF of VMAF for Conflux with varying packet sizes	104
5.2	Comparison of Reserved and Actual sending rates for system with retransmission	113

5.3	Cumulative delayed data of the video stream	113
5.4	Comparison of Retransmission and No Retransmission	114
5.5	Available bandwidth follows Mei et al. workload and jitter is introduced every 30 seconds for 1 - 2 seconds	115
5.6	% incomplete frames in workload with significant drops in available bandwidth inspired by Fouladi et al. [46]	116
5.7	% incomplete frames for Loss-Aware Conflux in the Bocharova et al. [19] packet loss model	118
5.8	% incomplete frames for Loss-Aware Conflux in the Feng et al. [44] loss model	118
5.9	Loss-Aware Conflux in a random loss environment	119

List of Tables

3.1	Packet Train and PacketBurst (Average, Standard Deviation) Bandwidth Estimation Comparisons for 0.5 Mbps link	47
3.2	Packet Train and PacketBurst (Average, Standard Deviation) Bandwidth Estimation Comparisons for 20 Mbps link	48
3.3	Packet Train and PacketBurst Bandwidth Estimates Mbps (Average, Standard Deviation) in Mbps for Testbed with 100 ms latency, 10 ms jitter, and 20 Mbps	51
3.4	Packet Train and PacketBurst Bandwidth Estimates Mbps (Average, Standard Deviation) in Mbps for Testbed with 100 ms latency, 50 ms jitter, and 20 Mbps	51
3.5	(f-stat, p-value). Bold numbers indicate attributes that are selected for example construction.	59
3.6	RFE rank. Bold numbers indicate attributes that are selected for example construction.	60
3.7	Summary of the best Classification (Class) rate given a maximum False Positive (FP) rate. The first column is the Instability (I) definition. The second and third columns indicate the FP and Class rate respectively. The ML Technique description is (name, history size (obs), gap size (ms), horizon size (ms), boosting amount (%)).	64
3.8	Impact of Window Size on Classification Rate (CR) and False Positive (FP) Rate for different instability definitions and ML techniques (Tech.). This table presents the Average (Standard Deviation) over all configurations of the specified ML technique. Gap window (1000 <i>ms</i>), horizon window (2000 <i>ms</i>), and boosting level are fixed (25%).	66

3.9	Impact of Gap Size on Classification Rate (CR) and False Positive (FP) Rate for different failure definitions and ML techniques (Tech.). This table presents the Average (Standard Deviation) over all configurations of the specified ML technique. History window (5 observations), horizon window (2000 <i>ms</i>) and boosting level are fixed (25%).	66
3.10	Percentage of combinations that where different window size yielded significant differences in Classification (CR) and False Positive (FP) rates	69
4.1	Algorithm 1 Constants	75
4.2	Test Video Description	85
5.1	Example showing Link 1's and 2's probability values that at least i packets arrive (PSuccess) and exactly i packets arrive (PExact)	102
5.2	$PVideoFrameArrival$ for different values of k when $n = 6$	102

Chapter 1

Introduction

Recent trends have shown that live video consumption is becoming more popular and widely used in our society [9]. Examples of popular live video streaming applications include Youtube Live, InstaLive, Twitch, and on-site newscasting. These applications typically demand low delay and high video bitrate. Low latency is necessary so that the video stream remains relevant with respect to the current event, and high video bitrate is needed to achieve high video quality so that viewers can have their Quality of Experience (QoE) requirements satisfied.

In on-site newscasting, the video person captures and sends video to the television station for rebroadcast [49]. Traditional methods for streaming news events use production trucks which comprise an antenna that streams video over microwave technologies or satellite. These solutions are expensive and require a direct line of sight to the television station or the relaying satellite. Streaming live video over wireless (e.g., 3G, LTE, 5G) is an alternate solution that affords the on-site news reporter greater physical freedom as it requires less set up time, and it is also less expensive since it reuses existing infrastructure to deliver the video stream.

Although live video streaming over wireless networks has many advantages over using a television production truck, it is challenging because wireless networks experience highly variable available bandwidth and latency. This is primarily caused by fluctuating signal strength due to physical barriers (e.g., weather) [18] and by an influx of users since the available bandwidth is divided among all users in the same area [102, 163].

Adaptive video bitrate protocols that select the video bitrate based on numerous signals, such as network link conditions, can help the streaming application adapt to highly variable network conditions. Although there are many existing adaptive video bitrate protocols for

on-demand video streaming (Dynamic Streaming over HTTP or DASH), they may not be suitable for on-site live newscasting. Firstly, many adaptive video bitrate protocols for DASH rely on large buffers to mitigate the impact of changing available bandwidth and use buffer occupancy to decide the video bitrate. This is not suitable for live video streaming since live streams require low delay so that the content remains relevant to the viewer. Furthermore, DASH protocols require the video to be transcoded to multiple bitrates, and the adaptive video bitrate protocol selects one of these bitrates based on network conditions and buffer occupancy. However, this setup is not ideal for the live environment as there may not be sufficient time or resources, such as hardware encoders, to create multiple encodings.

Many of these adaptive video bitrate protocols are also designed to be only used with a single and stable wired link. Protocols that use a single link are referred to as *single-homed* protocols. In single-homed video streaming, adaptive video bitrate protocols have limited options to mitigate a drop in available bandwidth or an increase in delay. Using multiple wireless links greatly expands the potential options that an adaptive bitrate protocol can undertake as there are now opportunities for bandwidth aggregation and for circumventing congestion by avoiding poor quality links. These techniques can improve the video stream's on-time arrival rate and increase the stream's bitrate. Finally, multiple links can also be used to provide redundancy for the video stream. If the video stream is fully replicated, then the application would be able to tolerate complete link failures. Protocols that use multiple links concurrently to stream video and redundant data are referred to as *multi-homed* protocols.

A number of commercial solutions are currently available to news broadcasters that use multiple wireless links [143, 51, 35]. These solutions are proprietary, and many require specialized hardware. This makes it difficult to understand their design trade-offs and to evaluate their effectiveness. Several non-commercial solutions, such as FRA-JSCC [148] and SCTP [65], use link quality metrics (e.g., latency, jitter, loss, and bandwidth) to build link quality models for multipath scheduling and retransmission. These protocols may require additional adjustments when the deployment environment (e.g. number, types, and behaviour of links) differs from the protocol's design assumptions. Moreover, some of these protocols also depend on knowing the impact that bitrate and loss have on video distortion to determine packet scheduling and the degree of redundancy to use, and this may not be practical for live streaming because the video content is unknown. Generalized multi-homed transport protocols that offer in-order and reliable delivery guarantees across multiple links, such as MPTCP, are an alternative to the aforementioned protocols. Although these protocols do not require additional adjustments for different environments and provide the necessary guarantees for bulk data transfers, they may not be suitable

for live video streaming since they may experience more deadline violations than using a single link because they are susceptible to Head-of-Line (HoL) blocking when any of their links experience loss or congestion.

Designing a multi-homed protocol that can meet the user's strict latency requirements while making efficient use of multiple links is challenging. The protocol must determine the appropriate degree of redundancy to use for each video frame and the amount of data to send on each link to avoid deadline violations caused by congestion or signal loss on one or more links. The protocol must also be scalable with the number of links as current commercial devices can employ as many as eight wireless network interfaces [43, 35]. Although using multiple links provides many potential benefits, poor scheduling and video bitrate adaptation decisions may result in worse video quality than if a single link had been used. Aspects and factors that affect the live video stream include:

- **Acceptable delay:** Live video streams have low delay requirements so that their content remains relevant to the viewer. Video data or video frames that arrive past the user's maximum acceptable delay, which we refer to as a *deadline*, are no longer useful to the video stream because the event has passed. Missed deadlines result in visual artifacts and interruptions in the video stream.
- **Available bandwidth:** The available bandwidth determines the rate at which data can be sent and is an upper limit for the video bitrate. Generally, video streams with higher bitrates offer higher video quality and user experience than video streams with lower video bitrates [25].
- **Link stability or quality:** Ideally, the video stream should not have any visual artifacts or be interrupted while being viewed. In highly variable wireless network environments, the available bandwidth may drop such that it is less than the video bitrate. Consequently, these fluctuations may introduce latency that cause missed deadlines which result in visual artifacts and interruptions.
- **Amount of recovery or redundant data:** Multi-homing offers additional opportunities to use redundancy that can improve the on-time video arrival rate as data sent on a congested link still has the opportunity to arrive on time on another link. However, using redundancy reduces the amount of available bandwidth that can be used to obtain a higher video bitrate.
- **Video encoder settings and video content:** Single-pass video encoders, which are common for live streaming applications, do not generate data at a fixed rate. The

video encoder’s generated data rate may vary from their target bitrate. A protocol should adapt to the amount of video data that the encoder produces so that the video stream does not experience congestion if the encoder produces more than the expected amount of video data.

- **Financial costs:** Users may have different bandwidth usage budgets and may not want to expend data for higher bitrates or for redundancy. This is because sending data over wireless LTE links has an inherent financial cost. A live video streaming protocol should ensure that the user does not exceed its budget.
- **User preferences:** A user may want to trade off a higher bitrate in favour of a lower likelihood of experiencing congestion and vice versa, and this can be accomplished by either sending redundant data or sending at a lower bitrate. Ideally, a video streaming protocol should satisfy different user preferences without requiring the entire system to be redesigned.

Consideration for the aforementioned factors is imperative when creating customizable, multi-homed adaptive video bitrate protocols for delay-sensitive video.

1.1 Thesis Contributions

Constructing protocols for adaptive, live video streaming in the multi-homed environment requires addressing the following problems and research challenges:

Link Quality Characterization: A video streaming application would ideally stream at a bitrate that is equal to its link’s maximum available bandwidth. Accurately measuring a link’s maximum available bandwidth is challenging as many works have found that kernel interrupts can cause *interrupt coalescing* which result in measurement inaccuracies [85, 110]. In this thesis, we present a solution to improve the accuracy of bandwidth measurements by removing inaccuracies caused by interrupt coalescing.

If the link’s quality deteriorates between raw image capture and video frame generation, the video frame may not be able to arrive within its user’s delay requirements due to insufficient available bandwidth or increased link delay. Predicting the link’s ability to maintain its current available bandwidth and delay conditions is an important challenge to address as it enables the application to determine if it should preemptively reduce the video bitrate or move the video stream to an alternate path. We present our method of constructing attributes that capture the trends of various network metrics (e.g., bandwidth, latency,

signal strength, etc.) which we use to train machine learning models. We also evaluate the ability of statistical machine learning models to predict link instability which include drops in available bandwidth, increases in link delay, and the presence of application-level packet loss.

Video Bitrate Adaptation in Multi-homed Environments: There is an inherent trade-off between fully utilizing the link to obtain the highest possible video bitrate and quality and sending at rates lower than the links' available bandwidth to avoid congestion and improve the probability of on-time video data arrival. The amount of data sent on each link directly affects the video stream's maximum, achievable video bitrate. Determining the appropriate trade-offs that maximize end-user QoE is challenging in multi-homed environments as link quality is heterogeneous and changes quickly. In this thesis, we address the challenge of video bitrate adaptation in multi-homed environments to determine the trade-offs that provide the user with the highest expected user QoE.

Recovery Mechanisms: Recovery data, such as retransmitted or redundant data, can be used to alleviate the impact of network congestion and packet loss. However, sending additional data comes at a cost of a reduced video bitrate and additional bandwidth usage which has an inherent financial cost. Determining the amount of recovery data to use during the video stream is challenging as an adaptive video bitrate protocol must consider the links' quality and the trade-offs of using additional recovery data. We present a model that characterizes the benefit of using redundant data and show when recovery mechanisms are effective at preventing delayed video data.

1.1.1 Link Quality Characterization

Because bandwidth measurements and link quality prediction are important to consider when determining how much to send on each link, this chapter makes the following contributions:

PacketBurst Bandwidth Estimation

Existing bandwidth estimation techniques can be classified as either self-loading periodic streams or packet trains [110]. Self-loading periodic stream-based bandwidth measurement techniques [22, 66, 117, 154] temporarily saturate the available link until the maximum achievable bandwidth is determined by the presence of additional queuing delay or the lack of an acknowledgement. Using this technique for latency-sensitive data can temporarily congest the link, delay data, and cause undesired link-layer re-transmission. These

techniques are not suitable for measuring the available bandwidth while streaming live video because their use can cause video data to be delayed, which results in visual artifacts that lower user QoE. Alternatively, packet-train estimates send a group of packets in short succession of each other, and the bandwidth estimate is calculated using packet inter-arrival time. This technique is more suitable for live video streaming than self-loading periodic streams because it does not require congesting the link; thus, it has a lower likelihood of causing delayed video data. Furthermore, this technique can be easily integrated into the live video stream as applications can send the video data as a packet train.

Packet-train estimations rely on accurate packet arrival timestamps collected by the receiver. One option is to use hardware or Network Interface Card (NIC) timestamps, which are generated at the network adapter upon packet arrival [3]. However, NIC timestamping is not universally available on all adapters and devices.

Kernel timestamps are an alternative to hardware timestamps and are preferable to user timestamps as they do not reflect inaccuracies caused by CPU application scheduling. However, using kernel timestamps does not account for other handling delays such as the time between packet arrival and servicing. For example, as packets arrive at the NIC, the NIC generates a kernel interrupt; however, the packet's kernel timestamp refers to when it was serviced by the kernel, not when it arrived at the NIC. Due to the scheduling delay between packet arrival and kernel service, multiple packets may arrive before the first packet's interrupt is serviced.

Multiple packets serviced by a single interrupt are assigned the same timestamp or timestamps that differ by a few microseconds. Using this packet inter-arrival time for calculating the available bandwidth results in significant overestimation. We refer to this phenomenon as *packet coalescing*. The discrepancies between packet arrival and servicing are present in many bandwidth estimation techniques. In works such as [66], inaccurate packet-train measurements that exhibit packet coalescing are discarded. Discarding estimates results in wasted probing. Furthermore, if estimates are required by the application, discarding estimates may also result in the application having to use an older or stale estimate. This may result in bandwidth underutilization or congestion.

We present the PacketBurst bandwidth estimate technique that accounts for kernel interrupts when determining the available bandwidth. The main contribution of PacketBurst is to remove inaccuracies in packet train bandwidth estimate calculations that are caused by kernel interrupts. We observe that consecutive packets in packet trains may have exceedingly small differences in their receive timestamps. Using these timestamps in a packet train bandwidth estimate calculation can result in an inaccurate estimate. We identify four cases where kernel interrupts impact the packet train's receive timestamps,

and we show how to address the timestamp inaccuracies of each case using our PacketBurst technique. We show that PacketBurst estimates can reduce measurement errors of packet trains. When using six MTU-sized packets to estimate available bandwidth, the PacketBurst technique experiences an 2.75% error whereas packet trains experience a 21.6% error when the available bandwidth is 20Mbps.

Link Stability Prediction

To ensure a smooth, high-quality video stream while using highly variable links, many video streaming applications employ adaptive video bitrate protocols that predict the available bandwidth [94, 120, 171] and adjust the bitrate accordingly. Many of these works do not consider other aspects of link quality such as latency and loss. However, these aspects must also be considered because an increase in latency can cause video to arrive past its deadline and stall, and the presence of packet loss can cause artifacts in the video stream. The ability to predict *link instability*, which we define as sharp increases in latency and application-level packet loss, or sudden drops in available bandwidth over a short period of time, is advantageous for adaptive video bitrate protocols. These protocols can use link instability predictions to reduce their bitrate or to send redundant data to ensure that their data arrives on time. Furthermore, as multi-homing becomes more common [84], adaptive video bitrate protocols can use this information to determine when to move the video stream to another link. These link instability predictions can be made using ML techniques.

We evaluate various Machine Learning (ML) techniques for classifying or predicting link instability. To train and evaluate these ML techniques, we first construct attributes from raw wireless LTE traces. These attributes include statistical summaries (e.g., average and standard deviation) and trends (e.g., slope and correlation) of various network metrics. We then determine the attributes that provide the greatest predictive ability to construct our ML examples for training and testing. Using these ML examples, we evaluate the predictive capability of various ML techniques. We find that we can predict up to 46% of sudden bandwidth drops using a decision tree, 40%-100% of increases in application-level packet loss, and 40%-100% of spikes in latency while maintaining false positive rates below 0.1. In general, we find that Decision Trees and K-Nearest Neighbours techniques provide the highest classification rate while incurring low false positive rates.

Both bandwidth measurement and link quality predictions important tools in determining each link's sending rate which is fundamental for mutli-homed scheduling and video bitrate adaptation.

1.1.2 Video Bitrate Adaptation in Multi-homed Environments

Constructing a generalized and scalable protocol for video bitrate adaptation in multi-homed environments is challenging due to link heterogeneity and different end-user preferences. To address this challenge, we introduce Conflux: a multi-homed adaptive bitrate protocol for on-site, live video streaming over wireless networks. Conflux has a modular design that encapsulates link quality characterization, packet scheduling, and video bitrate adaptation into different, independent modules. This approach enables Conflux to support different configurations (e.g., number or types of links) or user preferences with changes that are localized to a single module.

This degree of modularity is possible from the use of a new link quality abstraction that is based on statistical measures which we refer to as the PSuccess function. This function maps sending rates to the probability that the link can support the the input rate while meeting the user-specified deadline. Using the PSuccess function for all of the available links, Conflux determines a *configuration*, which consists of the sending rate on each link and the video encoder’s bitrate, that maximizes the user’s utility based on their provided utility function. Utility functions are tailored to individual users such as high-action sportscasters who prefer high video quality or interactive streamers who prefer fewer dropped frames. Conflux differentiates itself from other multi-homed, live streaming protocols with its design as it can and adapt its protocol to different hardware settings, network types, and user requirements with minimal changes to the core framework. Furthermore, Conflux is able to scale with the use of additional links without requiring protocol changes.

Conflux constructs the link’s PSuccess function by observing its success in delivering data at different rates. We say that a link is unable to deliver at a rate if some of its video data cannot arrive by the user-specified deadline. Higher rates are explored through a multi-link optimized, non-disruptive approach that probes one link at a time using redundant data. This allows Conflux to continue to stream video on other paths and to avoid missed deadlines if the probed link experiences congestion. Probability values are used by Conflux to directly determine the expected improvement from changing the sending rate.

Using the links’ PSuccess functions, Conflux determines the sending rate on each link and the video bitrate. Conflux works in conjunction with the streaming application’s video encoder by providing it with the target video bitrate; however, the encoder may generate frames that are larger than requested [73, 82]. Immediately sending these frames may cause congestion, and buffering them increases their delay which could result in missed deadlines. To address this, Conflux dynamically reduces the target bitrate based on the sender-side buffer occupancy, and the target bitrate does not exceed the video bitrate as determined by Conflux. This causes the encoder to generate frames that are sufficiently

small for Conflux to drain the sender-side buffer. This mechanism effectively works around a limitation that is common in single-pass video encoders.

We have implemented the Conflux protocol as a standalone library that allows applications to support live video streaming according to their preferences. Conflux makes the following contributions:

- An abstraction that represents link quality as a probability, which allows our system to directly quantify the trade-offs of different streaming and encoding settings.
- A modular system design for multi-homed video bitrate adaptation that requires minimal changes for additional links, different types of links, and users with varying requirements.
- Approaches the video quality of optimal video streaming systems, and at least a 13% improvement in video quality, expressed using the Video Multimethod Assessment Fusion (VMAF) metric, over other non-optimal, multi-homed comparison systems that have oracle-based knowledge of either each link’s available bandwidth or the total aggregate available bandwidth.

1.1.3 An Exploration on Recovery Mechanisms

It is important to use recovery mechanisms such as retransmission and redundancy to ensure that video data arrives on time. This is because delayed data can cause visual artifacts in the video stream, and these visual artifacts result in significantly lower user QoE [139].

We investigate two main recovery strategies that can prevent the video stream from incurring delayed data: retransmission and redundancy in the form of Forward Error Correction (FEC). We introduce retransmission to Conflux by resending packets that have not been acknowledged within a pre-determined period of time (i.e., four round-trips on the link that the packets were sent).

We also introduce redundancy using FEC to proactively protect the video stream against deadline violations caused by unexpected delays and congestion on one or more of the used paths. By sending redundant video data on multiple paths, video data may still be on-time even if one or more paths experience congestion. We introduce redundancy in Conflux using a block FEC code. The amount of redundancy is expressed as the *coding rate* which is the ratio of the amount of non-redundant (i.e., original) data to the total

amount of data sent. Because of the probabilistic interface offered by each link’s PSuccess function, Conflux can dynamically determine the coding rate that maximizes expected end-user utility.

We investigate Conflux’s ability to deliver video data by the user’s acceptable delay and the effectiveness of its recovery mechanisms in various network environments. We first evaluate Conflux in network environments that have retransmission mechanisms at the link layer, which is common for LTE [59, 19]. Furthermore, we also evaluate Conflux in network environments where lost packets are not retransmitted by the link layer. Without any underlying retransmission mechanisms, the video stream fully relies on Conflux’s recovery mechanisms to prevent lost data. This allows us to illustrate Conflux’s ability to deliver data by the user’s acceptable delay for other types of network environments such as WiFi. To measure Conflux’s ability to deliver data by the user’s deadline, we use the *percentage of incomplete frames*, where an incomplete frame is a video frame that is missing video data due to delay or loss, as our primary evaluation metric.

This chapter makes the following contributions to the area of recovery mechanisms for video streaming:

- We present extensions to Conflux that enable retransmission and FEC. Furthermore, we introduce new User Preference modules that ensure sufficiently high redundancy level to prevent the video stream from experiencing delayed data. These User Preferences can be tailored to the user’s preferred redundancy level.
- We evaluate the impact that Conflux’s recovery mechanisms have on the video stream in challenging network environments. In a five-link environment that does not drop packets and experiences large drops in available bandwidth, we find that using recovery mechanisms reduces the percentage of incomplete frames from 7% to 2%.
- We show that using recovery mechanisms can significantly reduce the percentage of incomplete frames in environments that do not retransmit packets at the link layer. We find that using Conflux without any recovery mechanisms to stream video results in 91% of incomplete frames in a five-link environment that has 2-5% random packet loss on each link. Using a redundancy level of 10% lowers the percentage of incomplete frames to 0.3%

1.2 Outline

Efficiently using multiple links for live, adaptive video streaming is challenging because a protocol must determine the amount of data to send on each link, the video bitrate, and the appropriate level of redundancy so that video frames can be delivered within the user’s strict latency requirements. In this thesis, we explore solutions to the challenges of multi-homed, adaptive live video streaming. This thesis has three main contributions:

- Firstly, we present the PacketBurst bandwidth measurement technique to improve the accuracy of the packet trains. Accurate bandwidth measurements indicate the maximum sending rate that a link can support. Furthermore, we evaluate various machine learning techniques for predicting link instability. These predictions can be used by adaptive video bitrate protocols to preemptively lower the video bitrate to avoid congestion. Multi-homed transport protocols can use these predictions to move the video stream to an alternate path.
- Secondly, we present Conflux: a multi-homed adaptive video bitrate protocol for live video streaming. Conflux differentiates itself from other multi-homed, live streaming protocols as it introduces a generalized approach to the multi-homed, adaptive live video streaming challenge. Its design enables it to be adaptable to different hardware settings, network types, and user requirements with minimal changes to the core framework.
- Finally, we present and evaluate recovery mechanisms extensions to Conflux. We show that these recovery mechanisms can lower the percentage of incomplete frames in many different network environments.

Chapter 2 presents related work pertaining to bandwidth estimation, adaptive video bitrate protocols, live streaming protocols, and multi-homed transport protocols. Chapter 3 presents PacketBurst and our evaluation of machine learning techniques to predict link instability. Chapter 4 presents the Conflux system. Chapter 5 presents recovery mechanisms such as retransmission and redundancy to Conflux, additional redundancy-based user preferences, and an in-depth evaluation of Conflux in various network environments. Finally, Chapter 6 describes future research directions and summarizes the contributions of this thesis.

Chapter 2

Related Work

This chapter first provides background on different video streaming applications. We then describe research in the area of bandwidth measurement, link quality characterization, and low-latency, adaptive multi-homed video streaming. We find that there are few works that present systems for low-latency, adaptive multi-homed video streaming. Nonetheless, there are many relevant research areas that address a single challenge found in this application.

2.1 Background

Popular video streaming applications can largely be classified as either live or on-demand (e.g., Netflix, YouTube). The focus of this thesis is on live video streaming where video content is captured at the sender and delivered to the receiver. However, we also provide background Dynamic Adaptive Streaming over HTTP (DASH) for on-demand video as most adaptive video bitrate protocols are designed for this application.

2.1.1 Live Video Streaming

Live video streaming involves capturing and streaming video footage of an event in real-time or near real-time. In the case of on-site, live newscasting, which is the focus of this thesis, many commercial offerings provide sub-second latency [50, 89] to allow for interactivity between the sender (i.e., news reporter) and the receiver (i.e., television station or news anchor).

In this application, video footage is encoded at the sender, sent over the network, and decoded at the receiver. Video encoders, such as x264 [109], generate encoded video frames that are sent over the network. Video frames have different priorities, and a video frame's priority is referred to as its type. A video frame's type is determined by the encoder when the encoder generates the frame. Video frame types include Independent (I), Predicted (P), and Bi-directional (B).

Typically, I frames are the most important frames because subsequent frames cannot be decoded without them. P frames require the previous frame to be decoded, and B frames require both the previous and subsequent frame to be decoded. Many works consider the priority or importance of these frames in their protocol design and focus on prioritizing I frames over P and B to reduce video distortion and to improve on-time video data arrival.

2.1.2 Dynamic Adaptive Streaming over HTTP (DASH) Video

In on-demand video streaming, the video content already exists on a server and is delivered to the viewer upon request. On-demand videos are often delivered over HTTP. In HTTP video streaming, multiple versions (i.e., different bitrate encodings) of a video are maintained at a web server. In order to view the video, the client determines its desired bitrate and makes HTTP GET requests to retrieve the video at this bitrate. The client may retrieve the entire video using one GET request, or it may make multiple byte range requests to retrieve an entire video [132]. Videos may also be broken down into multiple chunks (i.e., seconds), where each chunk has its own unique Uniform Resource Identifier (URI). The client retrieves multiple files to play the video.

HTTP video streaming affords many advantages over protocols that directly control the media stream such as RTSP and RTMP. Firstly, unlike RTSP and RTMP, HTTP video streaming does not maintain the video stream's state (e.g., position, network statistics, etc.). Thus, there is often less communication overhead between the client and its video server. For example, a client does not communicate with a server as frequently if it retrieves an entire video through a single GET request. Secondly, video streamed over HTTP can more easily traverse network address translators and firewalls than RTSP and RTMP. Finally, using HTTP for video streaming is simpler and more cost-effective since existing infrastructure already supports webpages as HTTP requests.

Although using HTTP to stream video provides many advantages, progressively downloading portions of video using HTTP may cause wasted bandwidth as the client may not watch the retrieved video (the entire byte range or chunk) to completion. Furthermore, basic HTTP video streaming systems/applications (where a client progressively downloads

a video) requires the entire video to exist on the web server so clients may retrieve it. Thus, without additional developments to using HTTP to deliver video, it cannot be used for live streaming that requires less than a second of delay.

Basic HTTP video streaming was further developed into Dynamic Adaptive Streaming over HTTP (DASH, or other wise known as MPEG-DASH) to address the shortcomings of using HTTP alone to deliver video. In DASH, the client is responsible for monitoring network conditions and for selecting the highest bitrate that can be downloaded without rebuffering or stalling.

In DASH, the content’s Media Presentation Description (MPD) maintains a manifest of the available bitrates, their respective URL addresses, and the video’s available segments. The client first obtains the MPD to determine the availability, media type, available streaming bitrates, locations on the network and other characteristics of the content. With the information in the MPD, the DASH client determines the appropriate bitrate and starts fetching the video or media using HTTP GET requests. MPEG-DASH only defines the MPD and segment formats. Client behaviour such as bitrate adaptation and playback are not defined in the standard. Consequently, there has been research in rate adaptation for HTTP streaming.

2.2 Link Quality Information

Applications found in mobile (e.g., wireless LTE) environments often use links that have highly variable available bandwidth. This variability is due to fluctuations in signal strength caused by physical conditions and the wireless network’s shared infrastructure [11, 8, 87, 14]. Sudden decreases in the available bandwidth may cause an application to send at a rate beyond what its link can support, and this can result in congestion and delayed data. To avoid congestion and delays, many applications adjust their sending rate in accordance to their links’ available bandwidth and quality. This section describes works that measure bandwidth and predict link quality.

2.2.1 Bandwidth Estimation

Adaptive video streaming protocols often use knowledge of their link’s available bandwidth and condition to determine their video encoding rate. Ideally, bandwidth estimates are accurate so that protocols do not over-send and cause congestion or under-send and forgo

higher video quality. Existing bandwidth estimation techniques can be classified as either self-loading periodic streams or packet trains [110].

Self-loading periodic stream bandwidth estimation techniques such as [66, 117, 154] temporarily saturate the available link until the maximum achievable bandwidth is determined by the presence of additional queuing delay or the lack of an acknowledgment. Pathload [66] sends a fleet of streams and employs an idle period between streams to let the path drain. The receiver checks the measured one-way delays for an increasing trend between each stream in a single fleet to determine the available bandwidth on a link. PathChirp [117] induces congestion by sending an exponential flight pattern of probes called chirps. PathChirp calculates the per-packet estimates of available bandwidth by analyzing the queuing delay of a train of chirps. SLoPS [154] sends probe streams at specific rates to induce congestion to measure the available bandwidth. The available bandwidth is determined by the current sending rate and the difference between the sending interval and the receiving interval. If packets are received within a longer time than they are sent, then the current sending rate is greater than the available bandwidth.

Using self-loading periodic streams for bandwidth estimation can temporarily congest the link, delay data, and cause undesired link-layer re-transmission. An alternate approach to bandwidth estimation is to use a packet train. Packet-train bandwidth estimation techniques send a group of packets in short succession of each other, and the bandwidth estimate is calculated using packet inter-arrival time.

Keshav [76] presents packet-pair probing for flow control to estimate available bandwidth on each link. Packet pairs involve sending two packets in short succession of each other. As they pass through the bottleneck link and are serviced by the receiver, the service time between the two packets differs by the bottleneck link's rate. The available bandwidth is measured as the size of the packets divided by the inter-packet service time.

Hu and Steenkiste [57] present the Initial Gap Increasing (IGI) and the Packet Transmission Rate (PTR) methods to estimate available bandwidth. This work presents a single-hop gap model that is used to determine if the packet pair's inter-arrival time, or gap, can be used to accurately estimate available bandwidth in the presence of competing traffic for a single-hop network. IGI and PTR algorithms send a sequence of packet trains, increase the initial gap from the source to the destination, monitor the difference between the average source and destination gap, and use this information to measure the competing traffic's bandwidth. The bandwidth estimate is calculated as the difference between bottleneck link's capacity and the competing traffic's bandwidth.

cprobe [22] is used to measure competing traffic for the purpose of determining a bottleneck link's bandwidth utilization which is then used to determine which server a client

is assigned to. `cprobe` sends a short stream of echo packets to the destination server and records the time between the receipt of the first packet and the receipt of the last packet to measure the presence of competing traffic on the bottleneck link. The available bandwidth is calculated as the total amount of bytes that are sent divided by the time difference between the first and last packet. `cprobe` relies on knowing the bottleneck link’s capacity, which can be measured using `bprobe` [22], to determine the size of the packet train that is used for probing.

Spruce (Spread PaiR Unused Capacity Estimate) [133] measures Cross-Traffic Rate (CTR) of the bottleneck link using a sequence of packet pairs that are spaced out exponentially. The packets within the pair are spaced out such that second probe packet arrives at the bottleneck before the first packet departs, and the CTR is measured using the difference in the inter-packet time between sending and receiving the packet pair. Spruce reports the average CTR over a sequence of packet pairs.

Using packet pairs or short packet trains may not accurately measure the available bandwidth when there is traffic shaping because techniques such as [76] rely on the assumption of fair queuing. Balasingam et al. [14] have found that packet trains need to be sufficiently long to span multiple scheduling cycles to obtain an accurate bandwidth estimate.

TOPP [98] sends trains of packet pairs and estimates available bandwidth over two phases. The first phase consists of injecting equally-sized, well-separated packet pairs into the network at an offered rate until a maximum offered rate is reached. On the receiver side, the packets are timestamped upon reception. The second phase estimates the available bandwidth by dividing the packets’ size by their time separation. Because TOPP uses a sequence of offered rates, it analyzes the sequence of bandwidth estimates to determine the rate that causes congestion, which is the bandwidth of the bottleneck link.

Although packet train techniques do not congest the link, their estimates may be inaccurate due to kernel scheduling [66]. Kernel scheduling can result in multiple packets being serviced in the same interrupt, resulting in an overestimation of the available bandwidth. Using this overestimation in application logic may result in congestion if the application attempts to send at the overestimated rate.

2.2.2 Link Quality Characterization

In network environments that experience sudden changes, bandwidth estimates may not be valid for an extended period of time. Consequently, it is advantageous for applications to predict the available bandwidth and adjust their sending rate accordingly [93]. This is

important for many live streaming applications since the video encoder often requires time to reach the new video bitrate.

Yue et al. [163] found that the most important features that can predict available bandwidth in LTE include: throughput, RSRP, RSRQ, CQI, BLER and handover events. This work also finds that using random forests for prediction results in an error of 3.9 - 17%. Raca et al. [111] evaluates random forests, support vector machines, and Long Short-Term Memory (LSTM) for predicting throughput. To accomplish this, Raca et al. [111] uses a history window which indicates the period of time that is used for calculating features and a horizon window that indicates the period of time for the predicted bandwidth. Summarization quantiles (i.e., 25th, 50th, 75th) of measurements are used as attributes for the machine learning model. These attributes are constructed from throughput, users-per-cell, CQI, RSRP, RSRQ, and SNR measurements. This work evaluates the different machine learning models and finds that LSTM provides the lowest average relative errors when using raw data as input. This work also finds that using longer history and horizon windows reduces relative errors.

Works such as Minovski et al [101] have also found that signal strength (RSSI, RRSP, and SINR) provided the greatest accuracy for predicting throughput. This work evaluated different models and found that decision trees provided the greatest accuracy. Finally, Na et al. [104] predicts throughput using a sliding window of previous measurements with LSTM.

These works do not predict other aspects of link quality, such as latency and loss, which could be beneficial to the video stream. Nonetheless, the ability to predict these aspects is important because an increase in latency can cause video to arrive past its deadline and stall, and the presence of packet loss can cause artifacts in the video stream.

2.3 Multi-homed Transport Protocols

Multi-homed transport protocols use multiple paths to deliver data. Using multiple paths instead of a single path may allow applications to obtain higher available bandwidth through aggregation. Furthermore, multi-homed protocols are more resilient to link or path failures than single-homed protocols as they have alternate paths to maintain the connection between the sender and the receiver when a link or path fails. Finally, a multi-homed transport protocol may also obtain lower latency than a single-homed transport protocol by sending data over the less congested paths that are available. In this section, we describe popular multi-homed transport protocols such as MPTCP, MPQUIC, and MP-DCCP.

2.3.1 MPTCP

One of the most popular multi-homed transport protocols is Multi-path TCP (MPTCP) [112]. MPTCP is designed to provide the end user with an unmodified socket API and the same service guarantees as TCP. An MPTCP connection consists of multiple paths (subflows), and per-subflow sequence numbers are used to detect losses and to determine retransmission. All flows share a single send and receive buffer.

MPTCP’s congestion control algorithm and scheduler determine how much data can be sent on a single subflow and how data is scheduled over all available subflows. Firstly, MPTCP’s congestion control algorithm determines the congestion window size of each subflow. Works in MPTCP congestion control [146] often focus on ensuring fairness in the bottleneck link for the multiple paths, or prioritizing subflows with specific characteristics such as low latency. The Linked Increase Algorithm (LIA) is MPTCP’s default congestion control algorithm. In LIA, the congestion window increase is parameterized by an aggressiveness factor and the total congestion window over all subflows. In LIA, a subflow’s congestion window increase is at most as the congestion window increase if the subflow was a single TCP connection. Kalili et al. [77] present the Opportunistic Linked-Increases Algorithm (OLIA) which increases the window size more quickly on paths that have low Round-Trip Time (RTT) and small window sizes. Finally, Peng et al. [108] present Balanced Link Adaptation (BALIA) that increases each path’s congestion window based on its RTT and decreases the congestion window if any packet loss is experienced.

MPTCP schedulers often use the subflows’ congestion window size and other network metrics, such as RTT, to determine which subflow to use and how much data to send on each subflow. The default MPTCP scheduler is Lowest RTT. This scheduler selects the subflow with the lowest RTT and space in its congestion window to send data. Many works have found that MPTCP’s default scheduler is not suitable for live video streaming, especially in the presence of link heterogeneity.

To address link heterogeneity, many MPTCP schedulers have been developed for low-latency workloads. Schedulers such as DAPS [80], OTIAS [160], and BLEST [45] use RTT information for scheduling with the goal of minimizing head-of-line blocking. ECF [83] and STFF [62] use subflow quality information, such as its speed and state, to make scheduling decisions that minimize segment transfer time. DPSAF [159] identifies and sends segments on under-utilized subflows.

In addition to low-latency schedulers, works such as [39, 20] introduce the notion of partial reliability to MPTCP for delay-sensitive workloads such as live video streaming. As data has a deadline in live video streaming, it is unnecessary for MPTCP to resend

delayed data as it is no longer needed. MPTCP-SD [39] is built for multimedia media applications and discards lower priority data when links are congested. MPTCP-SD [39] sends negative acknowledgments for packets that are delayed and returns data out-of-order to enable data to meet its deadline. PR-MPTCP+ [20] uses the advertised receive window size to determine if the receiver will experience buffer blocking. In the case where buffer blocking occurs, PR-MPTCP+ enables partial reliability and drops low-priority data.

Furthermore, there are works that extend MPTCP for multimedia streaming. ADMIT [150] extends MPTCP to support multimedia transactions by integrating Forward Error Correction (FEC) to replace packet retransmission and selects subflows that exhibit high reliability (i.e., few throughput fluctuations). DEAM [149] aims to minimize energy consumption of multimedia streaming over MPTCP. It uses a mathematical model of the wireless networks, video distortion, and energy consumption to determine the subflow allocation.

Using MPTCP or other protocols that offer in-order and reliable delivery guarantees across multiple links can result in more deadline violations than using a single link. This is because MPTCP is susceptible to HoL blocking when any of its links experiences loss or congestion. Furthermore, these works do not address video bitrate adaptation. Without video bitrate adaptation, the video encoder may provide too much data resulting in congestion and increased delays as the aggregate available bandwidth on all subflows may still be insufficient to deliver the video data. Alternatively, MPTCP may forgo higher video quality from using a higher video bitrate by not saturating the available paths.

2.3.2 MPQUIC

QUIC is an emerging transport layer protocol built on top of UDP to enable low-latency data transfer. QUIC (Quick UDP Internet Connection) combines HTTP/2, TLS, and TCP over UDP to reduce latency of client-server communication. MPQUIC [142] is multipath-enabled QUIC which is designed to leverage multiple network interfaces. The advantages of using MPQUIC over MPTCP include shorter subflow establishment (half a RTT) and finer grained scheduling which allows for stream awareness.

There have also been works that investigate the use of MPQUIC for on-demand video streaming using DASH [161, 152]. AMSQ [161] introduces an adaptive stream scheduler for scalable video coding in multipath QUIC. It matches different layers to different subpaths to prevent frequent switching of video quality. AMSQ schedules segments with earlier deadlines on higher priority streams, and each high-priority stream is assigned to one path exclusively. Low-priority streams (enhancement layer) are not transmitted until

high-priority streams complete. MQ-FGTM [152] performs finer-grained scheduling for groups of video frames and assigns higher priority frames to higher priority paths instead of scheduling large video segments. XLINK [167] delivers short videos using concurrent streams and re-injects delayed video frames that would otherwise cause HoL blocking. 4D-MAP [128] present a linear upper confidence bound based online learning algorithm for making packet scheduling decisions using round-trip time, congestion window size, and packet loss information.

2.3.3 MP-DCCP

The Multi-path Datagram Congestion Control Protocol (MP-DCCP) provides multi-path transport for latency-sensitive applications that do not require reliable delivery [36]. MP-DCCP implements the Datagram Congestion Control Protocol (DCCP) for multiple paths and allows users to select how data is scheduled over the available paths. These schedulers include:

- **Default:** Schedules data on the first available path
- **SRTT:** Schedules data on the path that has the shortest round trip time
- **RR:** Round-robin scheduling over all available paths
- **Redundant:** Replicates data over all available paths
- **Out-of-Order Transmission for In-Order Arrival:** Schedules data such that video data is expected to arrive in order at the receiver
- **CFP:** Paths have a predefined priority. If a prioritized path is congested, the next available path is selected for transmission

MP-DCCP may not be suitable for streaming live video in a multi-homed environment as each of its schedulers has short-comings. Firstly, the Default, RR, and CFP schedulers do not consider link heterogeneity and may schedule data on high-latency links, resulting in delayed video data. Furthermore, the Redundant scheduler may make inefficient use of the available bandwidth which results in low video quality. Although the SRTT and Out-of-Order schedulers may reduce delays, the bandwidth of the available links is not considered. Therefore, if the selected bitrate is greater than the total aggregate available bandwidth, MP-DCCP is still susceptible to congestion and increased delays.

2.4 Multimedia Transport Protocols

Many video streaming applications utilize multimedia transport protocols to deliver data. These protocols may be single-homed (e.g., RTP, RTCP) or multi-homed (e.g., MRTP, SCTP). Some multimedia transport protocols are content-aware and use the priority of the video data (or frame) to make scheduling decisions.

2.4.1 Single-homed Multimedia Protocols

The Real-time Transport Protocol (RTP) [121] provides a standardized packet structure for audio and video data fields, enabling different multimedia applications to work with each other. The Real-time Transport Control Protocol (RTCP) [61] is used in conjunction with RTP. RTCP constructs and delivers sender and receiver status reports which include information such as packets sent, packets lost, and jitter. RTP and RTCP are primarily responsible for connection establishment and management. They are not responsible for video bitrate adaptation and congestion control. These features are typically implemented by other protocols that build on top of RTP and RTCP. We discuss these protocols in Section 2.5.3.

2.4.2 Multi-homed Multimedia Protocols

Multi-homed transport protocols for multimedia include MRTP [95] and MPRT [127], which are extensions of RTP. A MRTP [95] session consists of multiple flows, and flows that experience excessive loss or congestion are deleted from the session. Multimedia packets are scheduled on the available flows in a round-robin fashion. The MRTP receiver uses a de jitter buffer to reorder and return the multimedia packets. MPRT schedules RTP traffic across multiple paths at the sender and uses a de jittering algorithm at the receiver side. The MPRT scheduler calculates the receiver rate and characterizes the paths as congested, lossy, or non-congested, and it schedules important data (e.g. I frames) on non-congested paths. On the receiver side, the MPRT player ensures that the buffer does not empty and that the playout stream does not stall by adjusting the playout speed.

Multi-homing is also available in SCTP [131], which is designed for data streams. In SCTP, a primary path is established and alternate paths are used for retransmission or backup. If the primary path fails, SCTP switches to the the backup paths. This scheme does not fully utilize the total aggregate available bandwidth. To address this shortcoming of SCTP, CMT-SCTP [65] was proposed. CMT transfers new data from a source to a

destination host via two or more end-to-end paths. Works such as LS-SCTP [7], WestwoodSCTP [23], CMT-QA [156], CMT-DA [147], and CMT-CA [151] focus on improving CMT-SCTP for real-time video streaming.

Load-Sharing SCTP (LS-SCTP) [7] is designed for networks with limited bandwidth, high-loss, and failure prone links. LS-SCTP separates flow control from congestion control. All paths at the sender and receiver share the same buffer for flow control, and congestion control is performed for each path. Load sharing occurs among the available paths where data is scheduled on paths with the highest bandwidth delay product.

WestwoodSCTP [23] load balances across multiple SCTP connections that have disjoint paths. Each path is characterized by its one-way latency and its available bandwidth. The available bandwidth is estimated using the rate of returning acknowledgment messages. Data is sent as a packet train of size two, and it is scheduled on the path that is expected to be the fastest for delivering the data.

CMT-QA [156] presents a path quality estimation model, a data distribution scheduler, and an optimal retransmission policy. The sender maintains data in a buffer until it is acknowledged, and the buffer's draining rate is used to estimate path quality. The data distribution scheduler uses the path quality estimation model to determine the paths that are suitable for load sharing and assigns the appropriate data flows. CMT-QA's optimal retransmission policy distinguishes between random loss and congestion to determine the minimum amount of data that it needs to retransmit, and it selects the path with the minimum transfer delay for retransmission.

CMT-DA [147] presents distortion-aware concurrent multi-path transfer which includes per-path status estimation and congestion control, flow rate allocation that optimizes for video quality, and delay and loss controlled data allocation. Path quality is estimated using congestion window size and round-trip time. To minimize end-to-end video distortion, CMT-DA sends data on paths with higher quality.

CMT-CA [151] considers the content of the video that is being sent. CMT-CA performs frame-level scheduling based on estimated video parameters and link quality. Using a link's round trip time, available bandwidth, and packet loss rate, CMT-CA models each link's ability to deliver data and uses a Markov-Decision Process for congestion control. Using the link's quality information, CMT-CA determines the links' aggregate sending rate, differentiates between I and P frames, and preemptively drops less important (P) frames to avoid congesting the network.

Although these works provide multi-homed transport, they lack video bitrate adaptation. As a result, users may be forgoing video quality if the total aggregate available bandwidth is greater than their selected video bitrate. Conversely, using these systems

may also result in delayed data if the total aggregate available bandwidth is lower than the user’s video bitrate. This motivates the need for video bitrate adaptation in conjunction with multi-homed transport.

2.5 Adaptive Bitrate Protocols

Most adaptive bitrate protocols are designed for on-demand Dynamic Adaptive Streaming over HTTP (DASH). Although the properties of on-demand video streaming are different from live and real-time video streaming, strategies from DASH can be applied when designing adaptive video bitrate protocols for live streaming. Non-DASH solutions for live, video bitrate adaptation are often found as congestion control protocols that set the video bitrate as a function of the congestion window. In this section, we survey adaptive video bitrate protocols for single and multi-homed DASH and for single-homed live streaming.

2.5.1 Single-homed Adaptive Bitrate Protocols for Dynamic Adaptive Streaming over HTTP (DASH) Video

In this section, we survey various adaptive bitrate protocols that aim to improve end-user QoE. Adaptive bitrate protocols typically use network measurements or buffer occupancy as feedback to determine a video’s bitrate. These protocols can be found on the client-side, in-network, or server-side. Client-side adaptive bitrate protocols use information such as buffer occupancy, latency, and other feedback from the network and/or video server to determine the appropriate video bitrate. Facilities within the network can determine how to allocate bandwidth to the clients and how to schedule video packets. In-network systems that have a global view improve end-user QoE by assigning clients to CDNs that are expected to provide the client with the highest QoE. Server-side support can also decide how to schedule packets such that the network is not overwhelmed. In this section, we survey various adaptive bitrate protocols and systems that aim to provide end users with high QoE.

Feedback-Based Algorithms

Yin et al. [162] uses Model Predictive Control (MPC) for dynamic adaptive video streaming over HTTP. This work formulates video streaming as a QoE maximization problem that is subject to constraints on the available bandwidth, buffer size, and playback. The

basic MPC algorithm involves predicting the throughput for the next N chunks (where the throughput predictor can be specified). MPC also determines the optimal bitrate given the current buffer occupancy, the previous bitrate, and the predicted throughput prediction (which is the solution to the QoE maximization problem). Finally, MPC downloads the subsequent chunks at the optimal bitrate. The effectiveness of MPC depends on the accuracy of the throughput prediction.

Liu et al. [86] present a client-side bitrate adaptation method for HTTP streaming. As HTTP video is typically streamed in segments, clients measure the segment fetch time, which is the period of time from issuing a GET request to receiving the segment. This work observes that the segment fetch time should be the same as the segment’s playback time; otherwise, any deviation would indicate that the available throughput is either too low or too high for the client’s requested bitrate. The presented work calculates the ratio between the segment’s duration and the segment fetch time, and this is used to calculate the TCP throughput by multiplying it by the video’s bitrate. The measured throughput is then smoothed before determining the appropriate bitrate. The presented rate adaptation algorithm increases the bitrate if the ratio of the media segment duration to the segment fetch time is greater than one plus a switch factor. The algorithm lowers the bitrate if the calculated ratio is below a switch down threshold.

Many feedback-based adaptive bitrate protocols depend on either accurate bandwidth estimation or other metrics such as latency or fetch time. Adaptive bitrate protocols that use bandwidth estimation to determine a bitrate require accurate measurements [162]. In a noisy, wireless environment, one needs to be able to measure bandwidth often and without saturating the network. In addition to collecting feedback, an adaptive bitrate protocol must be able to react quickly and appropriately. Significant reactions to spurious network congestion may result in an end user receiving a far lower bitrate than is necessary.

Buffer-Based

Bitrate adaptation algorithms may also use buffer occupancy or the changes to buffer occupancy to determine the video’s bitrate. Players may also use the playback buffer as a way to smooth out video in order to improve end-user QoE.

Huang et al. [60] use buffer occupancy to directly choose the video rate with the goal of avoiding unnecessary rebuffering and maximizing the average video rate. This work finds that a pure, buffer-based approach works well when there is sufficient past observations (e.g., network measurements). However, when the buffer is initially growing (i.e., video is in the start-up phase), there is insufficient information to determine the appropriate bit

rate. This work presents a class of algorithms that uses a rate map to directly select the video rate based on the buffer occupancy. The presented work designates a reservoir regions within the buffer that is used to absorb variation in network capacity. While this reservoir is filling up, the video is requested at the minimum bitrate. As the buffer approaches capacity, the algorithm selects the maximum available bitrate.

Mok et al. [103] propose a QoE-aware system named QDASH that assists clients in selecting the most suitable video quality level. This work also finds that inserting intermediate bitrate levels between current available bitrates provides better QoE because it enables changes to the video’s bitrate to be less noticeable by the end user. QDASH is composed of two components: QDASH-abw and QDASH-qoe. QDASH-abw resides on the video server and is responsible for inspecting video data flows and shaping the sending rate. QDASH-abw is also responsible for measuring bandwidth using a proxy that reshapes TCP packets into packet trains. QDASH-qoe on the client side is responsible for bitrate adaptation and uses information from QDASH-abw. When there is less available bandwidth for video streaming, QDASH uses buffered video to continue playback while downloading video at an intermediate level. This enables QDASH to smooth out the picture quality change, thereby improving end-user QoE.

Tian et al. [138] demonstrate that client-side buffered video time is a good feedback signal for video adaptation and present a system that balances video rate smoothness and bandwidth utilization. This work uses the changes in buffered video time to adjust the bitrate by setting a target video buffer time to build a PID controller to regulate the requested video rate.

Jiang et al. [71] presents FESTIVE, which is an HTTP-based adaptive video streaming algorithm that trades off fairness, efficiency, and stability. The core FESTIVE algorithm focuses on steady-state behaviour. It estimates the bandwidth using the last k (20) throughput estimates. This estimate is then used to compute a reference bitrate, which is used to calculate an efficiency and stability score. The algorithm then determines how it wants to trade off efficiency and stability through a delayed update. Finally, FESTIVE uses a randomized scheduler and chooses a random target buffer size for the playback buffer in order to avoid biases that are induced by initial conditions (i.e., periodic scheduling of video chunks).

Buffer-based adaptive bitrate protocols depend on the availability of video to buffer (e.g., on-demand content delivery such as Netflix and YouTube). In some scenarios, such as live streaming or video conferencing, a sufficiently large buffer to absorb changes in the network may not be available; thus, these strategies are unusable.

Machine Learning and Classification Approaches

There have been many works that use various machine learning and artificial intelligence approaches to determine the appropriate video bitrate, to control video streams, and to select CDN servers used to stream video. Some works focus on selecting attributes (e.g., critical features) to infer the performance of a network link in order to improve end-user QoE, and other works formulate the QoE problem as an exploration-exploitation based problem. In this subsection, we describe works that use machine learning for throughput prediction, which is used to select the video’s bitrate. We also outline works that use video session features as means to select bitrates and CDN servers.

Critical Feature Selection and Similarity

Ganjam et al. [47] describes a split control plane architecture for optimizing Internet video delivery. In this work, aggregation is used as a strategy to scale out the ability to make control decisions at the global level. This work employs a coarse-grained global model layer that uses a global view of client quality measurements to build a data-driven prediction global model of video quality. A fine-grained per-client decision layer is responsible for making decisions (i.e., CDN selection and bitrate) for clients. The intuition behind this approach is that clients can tolerate staleness in the network model and sessions with similar features (i.e., CDN, ISP, content provider time of day, etc.) have similar quality. This work uses a nearest-neighbour like prediction model to select CDNs and bitrates.

CS2P [134] aims to improve initial video bitrate selection in order to lower startup delay time and to improve QoE during video playback. The insight behind this work is that video sessions with similar features tend to exhibit similar initial throughput conditions and throughput evolution (how the available throughput changes throughout the video session). The system in [134] groups similar sessions that share the same set of critical features values to build prediction models. It learns a Hidden Markov Model for each cluster of similar sessions to predict how the throughput evolves over the duration of the video stream.

Jiang et al. [70] present Critical Feature Analytics, which learns critical features for different video sessions and uses these critical features to predict video quality in order to decide the CDN and the bitrate of the client. Using the intuition from [134], this system learns the video streams’ critical features (i.e., features that affect video quality such as CDN, ISP, content). CFA uses partitions of data based on the most recently learned critical features to create a table that maps a finest partition (which is the distinct values of all features) to a quality estimate. To estimate the video quality in real time, the client looks up its critical features to determine an estimate. Critical feature learning runs offline every tens of minutes. Quality estimation runs every tens of seconds.

VIA [69] aims to improve end-user QoE for Internet telephony, which has the additional requirement of low latency as calls are latency sensitive. VIA uses relay nodes placed at globally distributed datacenters to optimize network performance and call quality. A call using the VIA system can take the default path or a relayed path, and the path the call takes is determined by a centralized controller. VIA uses a prediction-guided exploration approach to determine which path to use. To do so, it gathers performance information from call history and uses network tomography to expand the coverage of the collected information. It then uses this information to predict performance and determines the top-k relaying options. Afterwards, it performs exploration-exploitation on the top-k relaying options.

Pytheas [72] is another system that uses exploration-exploitation to improve QoE. Pytheas is a framework that models QoE as a multi-armed bandit problem where measurement collection (exploration) and decision making (exploitation) are integrated together. Pytheas aims to update decision in real time with fresh data and to capture the interactions between session features and QoE. This work introduces the notion of group-based exploration-exploitation (from the observation that video sessions with similar features have similar performance); thus, exploration-exploitation only needs to be performed for groups of sessions instead of all individual session. Frontend clusters map a session to a group based on its features, and each session group is managed by one per-group exploration-exploitation process that is used to return a decision (CDN/bitrate) to the session. A session is also responsible for reporting QoE metrics to its front-end cluster, which is then used to update the exploration-exploitation logic of the group. The backend is responsible for determining the session groups.

Markov-Decision Processes

Some works model the bitrate adaptation problem as a Markov-Decision process that is used to optimize end-user QoE.

A Markov Decision Process is:

- A set of states
- A set of actions
- A reward function that is determined by the state and action
- A description of how each action affects the state and the specification of a probability distribution over the next states

Markov Decision Processes follow the Markov property, which is the property that the effects of an action taken in a state depend only on the current state and do not depend on the history.

Zhou et al. [168] formulate the bitrate adaptation problem as a Markov Decision Process. A decision on the bitrate is made for every segment of the video. The state vector used is the buffered video time, average changing rate of buffered video time, video rate vector (of the video rates assigned to fragments), average bandwidth during downloading of fragment k (which is also used as a bandwidth estimation), and a 0/1 indicator that represents if the latest N fragments have the same video rate. Rewards are associated with an action, and the goal is to find an optimal strategy that maximizes the reward. This work uses a time-varying Markov model that is used to estimate future bandwidth and sets the reward function based on user perception (which is dependent on rate changes and playback freeze). Rewards for rate switching are dependent on buffer overflow, buffer underflow, and smoothness of rate switching. This work uses a greedy algorithm that selects video rates for individual fragments instead of determining video rates for all fragments concurrently. Furthermore, it finds there is a trade-off between visual quality and computational complexity.

Xiang et al. [153] focus on modelling the rate adaptation problem as a Markov Decision Process for DASH streaming on wireless networks. The clients use a Markov Decision Process for each video segment. The client decides the video version / resolution of the next segment or if it should be idle to avoid buffer overflow. Actions include accessing a higher layer, accessing a lower layer, or waiting. A state is defined by the following: number of buffered frames, queue length variation after a new segment has been retrieved, version index of the last received segment, difference of video versions requested in consecutive steps, available bandwidth, and the number of received segments. Using available bandwidth for the current segment, the work estimates a probability distribution of the bandwidth for the next segment using the state transition probability matrix of the Markov model. The rate adaptation problem is formulated as an optimization problem, which is solved using dynamic programming. The Markov Decision Process model trades off between average video quality and playback smoothness using a parameter in the reward function. The result is a table that maps a state to an action, which is then used for online decision making. Designing an online algorithm is left as future work. Other future work include how to organize layer segments, and efficient segment size research.

Reinforcement-Based Learning Techniques

Mao et al. [94] present Pensieve, which is a system that learns adaptive bitrate algorithms automatically using reinforcement learning. State observations such as the current

buffer occupancy, rebuffering time, chunk download time, size of the next chunk, and the number of remaining chunks in the video are passed in as input to the reinforcement learning agent to determine a policy. A policy is defined as a probability distribution over actions, where an action corresponds to the bitrate for the next video chunk. Pensieve uses a neural network to represent a policy that is adjustable using a number of parameters. Although the work shows that Pensieve outperforms many existing adaptive bitrate algorithms, it is not suitable for live video as some inputs (such as the number of remaining chunks) may not be available to determine a policy.

Sengupta et al. [125] present HotDash Adaptive Bitrate Algorithm that uses an actor-critic reinforcement learning algorithm to determine which video segments to pre-fetch and the video bitrate to use. HotDash prioritizes video segments that have high user interest (e.g., featuring an event or person).

Video Bitrate Prediction Techniques

Recent works such as [17, 120, 171] predict the video bitrate that provides the highest user Quality of Experience (QoE).

Zou et al. [171] demonstrate that using perfect bandwidth predictions alone is not sufficient to provide users with high QoE since it results in high bitrate switching. This work augments existing adaptive video bitrate protocols with perfect available bandwidth predictions and finds that the predictions can be used by these protocols to achieve 96% of optimal's video bitrate.

Sani et al. [120] presents an adaptive video bitrate protocol (SMASH) that is trained on the decisions made by nine other adaptive video bitrate protocols which are buffer-based, rate-based and hybrid protocols. This work trains eight machine learning models and finds that a random forest classifier provides the highest accuracy at determining the decisions of other adaptive video bitrate protocols. This work shows SMASH achieves higher video bitrate and fewer bitrate switches than Pensieve.

Bentaleb et al. [17] predict available bandwidth using historical bandwidth measurements and a recursive least squares algorithm. This bandwidth prediction is used to determine the optimal bitrate that minimizes the estimated error of a video chunk's arrival time and maximizes the user's QoE which is expressed as an objective function.

2.5.2 Multi-homed Adaptive Bitrate Protocols for DASH Video

In the scenario where a single link or path does not have sufficient bandwidth to stream video, the video stream may be partitioned and streamed on multiple paths. However,

blindly using multiple paths can often lead to worse video quality than if a single path had been used. James et al. [67] investigates the use of Multipath TCP to stream HTTP based video. This work emulates WiFi and cellular LTE links and uses these connections to send video from a client to a server. The presented results demonstrate that if the bandwidth over the aggregate links is borderline to supporting a bitrate, using MPTCP could be detrimental as the video stream would undergo significantly more bitrate switches than if the primary link was stable. If the secondary link is stable, then it can improve average segment quality. Therefore, this work suggests that a multi-homed adaptive bitrate algorithm must determine the stability of the connections it uses in order to determine if there is any benefit to using the available multiple paths (or connections).

Xing et al. [155] investigates the use of multiple wireless links such as WiFi, Bluetooth and cellular to stream video. This work formulates multi-link video streaming as a reinforcement learning task. The current state of the video stream, which is comprised of unplayed queued segments, SVC video layer index, total traffic that Bluetooth used to download the last segment, and the current bandwidth of the WiFi and Bluetooth link, is used in a Markov Decision Process. A rate adaptation agent receives the state and decides to either upgrade, downgrade or maintain the current quality for the next segment. Furthermore, the agent may also decide to request a higher enhancement layer for segments that have yet to be played back. Both Bluetooth and WiFi is used to download the segment simultaneously. The presented system uses two discrete-time finite-state Markov models to describe the available bandwidth for WiFi and Bluetooth. The system measures how good the actions are based on a reward value that is related to the provided QoS. The reward function is designed to consider video QoE requirements such as rebuffering rate, playback quality and smoothness, and service costs. However, requesting video on both Bluetooth and WiFi is relatively expensive in terms of energy consumption and bandwidth cost, as the marginal increase in cost can be significant if the video could be obtained solely from WiFi.

Han et al. [53] addresses some of the shortcomings of using Multipath TCP, particularly the lack of support for prioritizing paths. Without prioritizing links, video may be streamed over expensive links (i.e., metered cellular link). This work focuses on network interface awareness to reduce cellular usage and energy consumed by using a cellular link. This work also finds that in cases where WiFi does not provide stable throughput for streaming video, using WiFi along with LTE often can. This paper presents MP-DASH which is comprised of a scheduler and video adapter. The scheduler is responsible for determining the best fetching strategy of the video chunks and serves as an overlay for MPTCP. The video adapter is responsible for determining the chunk size and the deadline and for allowing DASH bitrate adaptation algorithms to be multipath (i.e., by overriding

bandwidth estimations to include all paths).

Zhao et al. [166] use the observation that bandwidth heterogeneity often results in poor video quality for DASH video streams and present Path Use Decision where MPTCP paths are enabled if their estimated available bandwidth is within a threshold of the maximum estimated available bandwidth of all paths. They experimentally determined that this threshold is 60% of the maximum estimated available bandwidth. Bandwidth estimates are calculated using a moving average of the path’s throughput.

2.5.3 Single-homed, Live Adaptive Video Bitrate Protocols

Non-DASH, adaptive video bitrate protocols are also used for video conferencing. Popular video conferencing applications include Skype, Google+ Hangouts, Zoom, and Webex. These video conferencing systems are proprietary; and as a result, we are not privy to their adaptive video bitrate protocol. However, measurement studies such as De Cicco et al [32], MacMillian et al. [90], Chang et al. [26] investigate their performance.

De Cicco et al. [32] find that Skype is adaptive between 40kbps and 450kbps of available bandwidth, and adjusts the frame rate, packet size, and the video resolution to adaptive to changing network bandwidth conditions. However, Skype refrains from fully utilizing all available bandwidth beyond 450kbps.

MacMillian et al. [90] find that even when there is a lot of available bandwidth (10Mbps), the average network utilization for video conferencing applications such as Zoom, Google Meet, and Microsoft teams range from 0.8 to 1.9 Mbps. These applications also suffer from long recovery times (at least 20 seconds) if the uplink drops to 0.25 Mbps.

Chang et al. [26] investigate Skype, Google Meet, and Webex. This work finds that all systems experience significant quality of experience degradation for high motion video. In particular, with Webex, the average lag grows by 200-300% for complex video content. This work also finds that Zoom and Webex react to rate limiting by sending more traffic in response to sudden frame losses, which is not productive in reducing congestion, especially when streaming over networks such as LTE where users would inflict congestion on themselves.

Xu et al. [158] evaluate iChat, Google+ and Skype. Skype uses a high level of forward error correction for protecting the uploaded video (the coding rate is between 0.47 - 0.55, meaning that half the packets are redundant). Google+ applies selective retransmission where lower video layers are retransmitted first and that retransmission occurs aggressively. Furthermore, iChat always retransmits lost packets. For all of these systems, as packets

are lost, it significantly increases the one-way delay of the video, which may negatively impact the end user’s quality of experience.

These measurement studies have shown that existing video conferencing applications can improve their bandwidth usage to obtain higher video bitrates. Furthermore, they have demonstrated that video bitrate adaptation, protection and distribution have to be considered together to attain high quality video streaming for telephony and conferencing [158].

Congestion control protocols for multimedia transport can also be used for low latency video bitrate adaptation. These include GCC [21], SQP [114], Scream [74], Nada [169], and Pudica [144]. These protocols use congestion signals that depend on the presence of queuing delay, loss, and the receiver rate [169, 74, 21, 114] to determine the presence of congestion which is used to calculate the sending rate/video bitrate. Pudica [144] uses non-payload probe packets to measure the bandwidth utilization ratio which is used for congestion control and rate adaptation.

2.6 Latency-Sensitive, Multi-Homed, Adaptive Video Streaming

There are few works that offer video bitrate adaptation for low-latency video over multi-homed environments. FRA-JSCC is designed for real-time streaming in heterogeneous, error-prone networks that experience packet loss. This work estimates the forward error correction redundancy, the source rate, and the rate allocation to minimize end-to-end video distortion. It determines the expected amount of delayed data due to loss and congestion and adjusts the level of redundancy to ensure that all video data arrives on time. Additional links are included only when they do not increase the expected distortion. To measure distortion, this work relies on encoding a video three times in order to determine specific constants for a video codec and video sequence.

Converge [38] presents multipath extensions to WebRTC, which provides real-time video streaming for the web and for mobile devices, for the video conferencing application. This work finds that relying on common schedulers that are found in MPTCP and MPRTCP cannot provide sufficiently low latency for the video streaming application and that multipath scheduling also needs to be video-aware or video structure aware. In this work, a path’s quality is determined using RTCP statistics, and packets with highest priority (e.g., retransmitted packets and I frames) are sent on paths with the highest quality where as the lower priority frames (e.g., P and B frames) are scheduled on the remaining paths.

This work also presents a path-specific FEC mechanism to determine the amount of FEC to use for each path, and this depends on the path's loss rate and the estimated available sending rate. As this work focuses on video conferencing, its goal is to achieve a 10Mbps video bitrate. If the available links have greater aggregate bandwidth, Converge does not attempt to fully utilize links to obtain a higher video bitrate.

There are few works that address latency-sensitive, adaptive video streaming in the multi-homed environment, thereby leaving this research area largely unexplored. Nonetheless, this research area is important because of its application in on-site newscasting which many rely on it to be informed and make decisions. Many alternatives, such as sending at lower rates and varying the amount of redundancy, have yet to be explored. Furthermore, no works have characterized the trade-offs between these decisions. This leads us to search for a model that enables us to reason about these trade-offs to select the video bitrate and scheduling and recovery strategy that is best suited for the end user.

Chapter 3

Link Quality Modelling

Consistently obtaining high quality video is challenging for applications that use wireless LTE links to stream video data because these links are often highly variable in quality and available bandwidth. The variability in the available bandwidth of wireless links is due to fluctuations in signal strength caused by physical conditions and the wireless network's shared infrastructure. Sudden decreases in the available bandwidth may cause an application to stream video at a rate beyond what its link can support, which can result in congestion and packet loss. This may cause the invocation of automatic repeat requests at the link layer which introduces additional latency and further reduces the application's available bandwidth. As a consequence, video data may be delayed or lost, and this results in poor video quality, which is detrimental to the performance of the application.

In this chapter, we introduce two techniques that characterize link quality; these techniques can be used to improve video bitrate adaptation. We first introduce the PacketBurst technique to improve the accuracy of packet train bandwidth measurements. Unlike self-loading periodic stream techniques such as iperf, a packet train does not saturate the link and induce congestion to measure bandwidth. Instead, it uses the inter-arrival time of packets sent in short succession. However, packet trains are susceptible to having multiple packets being serviced by the same interrupt due to kernel scheduling, and this results in significant over estimations. Removing this inaccuracy is important because accurate bandwidth measurements indicate the bitrate that fully utilizes the link and does not cause congestion.

Bandwidth measurements alone may not be enough for a video streaming protocol to effectively adapt to changing network conditions. This is because existing bandwidth measurement techniques do not capture the variance and the temporal nature of link's

available bandwidth. Therefore, we also introduce our framework that predicts *link instability*, which we define as sharp increases in latency and application-level packet loss, or sudden drops in available bandwidth over a short period of time. We evaluate various machine learning techniques to predict link instability which include decision trees, neural networks, and support vector machines. These link instability predictions can be used by video streaming application to preemptively lower the video bitrate and avoid congestion, and this is useful because the live video encoder experiences a delay between raw image capture and video frame generation at the target video bitrate.

Section 3.1 of this chapter describes the PacketBurst’s contribution and its design, the impact of kernel interrupts on bandwidth estimates, and an evaluation of PacketBurst’s ability to obtain accurate bandwidth measurements. Section 3.2 describes and analyzes our industry-collected network traces, our framework for creating machine learning examples, and an evaluation of statistical machine learning techniques to predict link instability.

3.1 Packet Burst Bandwidth Estimation

Packet-train bandwidth estimates use packet arrival timestamps collected by the receiver to measure the available bandwidth of the link. Obtaining an accurate packet-train bandwidth estimate relies on the accuracy of the receive timestamps. One option is to use hardware or Network Interface Card (NIC) timestamps; however, these are not universally available on all devices, may require specialized driver support [75], and may also not be available in certain deployments such as those using virtual machines.

Alternatively, kernel timestamps are readily available and are preferable to user-space timestamps because they do not reflect inaccuracies caused by CPU scheduling. Unfortunately, kernel timestamps do not account for other handling delays such as the time between packet arrival and servicing. Therefore, due to the scheduling delay between packet arrival and kernel service, multiple packets may arrive before the first packet’s interrupt is serviced.

When multiple packets are serviced by a single interrupt, they are assigned the same timestamp or timestamps that differ by a few microseconds. Using this packet inter-arrival time for calculating the available bandwidth results in significant overestimation. We refer to this phenomenon as *packet coalescing*. The discrepancies between packet arrival and servicing are present in many bandwidth estimation techniques [66].

3.1.1 Contributions

In this section, we present PacketBurst: a bandwidth measurement technique that accounts for kernel interrupts. The PacketBurst technique is able to utilize packet-train measurements that exhibit timestamp inaccuracies that are caused by packet coalescing. To eliminate the impact of inaccurate timestamps on bandwidth estimates, we remove coalesced packets that are at the beginning and at the end of the packet train. This allows an application to utilize packet trains that may otherwise be discarded.

This section makes the following contributions:

- We first demonstrate the inaccuracies of packet-train bandwidth estimates or estimates that rely on packet inter-arrival time. In our measurements, we notice a systematic overestimation of the available bandwidth, particularly in short packet trains.
- We present our PacketBurst technique for packet-train based bandwidth estimation. The PacketBurst technique improves packet train bandwidth estimates by factoring in kernel scheduling.
- We evaluate our PacketBurst technique in a Mininet [100] emulation and in a real testbed with network emulation. We use Linux’s Traffic Control (*tc* [116]) utility to control the condition of the link between the two machines. We demonstrate that PacketBurst bandwidth estimations are more accurate than packet train estimations.

Having accurate bandwidth estimates benefits all streaming applications as they indicate the maximum streaming rate an application can use. This leads to an increase in network utilization and reduces the risk of congestion.

3.1.2 Background

Bandwidth estimation techniques include self-loading periodic streams that temporarily saturate the link’s available bandwidth and those that send a train of packets to estimate the available bandwidth using the packet inter-arrival time [110]. Self-loading periodic stream techniques, such as iperf [52], saturate the link until an increase in latency or the lack of acknowledgement is recorded.

Packet pairs and packet trains are alternative methods for bandwidth estimation, and these techniques are not designed to saturate the link [117, 41, 133, 22]. In these methods,

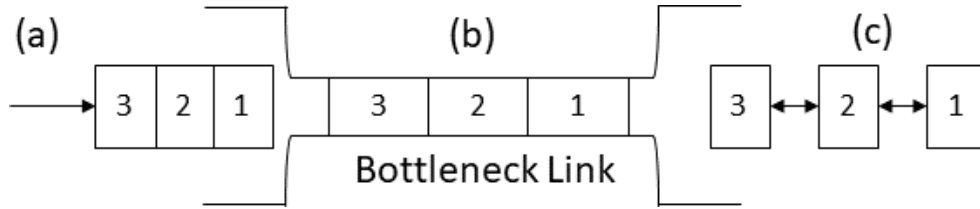


Figure 3.1: Three packets (a) are sent on short succession and must be queued before traversing the bottleneck link (b). Packets are spaced out as they arrive at the receiver (c).

packets are sent in short succession after one another. The bottleneck link spaces out the packets in accordance to its available bandwidth, which is shown in Figure 3.1. In this figure, three packets are sent in short succession and must be queued before traversing the bottleneck link. These packets incur transmission delay in accordance to the bottleneck link’s available bandwidth, leading to a difference in their received timestamps. The available bandwidth measurement of a packet pair, which is a packet train of length two, is calculated as $b = \frac{s}{t_{R2} - t_{R1}}$ where s is the size of the packet, t_{R2} is the received timestamp of the second packet, and t_{R1} is the received timestamp of the first packet. This formula generalizes to packet trains of length n as $b = \frac{s \times (n-1)}{t_{Rn} - t_{R1}}$.

Packet pair and packet train bandwidth estimates may be inaccurate due to kernel scheduling, as noted by [66, 133, 98]. To handle the inaccuracies caused by CPU scheduling, Jain et al. [66] discards bandwidth measurements that exhibit low inter-arrival times that are the result of context switching. This results in wasted packets as well as the forgone bandwidth estimate at the time of measurement. This phenomenon caused by kernel interrupts is systematic and common.

A drawback of using self-loading periodic streams for bandwidth estimation is that it causes temporary congestion on the link. This congestion may lead to video data being delayed, which is detrimental to the quality of experience for the video stream. Packet pairs and packet trains are preferable to self-loading periodic streams because they do not temporarily saturate the link unless the probing rate is higher than the available bandwidth, in which case the link would likely not have sufficient bandwidth to stream the video. However, packet pairs and packet trains estimates may not be accurate due to kernel interrupts, and inaccurate estimations can cause video data to be delayed.

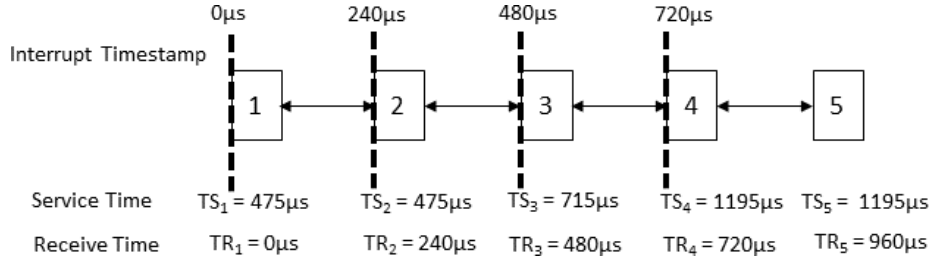


Figure 3.2: Packet Coalescing Example. The timestamps that are provided to the application are the service times (time from when the interrupt is generated to when the packet is actually serviced).

3.1.3 Design

Packet pairs and packet trains send packets in quick succession and use the difference in receive timestamps and the amount of data transmitted to estimate bandwidth. For example, the packet pair technique sends two packets ($Packet_0$ and $Packet_1$) in quick succession of each other. The bandwidth b is calculated by $b = \frac{s_1}{t_{recv}^1 - t_{recv}^0}$ [81], where s_1 is the size of $Packet_1$ and t_{recv}^0 and t_{recv}^1 are the receive times of $Packet_0$ and $Packet_1$, respectively. Similarly, packet trains use the difference in the receive timestamps of first and the last packet and the total amount of data transmitted from the second through n_{th} packets. However, these techniques do not account for additional interrupt processing latencies that exist on the receiver; thus, a packet's timestamp may not accurately reflect the packet's arrival time.

The Impact of Kernel Interrupts on Bandwidth Estimates

Inaccuracies of packet arrival time are largely due to kernel interrupts and scheduling. When a packet arrives at the NIC, a hardware interrupt is generated and sent to the CPU to service the packet. However, before the interrupt is serviced, a second packet may arrive, resulting in the interrupt handler servicing both packets. The receiving process sees both packets as having arrived in quick succession of each other, resulting in a short (perhaps a microsecond or zero) inter-arrival time. The corresponding sending rate calculated using packet-pair dispersion is not possible using the inter-arrival time that the timestamps indicate since the resulting estimation is higher than what the medium is able to support.

Figure 3.2 illustrates the impact caused by short packet inter-arrival times for a packet train of length five. In this example, the available bandwidth of the link is 50 Mbps,

which is obtainable in LTE 4G networks, and each packet is 1500 bytes. As a result, the inter-arrival time between two consecutive packets should be $240\mu s$. However, the receive timestamps of these packets are determined by the interrupt handler which is responsible for time-stamping the packets as they are received.

Suppose that the interrupt handling latency is long ($475\mu s$ from interrupt generation to interrupt servicing). As the first packet is serviced, the second packet arrives. The timestamps provided for bandwidth estimation are the *service* time whereas the *receive* time is the actual time that the packet is received. Every time a packet is received, an interrupt is generated. The bandwidth estimation provided by packet train calculations would be $\frac{1500\text{bytes} \times 8 \times 4}{1195\mu s - 475\mu s} = 66.6\text{Mbps}$. However, because packet coalescing occurs between the first two packets and the last two packets, we remove the first and the last packet from our bandwidth estimate calculation as they do not equally reflect the interrupt handling latency at the beginning and the end. By removing coalesced packets, our bandwidth estimate is now: $\frac{1500\text{bytes} \times 8 \times 3}{1195\mu s - 475\mu s} = 50\text{Mbps}$. We evaluate the impact of kernel interrupts on packet train measurements in Section 3.1.4.

Kernel Interrupt Model

To account for timestamping inaccuracies caused by kernel interrupts, we make the following assumptions in our model:

1. Interrupts are generated when packets arrive at the network card.
2. Timestamps are generated for the packet when they are serviced by the kernel.
3. We treat the processing delay between interrupt generation and servicing as being constant. In practice, processing delay likely falls into a narrow range and distribution [115]. Deviations in interrupt processing latency can lead to inaccurate measurements.
4. We do not assume time synchronization between the sender and the receiver; we only use the receive timestamps to calculate bandwidth.
5. Packets in a packet train are sent in quick succession. With the exception of the first packet in the packet train, packets are serviced by interrupts that were triggered by packets in the same train. The first packet in the packet train may be handled by an interrupt that was generated by a packet belonging to a different application.

Ideally, we would use perfectly accurate timestamps for the first and the last packets in our packet train, and we estimate bandwidth using the packets that arrived between the receive timestamp of the first packet and the last packet in the packet train [41]. Using this method, bandwidth is calculated as:

$$b = \frac{(n - 1) \times PS}{t_{recv}^n - t_{recv}^0} \quad (3.1)$$

where b is bandwidth, n is the total number of packets in our packet train, PS is the packet size, and t_{recv}^i is the timestamp of the packet at the receiver.

Unfortunately, we do not have perfectly accurate timestamps because they are delayed at the interrupt handler. Timestamps can vary based on processing delays and interrupt coalescing, which can cause consecutive packets to have very similar timestamps. For example, an arriving packet may generate an interrupt, and while this packet is being handled, a second packet may arrive. The result is *packet coalescing*, where two packets seem to arrive in quick succession of each other, leading to bandwidth overestimation.

PacketBurst Bandwidth Estimation

We present the PacketBurst technique that obtains accurate bandwidth estimates by considering the different cases of packet coalescing. The primary logic and intuition behind the PacketBurst technique is to compensate for the processing latency resulting from kernel interrupts. To accomplish this, we model packet arrival time and determine if its timestamp fully reflects the kernel servicing time.

In our model, the receive timestamp (t_{recv}^i) is the receive time plus packet-service time. This is expressed as $t_{recv}^i = t_{arrival}^i + t_{proc}$, where $t_{arrival}^i$ indicates the time that $Packet_i$ arrives, and t_{proc} is the interrupt processing latency. The PacketBurst technique ensures that processing latencies (t_{proc} values) are reflected at the beginning and at the end of the packet train. Therefore, the values for t_{proc} must be the same for t_{recv}^0 and t_{recv}^n so that they cancel each other out in our bandwidth estimate calculation. We are not concerned about any packet coalescing in the middle of our packet packet train because their receive timestamps are not used in the packet train formula. Therefore, this leads to four packet coalescing cases for bandwidth estimation:

- a) No packet coalescing at the beginning and at the end of our packet train. (Shown in Figure 3.3a)

- b) No packet coalescing at the beginning of the packet train and packet coalescing at the end of our packet train. (Shown in Figure 3.3b)
- c) Packet coalescing at the beginning at the packet train, and no coalescing at the end of our packet train. (Shown in Figure 3.3c)
- d) Packet coalescing at the beginning and at the end of our packet train. (Shown in Figure 3.3d)

In the first two cases, we do not use the receive timestamp of the first packet in our bandwidth calculation since a packet from another application may have triggered the interrupt. The first packet may have a shorter processing latency than the other packets in the packet train. Because we want all timestamps to contain the entire interrupt processing latency (t_{proc}), we use the receive timestamp of the second packet as t_{recv}^0 in Equation 3.1, and this is shown in Figures 3.3a and 3.3b. By omitting the first packet from the packet train calculation, we do not consider the shorter processing latency that may result in bandwidth overestimation.

In cases (c) and (d), which are depicted in Figures 3.3c and 3.3d, the packet train begins with a group of coalesced packets that are handled by the same interrupt. Suppose that there are i packets in this group. Because we do not know if another application's packet triggered the interrupt that handled the first i packets, we use the receive timestamp of the $i + 1$ packet as t_{recv}^0 in Equation 3.1. The reason for omitting the first i packets is to ensure that the processing latency is equally reflected at the beginning of the packet train and at the end of the packet train.

We now consider the possible values of t_{recv}^n and the numerator (i.e., number of packets counted) in Equation 3.1. In the scenario where there is packet coalescing at the end of the packet train (cases (b) and (d), and Figures 3.3b and 3.3d respectively), we only include the first packet in the coalesced group when counting the packets for our bandwidth estimation. For example, if $Packet_i$ through to $Packet_n$ are coalesced, we include up to $Packet_i$ for the packet count in Equation 3.1's numerator, and we use $Packet_i$'s timestamp for t_{recv}^n in Equation 3.1. This is because $Packet_{i+1}$ through to $Packet_n$ are handled by the same interrupt as $Packet_i$, and their respective timestamps do not include the interrupt latency. $Packet_{i+1}$ through to $Packet_n$ are omitted from the bandwidth estimate calculation because their handling latency is shorter than the first packet's handling latency. Including these packets in the packet train calculation results in overestimation. We do not need to discount packets when there is no coalescing at the end, as shown in Figures 3.3a and 3.3c, because the timestamp of the last packet in the train includes the full interrupt latency.

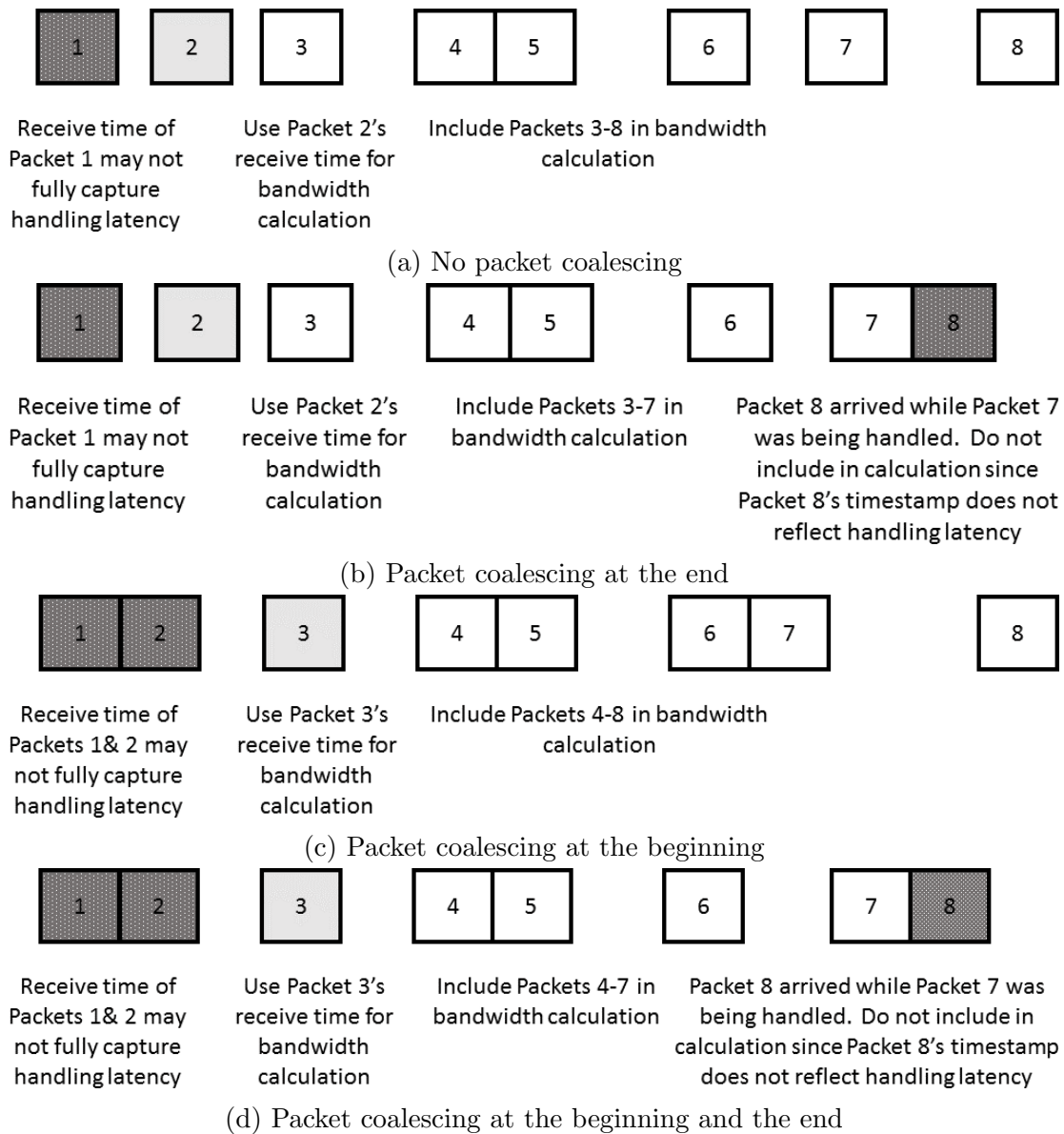


Figure 3.3: Bandwidth estimation using packet bursts

Finally, we must determine the time threshold that classifies if two subsequent packets experience packet coalescing. If we have knowledge of the maximum speed of the link, then we can simply select a sufficiently small threshold. For example, LTE's theoretical

down-link bandwidth is 100 Mbps for a 20 Hz channel [1]. Therefore, MTU-sized packets must have an inter-spacing of at least 120 μs when using a LTE link. Ideally, the threshold for determining if two subsequent packets are serviced by the same interrupt would be the interrupt handling latency of the receiving machine. Using too low of a threshold may cause coalesced packets to be counted in the packet estimate. A significantly high threshold may cause too many packets to be removed, which may result in no packets being left for the PacketBurst bandwidth estimate calculation. Regardless of the threshold, if the receiver services other tasks instead of the packets as they are received, this would lead to a significant under estimation.

3.1.4 Evaluation

We first demonstrate the impact that kernel interrupts have on packet train estimates. Our evaluation results show that using short packet train sizes results in significant over-estimations. We then evaluate the effectiveness of our PacketBurst bandwidth estimation technique that accounts for kernel interrupts and packet coalescing, and we compare against packet train estimates using the formula found in cprobe [22]. The method and formula for bandwidth estimation using packet trains is common to many packet train approaches that involve sending packets in short succession and calculating their inter-arrival times [117, 40].

Kernel Interrupts’ Impact on Bandwidth Estimates

We first demonstrate the impact of kernel interrupts that occur in practice in the following experiments. Using Mininet [100] as our network emulator, we send packet trains of varying lengths and calculate the estimated available bandwidth using the formula: $bw = \frac{PacketSize \times (TrainLength - 1)}{LastPacketRecvTS - FirstPacketRecvTS}$. We also vary the available bandwidth on the link to determine the impact that the available bandwidth has on presence of packet interrupts.

In the experiment for Figure 3.4, we evaluate the impact that the train length has on the available bandwidth estimate. For this experiment, we fix the available bandwidth to be 20Mbps, and we vary the train length. We use a MTU-sized packet of 1460 bytes in our packet trains. We collected the received kernel timestamps 100 packet trains and estimated the available bandwidth using the aforementioned packet-train bandwidth estimation formula. We plot a boxplot of the bandwidth estimates in Figure 3.4 for packet train lengths that are greater than four. We omit packet train size of two since the estimates were significantly greater than the estimates produced by trains of length four or greater.

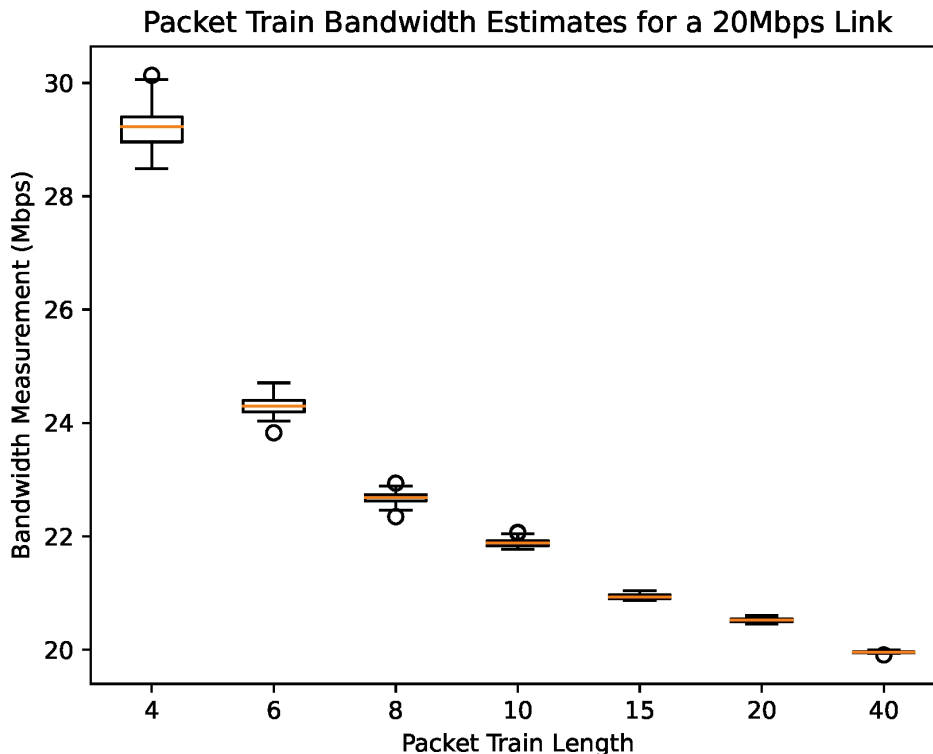


Figure 3.4: Packet Train Length - Bandwidth Estimates

As the train length grows, as shown in Figure 3.4, our packet train estimates become more accurate in terms of absolute and percentage error. This is because the kernel interrupt, caused by the CPU scheduler, has a lower impact for larger train lengths. Therefore, if the train length is sufficiently large, then we do not need to be as concerned regarding the impact of kernel interrupts. From Figure 3.5, we can see the percentage error of the bandwidth estimates. For a train length of four, the percentage error can be as great as 50%. This decreases to 10% even if the train length more than doubles to ten. These figures suggest that there is a systematic error when using timestamps for bandwidth estimation, and the error is reduced as the train size increases.

We now investigate the impact that the available bandwidth has on the accuracy of packet train bandwidth estimates. In this experiment, we fix the packet train size to be four, and we vary the available bandwidth from 0.5 Mbps to 20.0 Mbps. Figure 3.6 depicts a box plot of 100 packet train measurements for varying link capacities. As the

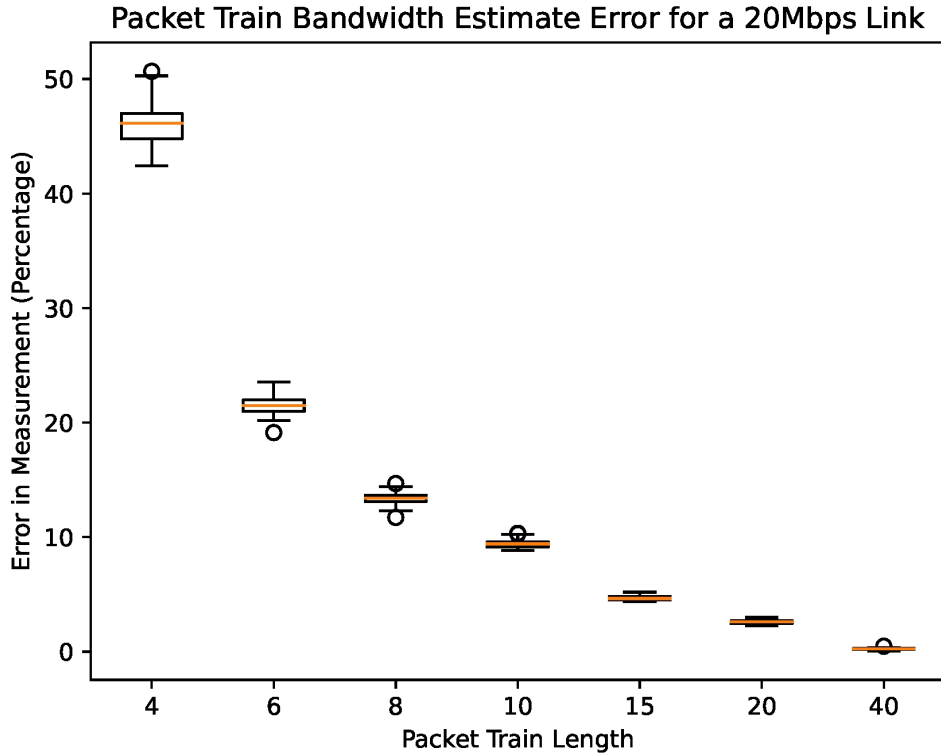


Figure 3.5: Packet Train Length - Bandwidth Estimation Percent Error

available bandwidth increases, so does the error on the packet estimates. Figure 3.7 shows the percentage error of the packet train estimates as the available bandwidth increases. Regardless of the available bandwidth, a packet train length of four yields at least a 44% percentage error, and the range in error is greater as the available bandwidth increases.

Figure 3.4 and Figure 3.6 demonstrate the impact that kernel interrupts and packet scheduling have on the accuracy of packet pairs and trains. As the number of packets increases, we can obtain more accurate bandwidth measurements. However, to utilize longer packet trains for bandwidth estimation, a streaming application needs to buffer data until there are sufficient number of bytes; this causes an increase in the application's delay. Ideally, a bandwidth estimation technique that can use shorter trains would be more beneficial, especially if the IoT application does not need to stream data at high rates.

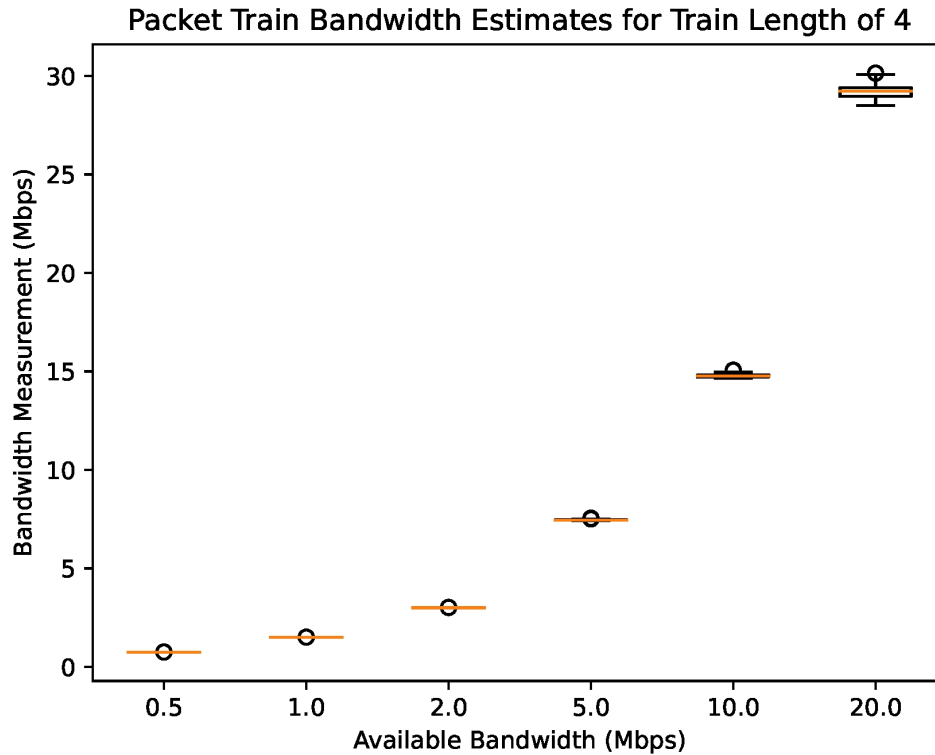


Figure 3.6: Available Bandwidth Impact on Bandwidth Estimates

Bandwidth Estimation

We now evaluate our PacketBurst technique that addresses timestamp inaccuracies caused by kernel interrupts. We use a Mininet emulation, and our setup consists of a sender node, a receiver node, and a connecting link. An application on the sender node sends a train of equally sized packets. Packet trains are spaced apart such that the link does not experience congestion. The receiver node logs the receive kernel timestamp of all packets and their respective packet ID and train number. We estimate the available bandwidth using the packet train formula and our PacketBurst technique using the logs collected at the receiver.

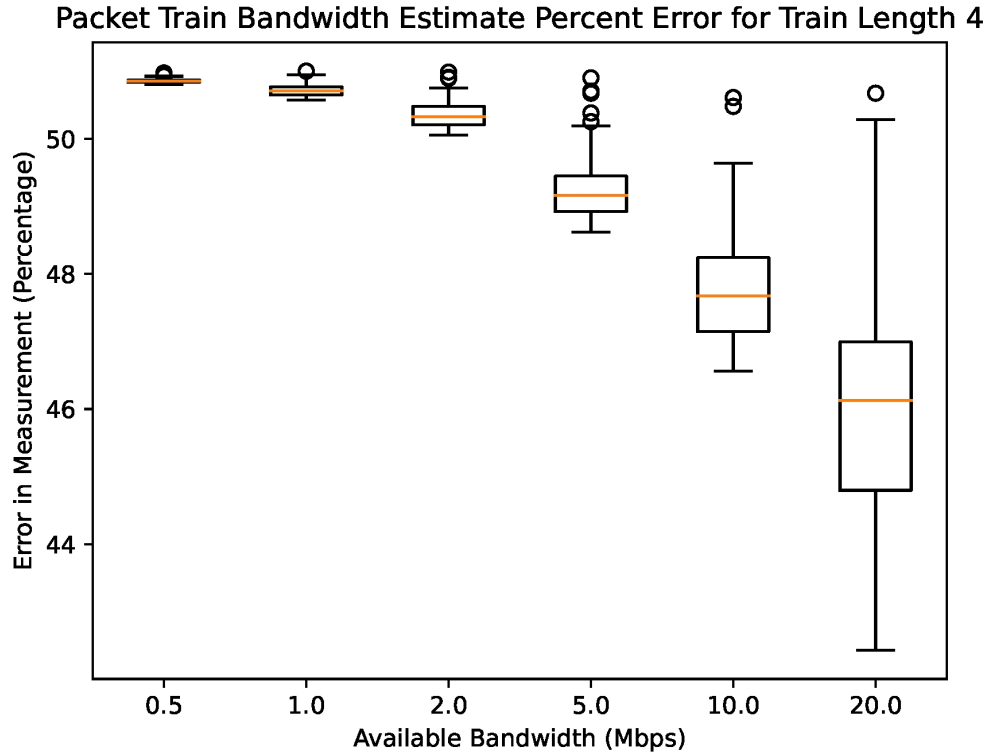


Figure 3.7: Available Bandwidth Impact on Bandwidth Estimation Percent Error

		512-byte Packet		1024-byte Packet		1454-byte Packet	
		Packet Train	Packet Burst	Packet Train	Packet Burst	Packet Train	Packet Burst
Train Size	2	318.73, 504.22	0.0, 0.0	672.26, 1017.8	0.0, 0.0	764.23, 1197.0	0.0, 0.0
	4	11.81, 0.18	0.0, 0.0	0.96, 0.0	0.0, 0.0	0.75, 0.0	0.01, 0.07
	6	1.09, 0.0	0.47, 0.04	0.69, 0.0	0.48, 0.01	0.62, 0.0	0.49, 0.0
	8	0.79, 0.0	0.46, 0.0	0.61, 0.0	0.48, 0.0	0.57, 0.0	0.49, 0.0
	10	0.68, 0.0	0.46, 0.01	0.58, 0.0	0.48, 0.0	0.55, 0.0	0.49, 0.0
	15	0.58, 0.0	0.46, 0.0	0.54, 0.0	0.48, 0.0	0.53, 0.0	0.49, 0.0
	20	0.54, 0.0	0.46, 0.0	0.52, 0.0	0.48, 0.0	0.51, 0.0	0.49, 0.0

Table 3.1: Packet Train and PacketBurst (Average, Standard Deviation) Bandwidth Estimation Comparisons for 0.5 Mbps link

Because wireless links can have a wide range of available bandwidth, we first evaluate our PacketBurst technique using a link of 500 kbps. In Table 3.1 each column indicates the packet size in bytes and the method of estimating the available bandwidth, and each row indicates a different train length. We present the average bandwidth estimate and the standard deviation of the bandwidth estimate.

For our PacketBurst technique in Table 3.1 and Table 3.2, a value of 0 means there was insufficient packet information to estimate the available bandwidth. Because the PacketBurst technique involves removing packets that exhibit packet coalescing at the beginning and end of the train, we find that a minimum of six, 512-byte packets are required to obtain useful estimates. As packet sizes and train lengths become larger, the estimates approach 500 kbps. The bandwidth estimate’s percentage error is calculated as:

$$\left| \frac{BW_{estimate} - BW_{actual}}{BW_{actual}} \right| \times 100\%$$

In the case of six 512-byte packets, the percentage error for packet trains is $\left| \frac{1.09-0.5}{0.5} \right| \times 100\% = 118\%$ and the percentage error for PacketBurst is $\left| \frac{0.47-0.5}{0.5} \right| \times 100\% = 6\%$. From Tables 3.1 and 3.2, the percentage error for packet train estimates is often higher than the PacketBurst technique. Using six, 512-byte packets in a packet train was determined experimentally for our environmental setup. The minimum number of packets in a packet train and the packet size to use depends on the speed of the link, the processor, and the operating system.

		512-byte Packet		1024-byte Packet		1454-byte Packet	
		Packet Train	Packet Burst	Packet Train	Packet Burst	Packet Train	Packet Burst
Train Size	2	412.21, 684.71	0.0, 0.0	501.0, 605.6	0.0, 0.0	802.09, 1153.22	0.0, 0.0
	4	78.01, 7.08	0.0, 0.0	36.3, 0.77	0.0, 0.0	29.18, 0.35	0.0, 0.0
	6	42.53, 1.21	0.0, 0.0	26.85, 0.26	18.82, 0.28	24.32, 0.16	19.45, 0.25
	8	31.27, 0.32	18.73, 0.98	24.12, 0.13	19.03, 0.16	22.69, 0.08	19.45, 0.1
	10	27.14, 0.17	18.62, 0.6	22.84, 0.07	19.05, 0.08	21.88, 0.07	19.44, 0.09
	15	23.22, 0.16	18.47, 0.35	21.4, 0.06	19.12, 0.06	20.95, 0.05	19.45, 0.09
	20	21.76, 0.07	18.53, 0.11	20.77, 0.04	19.14, 0.05	20.53, 0.03	19.45, 0.08

Table 3.2: Packet Train and PacketBurst (Average, Standard Deviation) Bandwidth Estimation Comparisons for 20 Mbps link

Packet trains are subject to more packet coalescing at higher rates as there are more opportunities for packets to arrive while the interrupt is being serviced. We increase the available bandwidth of the link to 20Mbps for our experiment summarized in Table 3.2.

Again, we find that the PacketBurst technique is unable to utilize short packet train lengths and small packet sizes and it requires a minimum train length and packet size. For for a packet train of six MTU-sized packets and 20 Mbps of available bandwidth (Table 3.2), the PacketBurst technique experiences an $|\frac{19.45-20.0}{20.0}| \times 100\% = 2.75\%$ error whereas packet trains experience a $|\frac{24.32-20.0}{20.0}| \times 100\% = 21.6\%$ error.

Occurrences of Packet Coalescing

We find that using the PacketBurst technique is necessary for nearly all packet trains. In all cases, we must remove the first packet in the packet train since it may not fully capture handling latency. The inaccuracy of packet train estimates increases if there is packet coalescing at the end of the train, which leads to a significantly higher bandwidth estimate.

We now determine the number of occurrences per case, where each case is as defined in Fig. 3.2. The least amount of coalescing is Case (a) where there is no additional coalescing at the beginning or at the end. Using the packet trains of length six or higher, we find that there are 7 and 3 occurrences of out of a total of 500 packet trains that exhibited the Case (a) coalescing pattern for a 0.5 Mbps a and 20.0 Mbps respectively. The remaining packet trains are classified as Case (c) where coalescing happened at the beginning of the packet train. We did not have any occurrences of Cases (b) and (d) where packet coalescing occurred at the end of the train. This demonstrates that there is a systematic error at the beginning of the packet train, and our PacketBurst technique removes this systematic error. In our analysis, we found that in all cases, there were two packets at the beginning of the train that were coalesced with each other.

3.1.5 Real Testbed with Emulation

We now evaluate our PacketBurst technique against packet trains at higher rates on a real testbed where we introduce jitter and burstiness, which would introduce inaccuracies in the packet train estimates. Our testbed consists of a sender and a receiver (Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz) connected by a Ethernet link which we modify using Ubuntu’s Traffic Control Utility (*tc*) [116]. Unless otherwise specified, we present the mean and standard deviations of 100 packet trains. We have selected a bandwidth rate of 20 Mbps as they reflect the available uplink bandwidth of wireless links that our target devices typically utilize [5, 24]. Fig. 3.8 illustrates our test-bed. The latency is defined as the time the link requires to transmit a packet from the sender to the receiver, which is 100 ms.

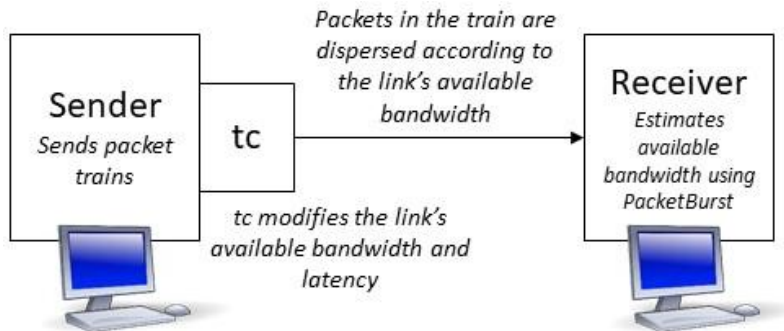


Figure 3.8: Testbed Setup

Impact of Latency and Jitter

In this section, we introduce jitter into our testbed setup. We use *netem*, which is component of *tc*, to set the available bandwidth using the command: `netem delay 100ms 10ms rate 20mbit`. With the introduction of jitter, we expect bandwidth estimates to be lower as the jitter would periodically increase packet inter-arrival time.

Table 3.3 presents our results, and we find that packet trains significantly under predicts the available bandwidth. This is due to jitter, which may cause the packet inter-arrival time to be greater. Our experiment also shows that using the PacketBurst technique improves packet train bandwidth estimates for train sizes as low as four. Because packet train estimates are calculated using the first and last receive timestamp, these estimates may be too low as the first and last packet inter-arrival times may be elongated due to jitter. The PacketBurst technique removes the first packet from its bandwidth estimate calculation as it does not fully capture the interrupt handling latency. By doing so, one source of additional latency is removed from the estimate, resulting in a higher estimate that is closer to the available bandwidth on the link.

Although Table 3.3 has shown that the PacketBurst technique can address inaccuracies caused by jitter, we find that a significant amount of jitter impacts the PacketBurst technique's ability to rectify the packet train's under estimation. Table 3.4 shows the packet train and PacketBurst bandwidth estimation when jitter is increased to 50 ms. The PacketBurst technique is only able to obtain an estimation that is 1.76 Mbps better than packet trains for a train size of 12. We find that in both these cases, packet coalescing is not observed. Nonetheless, the PacketBurst technique is able to improve upon the packet train estimates.

Train Size	Packet Train		Packet Burst	
	Mean	Std	Mean	Std
2	13.49	9.51	-	-
4	10.25	7.72	14.26	7.45
6	9.27	6.3	12.99	6.85
8	10.05	5.74	13.08	5.87
10	10.8	5.43	14.95	5.23
12	11.7	4.95	14.24	5.01
14	11.98	4.78	14.27	4.96
16	13.01	4.76	15.46	4.24
18	12.68	4.3	15.08	3.95
20	13.92	3.97	16.06	3.5

Table 3.3: Packet Train and PacketBurst Bandwidth Estimates Mbps (Average, Standard Deviation) in Mbps for Testbed with 100 ms latency, 10 ms jitter, and 20 Mbps

Train Size	Packet Train		Packet Burst	
	Mean	Std	Mean	Std
2	10.29	10.29	-	-
4	7.37	8.51	11.51	9.17
6	6.44	7.63	11.03	8.61
8	5.59	6.47	9.09	8.0
10	5.34	6.3	8.16	7.4
12	6.85	6.87	8.61	7.06
14	6.01	5.75	8.78	6.47
16	6.44	6.22	9.14	6.9
18	6.59	5.51	8.79	6.02
20	6.28	4.89	9.38	5.9

Table 3.4: Packet Train and PacketBurst Bandwidth Estimates Mbps (Average, Standard Deviation) in Mbps for Testbed with 100 ms latency, 50 ms jitter, and 20 Mbps

3.1.6 Discussion and Summary of PacketBurst

The impact of operating system schedule and kernel interrupts has been acknowledged in many related works [66, 133]. However, of our surveyed works, none address how we may account for discrepancies in receive timestamps as a result of the scheduling. Our PacketBurst bandwidth estimation technique accounts for interrupt latency and packet coalescing and is able to improve upon packet train estimations.

Having an accurate bandwidth estimate is important as adaptive video bitrate protocols often use this estimate to select the highest video bitrate [12, 16]. A packet train estimate may indicate the maximum video bitrate that can be obtained while not increasing the user’s delay. Consequently, improving these estimations can also improve the viewer’s quality of experience.

The PacketBurst estimation technique can only be used to improve upon packet train estimates. If the packet train is insufficiently long to capture traffic shaping, then it cannot be expected that the PacketBurst technique is able to correctly estimate the bandwidth. An example of this is found in LTE where Balasingam [14] showed that packet trains need to be sufficiently long such that they span multiple scheduling cycles; otherwise, packet trains measure the network’s capacity (the medium’s rate) as opposed to the available bandwidth.

In summary, the PacketBurst technique can be used to obtain more accurate bandwidth estimates than packet trains. We first identify the systematic inaccuracies of packet train bandwidth estimates caused by packet coalescing. We identify four cases of packet coalescing and remove coalesced packets from packet train calculations to obtain a more accurate bandwidth estimate. The PacketBurst technique removes coalesced packets from the packet train to obtain a more accurate bandwidth estimate. It can be applied to existing packet-train-based bandwidth estimation methods to reduce the error caused by kernel interrupts, thereby making packet trains practical for live video streaming protocols.

3.2 Machine Learning for Predicting Link Instability

Adaptive video bitrate protocols are used to provide video streaming applications with a smooth, high-quality video stream while using highly variable links. To accomplish this, it is important for these protocols to consider aspects such as latency and loss. Latency can cause video to arrive past its deadline and stall, and the presence of packet loss can cause artifacts in the video stream. The ability to predict *link instability*, which we define

as sharp increases in latency and application-level packet loss, or sudden drops in available bandwidth over a short period of time, is advantageous for adaptive video bitrate protocols. These protocols can use link instability predictions to reduce their bitrate or to send redundant data to ensure that their data arrives on time. These link instability predictions can be made using ML techniques.

3.2.1 Contributions

In this section, we make the following contributions:

- Firstly, we analyze traces of large, Canadian LTE providers that were provided to us from a live video streaming service provider. We characterize these traces and show that LTE links are highly variable, thereby motivating the need for instability prediction.
- Secondly, we describe example construction, which maps a set of attributes to a classification of stable or unstable, using the provided raw data. The classification is used for prediction.
- Thirdly, we determine the attributes that provide the greatest predictive ability for various definitions of link instability using Analysis of Variance (ANOVA) f-statistics and Recursive Feature Elimination (RFE) rankings.
- Finally, we evaluate the classification and false positive rates of various ML techniques. This contribution provides us with an overview of the predictive ability of many ML techniques and with insight on the techniques to use when predicting changes in bandwidth, latency, and loss. Specifically, we find that DTs and KNNs are able to provide a high classification rates while maintaining low false positive rates.

3.2.2 Trace Description

In this Section, we analyze traces that were collected using a multi-homed, commercial video streaming application that was provided to us by the application developer. The application’s video streaming session was stationary (i.e., no movement), half an hour in duration, and consisted of aggregating four LTE links concurrently. The LTE providers are Canadian and include Telus, Bell ×2, and Rogers. Over the duration of the video

stream, the application periodically collected network metrics such as mean latency, two standard deviations of latency for a recent window (between feedback intervals), jitter, and instantaneous measured received bandwidth.

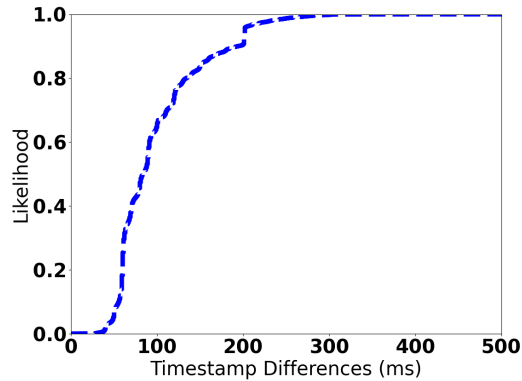
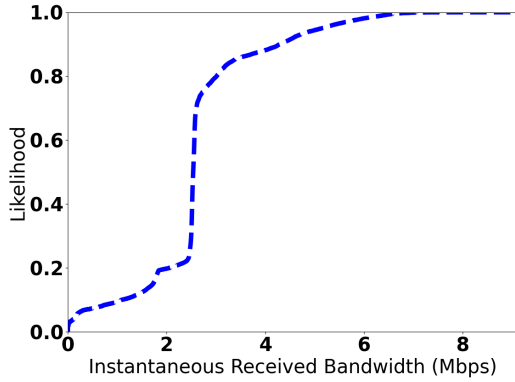


Figure 3.9: CDF of Measurement Intervals (ms)

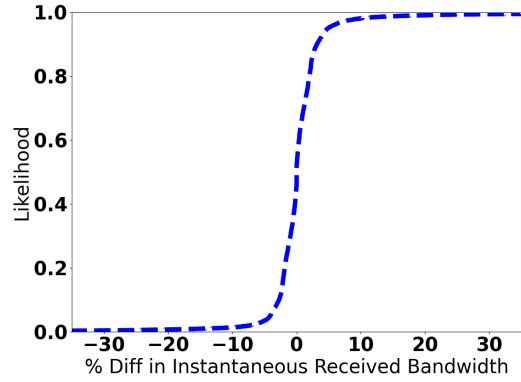
Because of the application’s design, the inter-arrival time between consecutive network measurements are not equal and depend on the amount of data received by the receiver. Network metrics collected during the video streaming session are shown in Figure 3.9 which depicts the Cumulative Distribution Function (CDF) of the difference in feedback intervals for our four traces. The average interval of our measurement samples is 104.5 *ms* with a standard deviation 1899.3 *ms*. The maximum difference in our measurement samples is approximately 500 *s*, which indicates that the link has failed at some point. We find that 90% of our feedback intervals are below 194 *ms*, and 99% of are feedback intervals are below 254 *ms*.

We analyze the feedback information that is provided to the application which includes available bandwidth, packet loss, latency, and jitter. Firstly, Figure 3.10a presents the CDF of the instantaneous received bandwidth, and Figure 3.10b presents the CDF of the percentage changes in the available bandwidth between subsequent measurements. On average, there are 2.6 *Mbps* available, with a standard deviation of 1.2 *Mbps*. The minimum bandwidth experienced during the testing period was 0.0 *Mbps* up to a maximum of 9.1 *Mbps*. We find that the measured bandwidth dropped by more than 5% for 3.43% of our measurements, where measurements are typically conducted within 100-200 *ms* of each other. This roughly translates to a 17% - 30% drop in available bandwidth within a second.

Secondly, our traces indicate that 99.8%, with a standard deviation of 2.1%, of the pack-

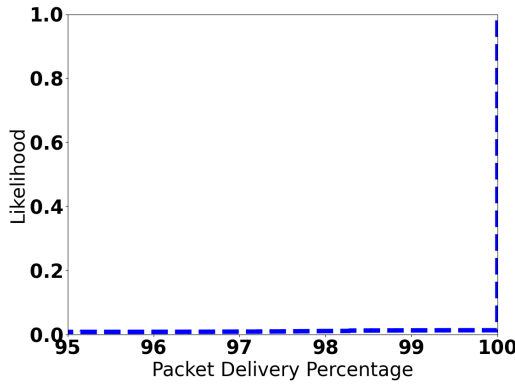


(a) CDF Recv BW

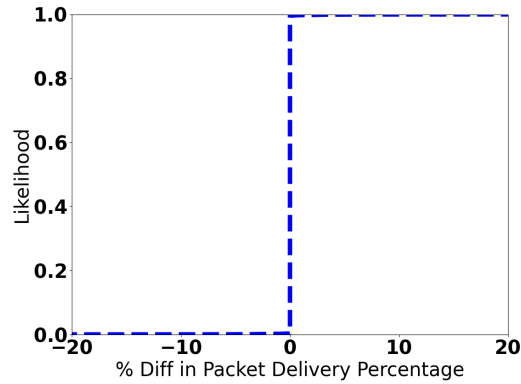


(b) CDF % Change

Figure 3.10: Instantaneous Received Bandwidth (Recv BW)



(a) CDF Packet Delivery Percentage



(b) CDF % Change

Figure 3.11: Analysis on Packet Delivery Percentage

ets arrived. The packet delivery percentage ranges between 8.09% to 100%, and 1.37% of our observations indicate a packet delivery percentage of less than 100%, which represents 147 *s*. On average, the packet delivery percentage changes 0.0085% with a standard deviation of 2%.

Thirdly, Figure 3.12a shows that the mean latency is 68.9 *ms* with a standard deviation of 65.3 *ms*. Typically, real-time communication requires latency to be lower than 150 *ms* [10]. Our traces exceed this threshold for 1.4% of our measurements, which accounts

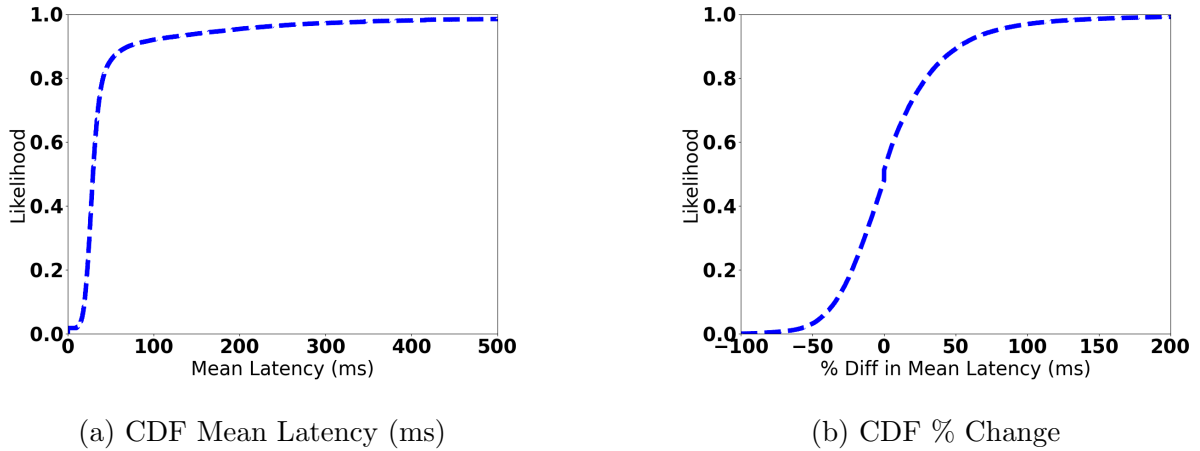


Figure 3.12: Analysis on Mean Latency

for 108.83 s. The difference in mean latency between subsequent measurements is 8.7% with a standard deviation of 412.75%. This demonstrates that mean latency measurements in wireless are highly variable.

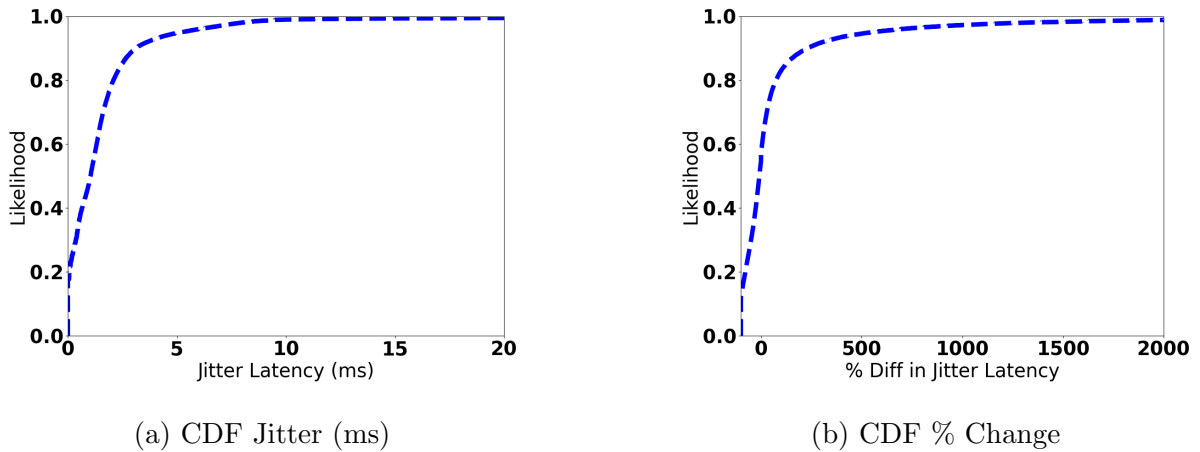


Figure 3.13: Analysis on Jitter

We depict the amount of jitter in Figures 3.13a and 3.13b. On average, there is 1.8 ms of jitter with a standard deviation of 7 ms. The maximum jitter that was experienced was 258.1 ms. On average, jitter changes 80.4% between measurement intervals with a standard deviation of 412.8%

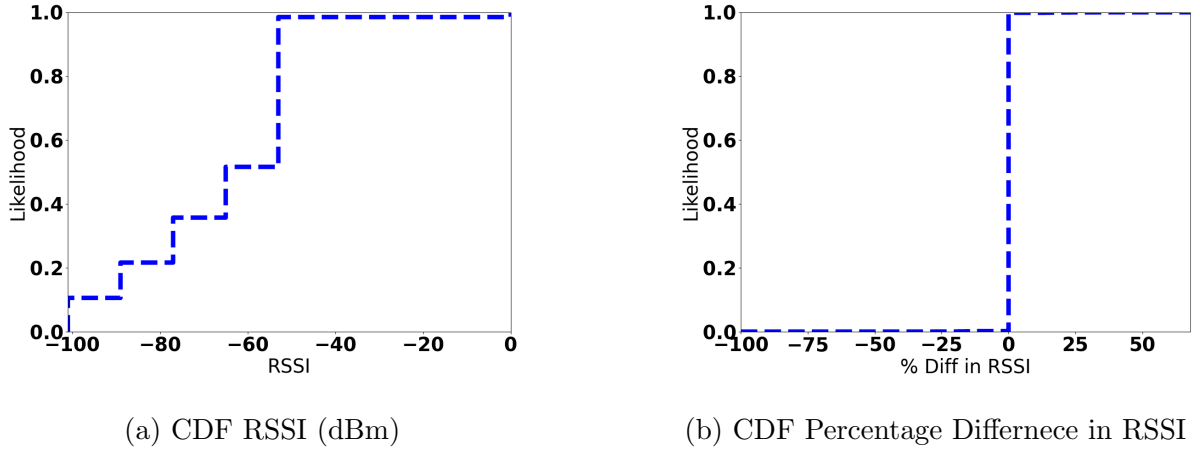


Figure 3.14: Analysis on RSSI

We analyze Received Signal Strength Indicator (RSSI). Figures 3.14a and 3.14b shows CDF of the RSSI measurements and the percentage difference of RSSI. On average, the mean RSSI is -66.5 dBm with a standard deviation of 18.7 dBm. The minimum RSSI experienced is -100 dBm. An RSSI value of -67 dBm or higher indicates a fairly strong signal, and a value of -90 dBm or lower indicates a poor signal [130]. We find that the RSSI value of below -90 dBm occurs for 0.006% of the measurements or for 0.68 s of the traces. On average, the percentage difference between measurements is 0.002% with a standard deviation of 1.48% .

3.2.3 Classifying Link Instability

In this section, we specify our definitions of link instability. Using these definitions, we describe example construction for training and testing using our raw data. We analyze attributes to determine those that best predict link instability. Finally, we outline the ML techniques that we use to predict link instability.

Link Instability Definitions

Many of the systems presented in Section 2.2.2 focus on using deep-learning techniques, decision trees, or random forests for bandwidth prediction. However, within the wireless environment, bandwidth predictions may not be valid for an extended period of time.

Latency and packet loss are also important metrics to consider when characterizing link instability as they affect the link’s ability to deliver data in a timely manner. We investigate the following definitions of link instability:

- **Drop in Available Bandwidth (BWD):** A link is unstable if its measured bandwidth is lower than the average bandwidth minus two standard deviations for the last two seconds. A significant drop in available bandwidth may be an indicator of poor signal strength or congestion. We select this threshold as it may indicate lower available bandwidth in the future for the video stream.
- **Recent Latency Increase (RLI):** The link is unstable if the minimum latency for the last second is greater than the minimum latency for last ten seconds. The increase in minimum latency for the recent time window indicates the presence of queuing delay, as inspired by COPA congestion control [13]. We select this signal as it is an indication of congestion, which may result in delayed video data.
- **Packet Loss (Loss):** The link is unstable if it exhibits an application-level packet loss of 2% or more. The loss is experienced by the application as it uses UDP to stream video. This threshold was selected as Tommasi et al. [139] found that a loss rate of 2% or more caused video streaming metrics such as Peak-Signal-to-Noise-Ratio (PSNR) to drop by half.
- **Highly Variable Latency (LatI):** A link is unstable if the latency is greater than the mean latency plus two standard deviations of latency measurements for the last two seconds. We select this threshold as a sudden increase in latency may be due to increased queuing delay resulting from congestion. As congestion can cause delayed data, we say that a link is unstable if it experiences a sudden increase in latency above this threshold.

Example Creation

Figure 3.15 illustrates how we construct our attributes and classification which form our training and testing examples. Our timestamped logs contain instantaneous measurements of latency, jitter, received bandwidth, packet arrival rate, and RSSI. Each measurement is represented by a circle with a label M_i . As shown in Figure 3.15, a series of timestamped, raw measurements is used for attribute construction and is referred to as our *history* window as it represents the collected measurements that we can use for link instability prediction. In Figure 3.15, M_1 , M_2 , M_3 form the history window. Because we do not have regular

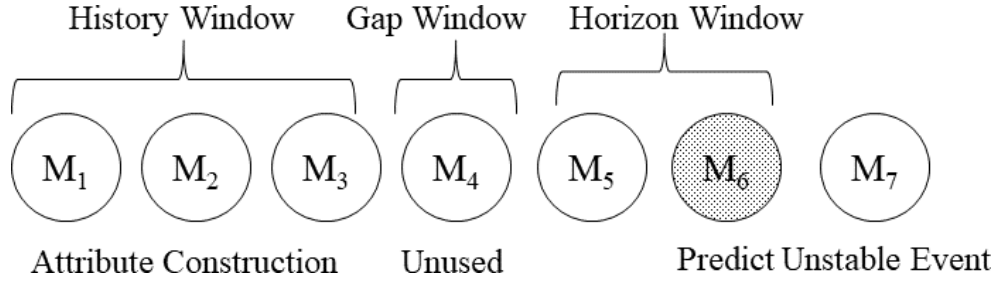


Figure 3.15: Example creation from raw traces.

Attribute	f-statistic, p-value			
	BWD	RLI	Loss	LatI
Recv BPS (avg)	15954.0, 0.0	15850.4, 0.0	50.4, 0.0	2172.5, 0.0
Recv BPS (r-value)	151.9, 0.0	70.9, 0.0	39.0, 0.0	562.7, 0.0
Recv BPS (slope)	165.0, 0.0	155.8, 0.0	7.7, 0.0	515.9, 0.0
Recv BPS (std)	78.7, 0.0	2276.2, 0.0	11.8, 0.1	11796.6, 0.0
Jitter (avg)	179.1, 0.0	9.8, 0.2	1422.8, 0.0	2986.5, 0.0
Jitter (r-value)	23.9, 0.0	77.3, 0.0	2.8, 0.4	197.0, 0.0
Jitter (slope)	10.7, 0.0	30.2, 0.0	29.3, 0.0	148.8, 0.0
Jitter (std)	42.0, 0.0	33.7, 0.2	574.2, 0.0	2359.1, 0.0
Mean Latency (r-value)	30.3, 0.0	10.7, 0.4	5.9, 0.3	65.9, 0.0
Mean Latency (slope)	25.2, 0.0	21.7, 0.0	427.3, 0.0	396.1, 0.0
Delivery % (avg)	2317.7, 0.0	53.8, 0.1	29.6, 0.0	149.1, 0.0
Delivery % (r-value)	86.9, 0.0	3.5, 0.2	1.4, 0.6	10.3, 0.2
Delivery % (slope)	99.9, 0.0	9.0, 0.1	0.6, 0.6	0.8, 0.5
Delivery % (std)	72.9, 0.0	41.3, 0.0	15.4, 0.1	2.6, 0.3
RSSI (avg)	421.3, 0.0	5145.8, 0.0	65.9, 0.0	41753.2, 0.0
RSSI (r-value)	48.9, 0.0	25.4, 0.0	35.7, 0.1	159.9, 0.0
RSSI (slope)	5.5, 0.2	3.3, 0.3	21.9, 0.0	106.6, 0.0
RSSI (std)	547.1, 0.0	173.3, 0.0	16.9, 0.1	388.0, 0.0

Table 3.5: (f-stat, p-value). Bold numbers indicate attributes that are selected for example construction.

sampling intervals, the size of our observation history window is specified in terms of number of measurements. Attributes are summaries of our raw measurements that include the mean, standard deviation, slope, and correlation (r-value).

Attribute	RFE Ranking			
	BWD	RLI	Loss	LatI
Recv BPS (avg)	3.6	10.4	7.6	10.8
Recv BPS (r-value)	7.4	9.8	7.2	5.8
Recv BPS (slope)	4.0	9.6	8.4	9.4
Recv BPS (std)	2.6	14.6	6.4	8.8
Jitter (avg)	6.0	8.2	6.8	10.6
Jitter (r-value)	14.4	4.4	10.4	6.8
Jitter (slope)	15.8	15.6	13.2	16.2
Jitter (std)	8.0	5.6	7.4	8.2
Mean Latency (r-value)	10.4	8.4	8.6	11.8
Mean Latency (slope)	12.6	5.6	9.4	9.4
Delivery % (avg)	3.8	8.0	4.2	4.6
Delivery % (r-value)	13.2	11.6	14.0	7.8
Delivery % (slope)	17.2	17.4	18.0	18.0
Delivery % (std)	9.0	4.4	11.8	6.4
RSSI (avg)	5.0	7.6	7.2	6.4
RSSI (r-value)	13.8	6.4	8.4	5.6
RSSI (slope)	17.8	17.2	13.4	15.6
RSSI (std)	6.4	6.2	8.6	8.8

Table 3.6: RFE rank. Bold numbers indicate attributes that are selected for example construction.

Figure 3.15 also illustrates the introduction of a *gap window*, represented as M_4 , that follows immediately from the history window. Raw data in the gap window is not used for attribute construction or for classification/prediction. The gap window represents the time that is used by the application to use the link instability prediction for deciding the appropriate actions to take. A gap window is required because video encoders often need time to lower the video bitrate smoothly. Insufficient time to lower the video bitrate results in larger frames or a higher data rate than what the link can support, resulting in congestion and delayed video data.

Finally, as shown in Figure 3.15, we define a *horizon window* that is used to determine the classification, or label, that indicates the link’s instability. An example’s classification is either unstable or stable. In Figure 3.15, the horizon window is represented by M_5 and M_6 . M_6 represents the unstable event, resulting in a classification of unstable for the

constructed example.

To construct our ML examples, we first partition our traces based on the definition of link instability. Multiple examples are constructed using a partition that ends with the unstable event. For example, let $M_1, M_2, \dots, M_{n-1}, M_n$ be subsequent measurements that are used to create an example. M_1 to $M_{HistorySize}$ are used to construct the attributes for our first example. The classification of this example depends on the presence of an unstable event in the horizon window, which is defined as the period between $TS(M_{HistorySize}) + GapSizeMs$ and $TS(M_{HistorySize}) + GapSizeMs + HorizonWindowSize$. We continue to construct examples by considering subsequent measurements; the next example would use measurements M_2 to $M_{HistorySize+2}$ to create the attributes.

Attribute Analysis

Ideally, attributes that provide the greatest information gain should be selected when creating examples for training a ML model. The appropriate tests to determine these attributes are ANOVA's f-statistic [37] and RFE [122] since our examples map numerical data to categorical data.

ANOVA's f-statistic compares the ratio of the variance between groups and the variance within a group (or the ratio of explained variance to unexplained variance). Intuitively, it indicates the predictive capability of the attribute. The p-value must be less than 0.05 and the f-statistic must be greater than 1.0 for the attribute to be considered significant.

RFE fits a model and removes the weakest attribute until the desired number of features is met. In our analysis, our target number of features is one, and we obtain rankings regarding the predictive ability of our different attributes.

Table 3.5 indicates the ANOVA f-statistic, and Table 3.6 indicates the RFE ranking for all of our attributes under the different definitions of link instability (BWD, RLI, Loss, and LatI). Because we have many examples, we randomly select a subset to determine the f-statistic and RFE ranking. We repeat this process five times and present the average.

Machine Learning Techniques

After constructing our examples, we evaluate the following ML techniques for link instability prediction by using sklearn's implementation of these methods:

- **Naive-Bayes (NB):** Uses Bayes-Theorem that assumes independence among predictors.
- **Logistic Regression (LR):** Coefficient or weights are learned to produce a 0 or 1 for classification
- **Stochastic Gradient Descent (SGD):** Maximizes an objective function. It learns a linear scoring function to make a prediction.
- **K-Nearest Neighbours (KNN):** The distance between the queried example is calculated between the training examples, and the classification is the majority's classification. We use neighbourhood sizes of 2, 5, and 10.
- **Decision Trees (DT):** Learns simple rules from attributes. We evaluate tree heights of 5, 10, 15, 20 and unbounded.
- **Neural Networks (NN):** A deep learning method that has been heavily investigated by literature. We evaluate the identity, tanh, relu, and logistic activation functions and use early stopping.
- **Support Vector Machines (SVM):** Partitions data using a hyperplane. We investigate polynomial functions with degrees between two to four.

We use $\frac{3}{4}$ of the examples for training and the remaining $\frac{1}{4}$ for testing. The top five attributes as determined by ANOVA's f-statistic and by RFE are used to construct the examples, which we have indicated in bold in Tables 3.5 and 3.6.

3.2.4 Evaluating Machine Learning Techniques

We evaluate the ability of different ML techniques to predict different definitions of link instability. We search through many combinations of history, gap, and horizon window sizes as well as different boosting percentage levels to determine the ML technique and the window settings that provide the highest classification rate.

Metrics

To evaluate the effectiveness of different ML techniques and training examples, we consider the *unstable event* prediction rate and the false positive rate. An *unstable event*

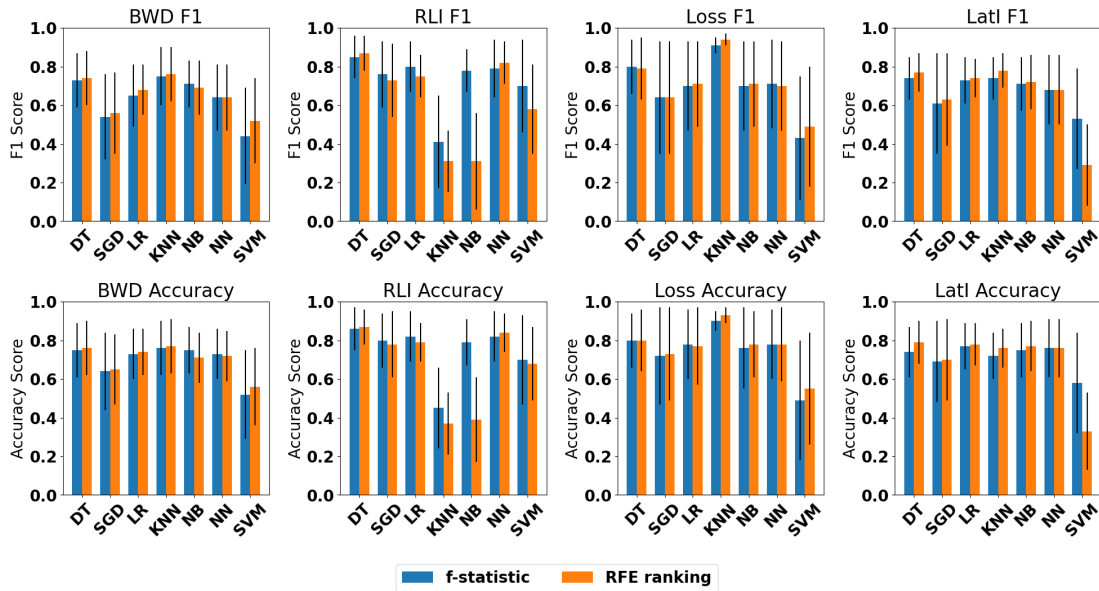


Figure 3.16: Average and StdDev of F1 and Accuracy score for different ML techniques for different Link Instability definitions

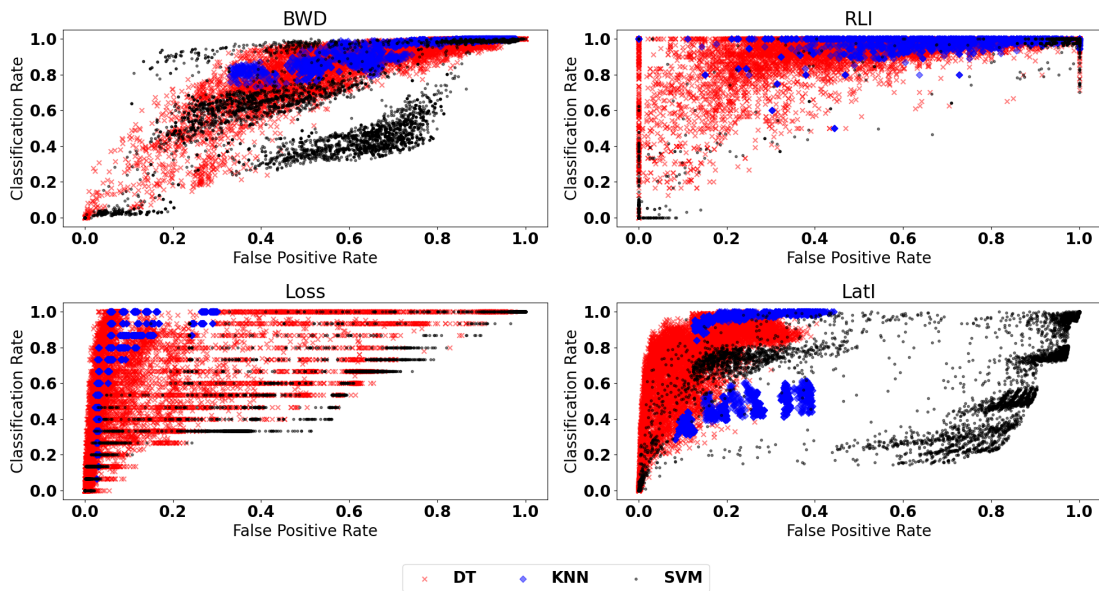


Figure 3.17: False Positive Rate v. Classification Rate for DT, KNN, and SVM

I Def.	FP Rate	Class Rate	ML Technique Description
BWD	0.0	0.05	DT (7, 1000, 1000, 25, f-statistic)
	0.01	0.15	DT (10, 250, 2000, 10, RFE)
	0.1	0.46	DT (10, 250, 2000, 10, RFE)
	0.25	0.94	SVM (10, 1250, 1000, 10, f-statistic)
	0.5	0.99	SVM (5, 2000, 5000, 10, RFE)
RLI	0.0	1.0	KNN (17, 1250, 4000, 10, f-statistic)
	0.01	1.0	KNN (17, 1250, 4000, 10, f-statistic)
	0.1	1.0	KNN (17, 1250, 4000, 10, f-statistic)
	0.25	1.0	KNN (15, 1000, 4000, 10, f-statistic)
	0.5	1.0	KNN (10, 1250, 1000, 10, RFE)
Loss	0.0	0.4	DT (5, 500, 5000, 10, RFE)
	0.01	0.8	DT (5, 1500, 5000, 10, RFE)
	0.1	1.0	KNN (10, 1000, 2000, 10, RFE)
	0.25	1.0	KNN (5, 250, 5000, 10, f-statistic)
	0.5	1.0	KNN (12, 1000, 4000, 10, f-statistic)
LatI	0.0	0.4	DT (10, 250, 4000, 10, RFE)
	0.01	0.73	DT (7, 250, 5000, 10, RFE)
	0.1	0.96	DT (10, 750, 5000, 25, RFE)
	0.25	1.0	KNN (10, 1000, 4000, 10, RFE)
	0.5	1.0	KNN (10, 1000, 3000, 25, RFE)

Table 3.7: Summary of the best Classification (Class) rate given a maximum False Positive (FP) rate. The first column is the Instability (I) definition. The second and third columns indicate the FP and Class rate respectively. The ML Technique description is (name, history size (obs), gap size (ms), horizon size (ms), boosting amount (%)).

occurs when a condition for link instability is satisfied. An unstable event may be used in multiple training examples depending on the size of the horizon window. For example, a packet arrival rate of 90% may occur at the beginning or at the end of a 3000 *ms* horizon window, and this event may be used in multiple examples. Therefore, we measure the success of a ML technique based on its ability to classify an unstable event before the unstable event reaches the gap period. The unstable event classification rate is defined as $\frac{TotalUnstableEventsClassified}{TotalNumUnstableEvents}$.

At times, a ML technique may predict link instability incorrectly, resulting in a false

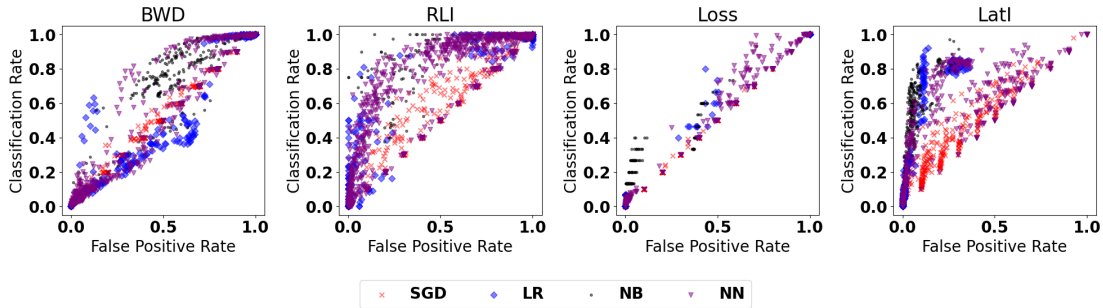


Figure 3.18: False Positive Rate v. Classification Rate for SGD, LR, NB, and NN

positive. The false positive rate is defined as the number of times the technique incorrectly predicts instability over the total number of stable classifications in our testing set. Ideally, a ML technique has a low false positive rate because an application may unnecessarily reduce its sending rate due, resulting in network under-utilization.

We evaluate our various ML techniques using examples that were constructed as specified in Section 3.2.3. Our experiments comprise different combinations and subsets of the history window size (5, 7, 10, 12, 15, 17, or 20 observations), gap size (250, 500, 750, 1000, 1250, 1500, 1750, or 2000 *ms*), horizon window size (1000, 2000, 3000, 4000, or 5000 *ms*), and boosting levels (10%, 25%, 50%, or None). The boosting level indicates the percentage of our training examples that have link instability because the majority of our training examples are classified as stable. Furthermore, we investigate the classification rate when we use the top five attributes as determined by f-statistic and by RFE ranking for each definition of instability. These attributes include Instantaneous Received Bitrate (Recv BPS) (avg, std, slope, r-value), Packet Delivery % (avg, std), RSSI (avg, std), Mean Latency (slope), and Jitter (avg, std, r-value) over all link instability definitions.

ML Tools to Use

We first explore the F1 and the accuracy score of our different ML techniques for different link instability definitions in Figure 3.16. The bar graphs in Figure 3.16 indicate the average and standard deviation across all created models using combinations of different history, gap, horizon and boosting level sizes. We find that for most cases, DTs have the highest F1 and accuracy score for our constructed examples where as SVMs tend to have lower F1 and accuracy scores. We find that KNN has higher accuracy for prediction Loss, LatI, and BWD than for RLI.

	Num Obs	5		10		15		20	
I Def.	Tech.	CR	FP	CR	FP	CR	FP	CR	FP
BWD	DT	0.77 (0.19)	0.48 (0.17)	0.84 (0.16)	0.54 (0.12)	0.79 (0.16)	0.5 (0.12)	0.82 (0.14)	0.56 (0.14)
	KNN	0.87 (0.0)	0.52 (0.0)	0.84 (0.01)	0.51 (0.01)	0.98 (0.01)	0.74 (0.01)	0.99 (0.0)	0.74 (0.01)
	SVM	0.66 (0.21)	0.44 (0.14)	0.41 (0.27)	0.31 (0.16)	0.56 (0.08)	0.42 (0.19)	0.52 (0.08)	0.47 (0.22)
Loss	DT	0.44 (0.28)	0.03 (0.02)	0.32 (0.26)	0.03 (0.02)	0.32 (0.26)	0.03 (0.02)	0.35 (0.15)	0.03 (0.02)
	KNN	0.8 (0.07)	0.06 (0.0)	0.9 (0.1)	0.06 (0.0)	0.83 (0.17)	0.06 (0.0)	0.77 (0.1)	0.06 (0.0)
	SVM	0.8 (0.27)	0.78 (0.26)	0.69 (0.32)	0.56 (0.38)	0.69 (0.27)	0.63 (0.28)	0.37 (0.18)	0.3 (0.24)
LatI	DT	0.72 (0.22)	0.09 (0.04)	0.79 (0.1)	0.09 (0.03)	0.78 (0.1)	0.12 (0.06)	0.73 (0.2)	0.1 (0.05)
	KNN	0.76 (0.23)	0.23 (0.03)	0.73 (0.26)	0.22 (0.02)	0.69 (0.27)	0.22 (0.02)	0.7 (0.28)	0.22 (0.02)
	SVM	0.59 (0.3)	0.54 (0.37)	0.74 (0.18)	0.68 (0.27)	0.7 (0.32)	0.6 (0.38)	0.74 (0.04)	0.67 (0.35)

Table 3.8: Impact of Window Size on Classification Rate (CR) and False Positive (FP) Rate for different instability definitions and ML techniques (Tech.). This table presents the Average (Standard Deviation) over all configurations of the specified ML technique. Gap window (1000 *ms*), horizon window (2000 *ms*), and boosting level are fixed (25%).

	Gap (<i>ms</i>)	250		500		1000		2000	
I Def.	Tech.	CR	FP	CR	FP	CR	FP	CR	FP
BWD	DT	0.82 (0.2)	0.48 (0.16)	0.81 (0.2)	0.47 (0.17)	0.8 (0.2)	0.48 (0.17)	0.87 (0.1)	0.6 (0.07)
	KNN	0.88 (0.0)	0.5 (0.01)	0.88 (0.0)	0.51 (0.01)	0.85 (0.0)	0.52 (0.01)	0.85 (0.01)	0.53 (0.0)
	SVM	0.58 (0.27)	0.61 (0.23)	0.82 (0.15)	0.51 (0.26)	0.59 (0.11)	0.48 (0.13)	0.78 (0.22)	0.7 (0.3)
Loss	DT	0.35 (0.24)	0.04 (0.02)	0.33 (0.22)	0.03 (0.01)	0.43 (0.32)	0.03 (0.02)	0.42 (0.27)	0.04 (0.01)
	KNN	0.8 (0.07)	0.06 (0.0)	0.73 (0.0)	0.06 (0.0)	0.83 (0.1)	0.06 (0.0)	0.83 (0.1)	0.06 (0.0)
	SVM	0.75 (0.28)	0.73 (0.28)	0.69 (0.27)	0.58 (0.33)	0.82 (0.2)	0.81 (0.18)	0.47 (0.29)	0.4 (0.33)
LatI	DT	0.78 (0.15)	0.11 (0.06)	0.68 (0.16)	0.07 (0.03)	0.81 (0.1)	0.09 (0.05)	0.62 (0.2)	0.07 (0.03)
	KNN	0.74 (0.25)	0.23 (0.03)	0.75 (0.24)	0.23 (0.03)	0.75 (0.23)	0.22 (0.03)	0.72 (0.26)	0.21 (0.02)
	SVM	0.68 (0.07)	0.47 (0.36)	0.42 (0.29)	0.58 (0.4)	0.43 (0.29)	0.62 (0.31)	0.57 (0.3)	0.48 (0.36)

Table 3.9: Impact of Gap Size on Classification Rate (CR) and False Positive (FP) Rate for different failure definitions and ML techniques (Tech.). This table presents the Average (Standard Deviation) over all configurations of the specified ML technique. History window (5 observations), horizon window (2000 *ms*) and boosting level are fixed (25%).

We further explore each ML technique’s ability to identify link instability for a given maximum false positive rate as different applications may have different tolerances for unnecessarily reducing the available bandwidth. The results are listed in Table 3.7, where each row specifies the maximum acceptable false positive rate and link instability definition. The *Class Rate* column indicates the maximum classification rate to two significant digits. In general, a tolerance for a higher false positive rate results in a higher classification rate.

Furthermore, Table 3.7 shows that using RFE for attribute selection results in higher classification rates given a maximum false positive rate. We find that DTs can achieve low false positive rates for different definitions of link instability. We also find that KNNs can achieve a high classification rate if the user is willing to tolerate a higher false positive rate.

False Positive Rate v. Classification Rate

The results from Table 3.7 suggest that there is a relationship between false positive rate and classification rates for link instability. In Figure 3.17, we plot the classification rate v. the false positive rate for various experiments, which vary across the different window sizes and boost percentage, for DT, KNN, and SVM as they were able to achieve the highest classification rate given a maximum false positive rate, and we plot the remaining ML techniques in Figure 3.18

These results show that many ML techniques can detect RLI; however, they may also have a high false positive rate. Although a single setting of KNN can detect RLI (Table 3.7), there are many models that have a high false positive rate, leading to lower accuracy (Figure 3.16). With regards to BWD and LatI, Figures 3.17 and 3.18 suggest that there may be a linear relationship between the false positive and classification rates. Finally, these figures suggest that Loss is difficult to detect for SGD, LR, NB, and NN since the false positive and classification rate are equal to each other. To detect Loss, Figure 3.17 suggests the use of a DT or a KNN.

We find that there is a linear relationship between false positive and classification rate for stochastic gradient descent in many of the cases. We find that in some cases such as predicting latency or loss, Support Vector Machines are unable to classify link instability or have a high classification rate with a high false positive rate. Finally, the trends in Figure 3.17 suggest that KNN is able to predict bandwidth drops whereas DTs are better at predicting increases in latency, and LR has higher classification rates for loss as a whole. We find that the classification rate and false positive rate for NNs and SVMs greatly vary. We evaluated different activation functions for NNs and different polynomial degrees for SVMs. Although these techniques may benefit from further tuning, such as learning rates and iterations, we wanted to maintain the same amount of time spent for tuning each ML technique. A simpler technique, such as a DT, may offer higher classification rates and lower false positive rates than NNs and SVMs without additional tuning time.

Window Size Analysis

We now determine how window sizes (history and gap) affect the predictive ability of our ML Techniques. Analyzing the impact of the history window size allows us to determine the window size that best captures trends while not being susceptible to noise. Analyzing the gap size allows us to determine how far ahead we can predict in the future. In our first experiment, we vary the history window size (5, 7, 10, 12, 15, 17, 20 observations) and fix the gap and horizon windows to be 1000 *ms* and 2000 *ms* respectively. We present

the results for history window sizes of 5, 10, 15, and 20 in Table 3.8. Table 3.8 shows the impact of horizon window size on the false positive rate and the classification rate for the loss (Loss), changes in bandwidth (BWD), and drops in latency (we select to use LatI as both LatI and RLI detect changes in latency). We find that smaller history window sizes are better for predicting Loss and LatI whereas larger (20 observations) window sizes are better for predicting BWD.

Our next experiment investigates the impact that gap size has on the classification and false positive rate. We vary the gap size between (250, 500, 750, 1000, 1250, 1500, 1750, 2000 *ms*). We also fix the history window size to be 7 observations, the horizon window size to be 2000 *ms* and the boosting level to 25%. Table 3.10 shows the false positive and classification rate as we increase the gap size. For the BWD and LatI, we find that the predictive ability of our ML technique decreases as we increase the gap size; however, the trend is less clear with Loss.

Finally, we determine how often changing the size of the history window and gap window affects the resulting classification rate and false positive rate significantly. To accomplish this, we perform the paired student-t test between all combinations of history window sizes and gap window sizes respectively for the DT, KNN, and SVM techniques. Each observation is paired according to the aspects of the attribute that do not change (e.g., horizon window size, boosting level, and any ML technique setting). The results of this experiment are found in Table 3.10. For DTs predicting Loss instability, we find that different history window sizes significantly affect the classification rate 86% of the time and the false positive rate 57% as the p-value of the student-t test was less than $\alpha = 0.05$ threshold. We find that the differences in history and gap window sizes impact the classification and false positive rates less often for SVMs than for DTs and KNNs.

3.2.5 Discussion

The inputs to the machine learning techniques are the summary statistics (e.g., average, standard deviation, slope, etc.) for the network metrics (e.g., delay, available bandwidth, signal strength, etc.). The user must measure the input network metrics, track the measurements for a given window, and calculate the summary statistics to use as input to the machine learning technique.

The machine learning technique produces a classification or prediction of either stable or not stable, and the user may use this prediction to adapt or lower the video bitrate. These predictions can also be used in the multi-homed environment to decide when to use alternate paths.

		Obs. Size		Gap Size	
I Def.	Technique	CR	FP	CR	FP
BWD	DT	81%	100%	95%	100%
	KNN	100%	95%	67%	86%
	SVM	29%	38%	14%	48%
Loss	DT	86%	57%	90%	71%
	KNN	67%	90%	71%	100%
	SVM	33%	38%	19%	24%
LatI	DT	76%	81%	95%	100%
	KNN	95%	86%	67%	100%
	SVM	10%	67%	38%	43%

Table 3.10: Percentage of combinations that where different window size yielded significant differences in Classification (CR) and False Positive (FP) rates

3.2.6 Summary

In this section, we have analyzed commercial wireless LTE traces that were used for video streaming and exhibited significant unstable events. Using these traces, we created examples for training and testing ML models for link instability. Because link instability can manifest itself as packet loss, increases in latency, or drops in available bandwidth, our goal was to determine if we could predict link instability using various ML techniques. We found that RFE provided better attributes for ML model training than ANOVA’s f-statistics as ML models trained on these attributes have higher classification or prediction rates. Furthermore, our analysis and results showed that different ML techniques should be used based on the definition of instability. We found that DTs often provided the highest classification rate given a maximum false positive rate; however, KNNs should be used when predicting increases in latency.

Our work addresses many overlooked areas in related work including how to address unequal sampling or measurement intervals in terms of time in raw data. As an alternative, we use a fixed number of observations and capture the trends in various network metrics. By doing so, we capture different metrics with different inter-time sampling intervals. We found that using 5 to 10 observations in the history window provided the highest classification rates given a maximum false positive rate. Furthermore, we have considered statistical ML techniques and have found that KNNs work well in many cases. We have also

investigated other definitions of link instability instead of focusing on available bandwidth prediction. Lastly, we have introduced a gap window and have shown that we can obtain a prediction early enough for applications to adapt.

3.3 Chapter Summary

In this chapter, we have introduced techniques for link quality modelling. Our PacketBurst bandwidth measurement techniques removes inaccuracies caused by kernel interrupts for packet trains. When using six MTU-sized packets to estimate available bandwidth, the PacketBurst technique experiences an 2.75% error whereas packet trains experience a 21.6% error when the available bandwidth is 20Mbps.

We have also presented a method for link instability prediction using traces collected by an application developer. We analyzed traces and determined process metrics (e.g., mean, standard deviation, slope, and correlation values) to capture trends in network measurements such as latency, application-level packet loss, and available bandwidth. Using these traces, we investigated the use of different statistical ML techniques and their predictive ability to determine link instability.

Chapter 4

Conflux: A Multi-homed Adaptive Bitrate Protocol for On-Site Live Video Streaming

In this chapter, we present Conflux: a multi-homed adaptive bitrate protocol for on-site, live video streaming over wireless networks. Conflux is a modular protocol that encapsulates link quality characterization, packet scheduling, and video bitrate adaptation into different, independent modules. This approach simplifies the design of each module and enables Conflux to support different configurations (e.g., number of links) or user preferences with changes that are localized to a single module.

Our chapter contents are as follows: Section 4.1 introduces the challenges that Conflux addresses. We describe Conflux’s protocol design in Section 4.2. We present our evaluation methodology in Section 4.3 and evaluate Conflux and its comparison systems in Section 4.4.

4.1 Unique Challenges Found in On-Site Live Video Streaming

The primary focus of Conflux is to provide low-latency, video bitrate adaptation for on-site, live news casting in the multi-homed environment. The importance of this application is demonstrated by existing proprietary solutions [34, 50, 89, 140] that stream low delay (80ms - 1s), broadcast-quality video. This application’s challenges differ from those found

in on-demand video (Netflix), studio live streaming (Youtube Live, Twitch TV) and video conferencing (WebRTC, Skype). In this section, we describe these challenges.

Challenge #1 Use of Mobile Networks

Many commercial solutions use 4G/5G links to stream video because wired and private WiFi links are often unavailable (e.g., disaster or remote areas) or do not provide the end user with the necessary mobility. Streaming video over 4G/5G is challenging because these links are highly variable and can drop to near zero usable bandwidth in a short period of time [38, 97, 106]. This variability can be the result of poor signal quality due to presence of physical barriers, weather, and distance to cell towers, and the number of other users accessing the base station [8, 11, 14, 87]. Thus, the link's data rate is often below the provider's advertised rate.

To address data rate variability and poor signal strength, providers often employ large in-network buffers [58] and link-layer Automatic Repeat reQuests (ARQ) [11] to prevent application-level packet loss which can increase network latency [68]. Given that our expected deployments are at areas of interest (e.g., disaster zones or large community events such as sports and concerts), a single link may not have sufficiently high bandwidth and low latency for the duration of the video stream. It is challenging to consistently deliver a video stream in these environments without the occasional disruption that is unacceptable for live broadcasting.

Challenge #2 Scalability

A single 4G/5G link may not be able to sustain a low-latency, broadcast-quality video stream for its duration [38]. And as video bitrate demands continue to increase [119], even two links may be insufficient. Therefore, it is important that an adaptive, live, multi-homed video streaming protocol be scalable and adaptable as video bitrate requirements and networks evolve.

Challenge #3 Flexibility for Users

Users have different bandwidth usage budgets and video bitrate preferences depending on their viewership and the type of event that is being captured. It is challenging to construct a flexible protocol that can be easily adapted to different users without redesigning core components such as video bitrate adaptation or multi-path scheduling.

There are many dimensions, such as video bitrate, multi-path scheduling, and the use of redundancy, that need to be considered simultaneously when streaming adaptive, low-latency video in a multi-homed environment. It is important that a protocol captures the trade-offs of these decisions to address the aforementioned challenges.

4.2 Conflux Protocol

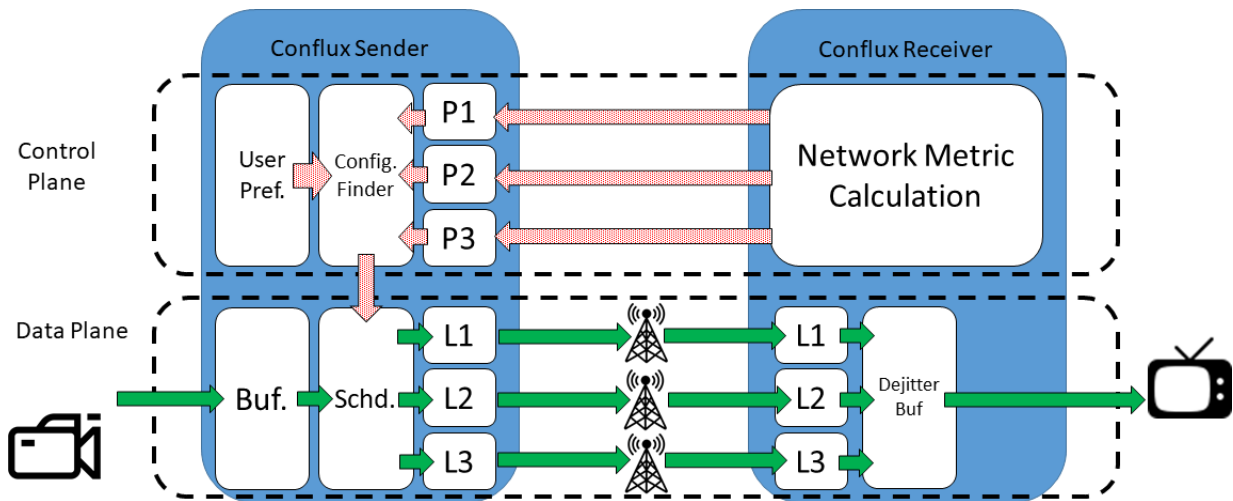


Figure 4.1: Overview of Conflux. Solid arrow depicts the flow of video data. Shaded arrow depicts the control information.

Designing a multi-homed, adaptive video bitrate protocol is challenging because it must address multi-homed transport and video bitrate adaptation. Conflux aims to solve both challenges. Figure 4.1 depicts Conflux’s design which comprises a Control Plane and a Data Plane and their respective modules. The Control Plane (Section 4.2.1) determines the sending rate on each link, the video bitrate, and the total sending rate which is affected by the amount of redundancy, and the Data Plane (Section 4.2.2) carries out the Control Plane’s decisions.

4.2.1 Conflux’s Control Plane

Conflux’s primary goal is to obtain the highest user QoE, which is affected by the video stream’s bitrate and its bandwidth usage cost. We can model the impact that these

attributes have on user QoE using a utility function that specifies the utility score, or reward, obtained by the video’s attributes. Given the user’s utility function, Conflux’s Control Plane maximises end-user QoE by determining the video streaming parameters that provide the highest expected utility. We characterize the challenge of maximising the user’s expected utility as a *decision making under uncertainty* problem.

Conflux represents the solution to this problem as a *configuration*, which is defined as the sending rate on each link and the video bitrate that is targeted by the video encoder. Conflux maximises end user’s QoE by finding the *best configuration*, which is the one that provides the *highest expected utility*. Determining the best configuration requires: (1) the probability that Conflux is able to achieve a video stream that does not stall or have visual artifacts, (2) a utility function that specifies a video stream’s reward based on its attributes such as video bitrate, and (3) the ability to search through different possible configurations to determine the one that provides the highest expected utility.

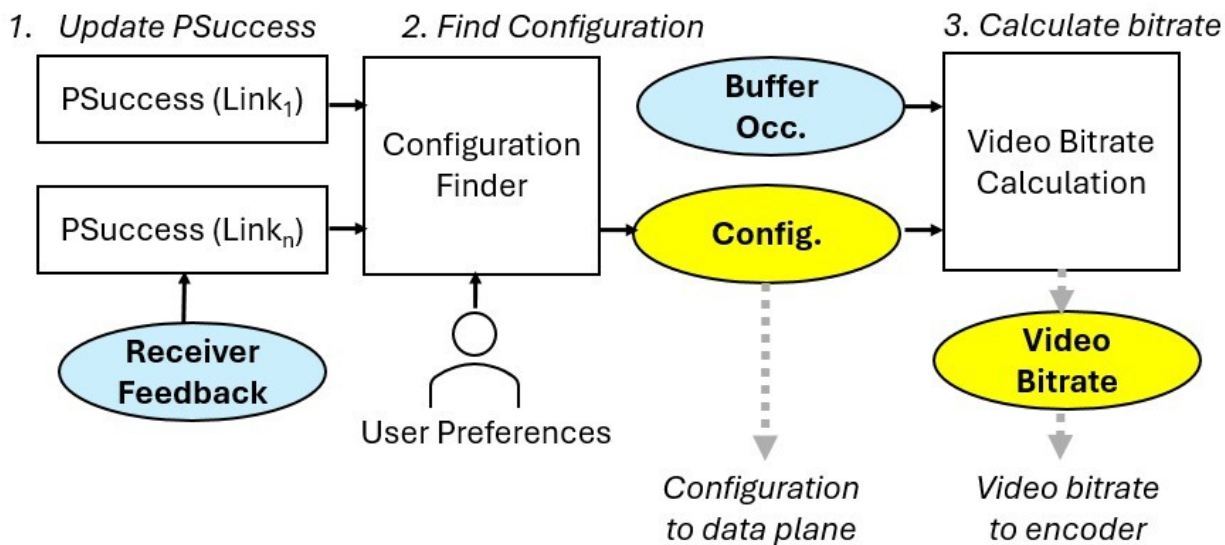


Figure 4.2: Overview of Conflux’s Control Plane

Conflux comprises different modules where each module addresses one of the aforementioned requirements. Figure 4.2 shows an overview of the control plane and how information flows between the different Conflux modules. We describe the Probability of Success, User Preferences, and Configuration Finder modules in turn.

Probability of Success

Constant	Description
α / α'	Constant to control the weight of the most recent good observation ($\alpha = 10000$, $\alpha' = 100$)
β / β'	Constant to control the weight of the most recent bad observation ($\beta = 10000$, $\beta' = 100$)
γ	Specifies the threshold of affected sending rates that incur β penalty when the observation is bad ($\gamma = 0.5$)
p_{good}	The threshold sending rate of affected buckets when the observation is good ($p_{good} = \min(1Mbps, SR \times 0.05 + 0.2Mbps)$)
p_{bad}	The fraction of sending rates (and their respective buckets) affected below the sending rate when the observation is bad ($p_{bad} = 0.9$)
r_{good}	The reduction factor applied to the weight of the observation for buckets above the sending rate when the observation is good ($r_{good} = 0.75$)
r_{bad}	The reduction factor applied to the weight of the observation for buckets below the sending rate when the observation is bad ($r_{bad} = 0.75$)

Table 4.1: Algorithm 1 Constants

Conflux’s *Probability of Success* module models link quality as a *PSuccess function* that maps an input sending rate to an estimate of the probability that all data is delivered at that rate by the user-specified deadline. As links are heterogeneous, each link has its own PSuccess function. Using the links’ PSuccess functions, Conflux estimates the video stream’s probability of on-time arrival. This probability is multiplied by the user’s utility score (Section 4.2.1). In this section, we describe our method of constructing the PSuccess function that uses latency measurements to detect the presence of queuing delay and congestion. We use a rolling window of these measurements to estimate the link’s probability of on-time data delivery for different sending rates.

We efficiently construct and update our PSuccess function using real network measurements to obtain an accurate model that reflects the network’s current condition. To accomplish this, we group similar bitrates together so that there are fewer rates to measure. We define a *bucket* to be a *bitrate range* that is bounded by a minimum and maximum. Each range is selected based on the number of fixed-sized packets used to send a video

Algorithm 1 UpdatePSuccess(feedback, sendRate)

```
1: if feedback.latency  $\geq$  acceptable_delay then
2:   for All Buckets B do
3:     B.Psuccess = 0.0
4:   end for
5: else if feedback.isGood == True then
6:   for All Buckets B below sendRate do
7:      $B.PSuccess = \frac{(100 \times B.PSuccess) + \alpha}{100 + \alpha}$ ;
8:   end for
9:    $w = \alpha'$ 
10:  for All Buckets B s.t.  $sendRate \leq B.max\_rate \leq p_{good} + sendRate$  do
11:     $B.PSuccess = \frac{100 \times B.PSuccess + w}{100 + w}$ ;
12:     $w = w \times r_{good}$ 
13:  end for
14: else if feedback.isGood == False then
15:  for All Buckets B above  $\gamma \times sendRate$  do
16:     $B.PSuccess = \frac{100 \times B.PSuccess}{100 + \beta}$ ;
17:  end for
18:   $w = \beta'$ 
19:  for All Buckets B such as  $\gamma \times sendRate \times p_{bad} \leq B.max\_rate \leq \gamma \times sendRate$  do
20:     $B.PSuccess = \frac{100 \times B.PSuccess}{100 + w}$ ;
21:     $w = w \times r_{bad}$ 
22:  end for
23: end if
```

frame where the last packet may be smaller than the fixed size. Equation 4.1 shows this calculation using the Video Bitrate (VBR), Frame Rate (FR), and Packet Size (PS). When using 512-bytes packets and a frame rate of 25 *fps*, an additional packet per video frame results in a VBR increase of 102.4*kbps* (Equation 4.2). Therefore, we use 102.4*kbps* as our bucket size.

$$NumPackets = \frac{VBR}{1/FR} \times \frac{1}{PS} \quad (4.1)$$

$$BucketSize = \frac{PS}{1/FR} \quad (4.2)$$

Each bucket's PSuccess value estimates the link's ability to deliver data at its bitrate.

This value is calculated using a rolling window of feedback measurements (i.e., one-way delay) that identify the presence of queuing delay and congestion. Using the insight proposed by COPA [13], a link’s condition is *good* when it does not exhibit an increase in queuing delay that is the result of congestion. This increase is identified when the minimum measured latency for a recent window (up to one second) of time is greater than the minimum measured latency for a longer window (ten seconds) of time. We say that the link’s condition is *good* and can support the sending rate when there is no increase in queuing delay. Otherwise, the link’s condition is *bad*.

Algorithm 1 updates the PSuccess function as feedback measurements arrive. Table 4.1 specifies Algorithm 1’s parameters and their values in our implementation, which were determined experimentally. Lines 1-5 show that if the link’s latency is above the user’s acceptable delay, the PSuccess function returns 0.0 since data cannot arrive by its deadline. When feedback indicates that the link’s latency is lower than the user’s acceptable delay, the PSuccess values are updated as a weighted rolling average. Lines 6-13 show that all buckets that are less than or equal to *SendingRate* update their PSuccess values by applying α to the weighted rolling average formula (Line 8) when feedback indicates that the link is good. Buckets that are between *SendingRate* and $p_{good} + \textit{SendingRate}$ are also updated using a lower weight (α') because the link may be able to support these rates. Lines 14-23 show that buckets that are greater than or equal to $\gamma \times \textit{SendingRate}$ (Line 18) reduce their PSuccess values using β when the link is bad. Furthermore, buckets between $p_{bad} \times \gamma \times \textit{SendingRate}$ and $\gamma \times \textit{SendingRate}$ update their PSuccess values using a lower weight (β'). Conflux also detects when feedback is not received within $4 \times \text{RTT}$, which indicates congestion as packets have not arrived to generate feedback. In this case, Conflux updates the link’s PSuccess function as though it was bad using the unacknowledged packets’ sending rate. This allows Conflux to ramp down the link’s sending rate quickly to avoid congestion.

The Conflux sender is designed to be deployed on a dedicated device (i.e., a video camera) that streams video over LTE. Because of these deployment expectations, it is not necessary for Conflux to be fair to other network flows on the same device. Furthermore, we do not need to consider fairness among different LTE users that share the same LTE spectrum since the eNodeB (or cell tower) schedules resources across users [6].

The use of each link may impact the decisions of other users, and this may result in changes in the amount of available bandwidth. As the LTE scheduler is responsible for dividing resources among all users, we do not model the impact of other users’ decisions in the PSuccess function. This is because we expect that the impact of other users’ decisions to be small compared to other factors such as physical barriers and the total number of users accessing the cell tower; thus, we leave this challenge for future work.

Building the PSuccess Function

The PSuccess function is built using real network measurements to accurately reflect current link conditions. This method of constructing PSuccess is challenging when (1) there is no link information available at the beginning of the video stream and (2) determining the PSuccess of rates that are higher than what the user has chosen to explore. Conflux probes links by sending data at unexplored bitrates, which are typically higher than the user-selected streaming rates, to address both challenges.

Selecting the appropriate probing strategy is important because we do not want probing to cause self-inflicted congestion which can lead to lower video quality [145]. In our design, Conflux probes links one at a time in a round-robin fashion so that alternate links remain available to resend data if the probed link becomes congested. Furthermore, we use redundant data (Section 4.2.1) for probing to reduce the risk of delayed data. When there is no link information available, we initially set the probing rate to be $102.4Kbps$, which is our bucket size (Section 4.2.1). We double our probing rate as positive feedback is received so that we can quickly explore high bitrates. However, once our probing mechanism discovers an upper limit to the link’s available bandwidth which is indicated by bad feedback, we probe at the lowest sending rate that normally has a PSuccess value of 0. This rate is either the lowest rate that we have not explored or is $102.4Kbps$ above the previously measured available bandwidth. Probing packets are sent after video data packets so that video data might arrive on time even if the probing rate is greater than the link’s available bandwidth. This probing mechanism ensures that we do not continue to cause significant amounts of self-inflicted congestion.

User Preferences

In addition to the probability of on-time video frame arrival, maximizing the user’s expected utility also requires the user’s utility score, which represents the user’s QoE. Conflux users specify their QoE model as a *utility function* in the *User Preferences* module. The utility function uses metrics that affect user QoE to calculate the user’s utility score. These metrics include video bitrate and the monetary cost of sending data over LTE. We select these metrics because video bitrate affects video quality and user QoE [113] and because users may have limited budget for using LTE [157]. The User Preferences module allows users to adjust their trade-offs.

Other video quality metrics directly capture video distortion and user QoE. However, collecting these metrics may not be practical for live, remote video streaming due to time

and energy constraints. Examples of these metrics include the Structural Similarity Index (SSIM), Peak Signal-to-Noise Ratio (PSNR), Video Multimethod Assessment Fusion (VMAF), and Mean Opinion Score (MOS). Accurately obtaining and comparing video quality metrics such as SSIM, PSNR, and VMAF requires additional hardware for encoding multiple copies of the video stream at different bitrates to compare the encoded video with the reference video. Unfortunately, this is infeasible given the power consumption budget of mobile devices [56]. Furthermore, measuring metrics such as MOS requires post-streaming analysis [78] which does not meet the delay requirements for live streaming.

Conflux uses user-specific utility functions to better adapt to different users since the relationship between video quality metrics and user QoE is often dependent on the type of video and user [170]. In our design, the utility function’s specified score is awarded if the video frame arrives on time. For simplicity, we treat all video frames equally when determining their expected utility score. These utility functions are used as inputs to the Configuration Finder (Section 4.2.1) which determines the Conflux-selected video bitrate (abbreviated as $C.VBR$) and the total sending rate (abbreviated as $C.SR$). We provide two example utility functions that address different user preferences.

Aggressive Equation 4.3 specifies a utility function that is designed for users, such as sports viewers, who have high video bitrate requirements and some tolerance for delayed video data [96]. Given these preferences, this utility function rewards high video bitrate according to the sigmoidal relationship between video bitrate and user QoE [55, 129]. Furthermore, it rewards significantly less utility to lower sending rates that are used to avoid congestion. To model the sigmoidal relationship between video bitrate and user QoE, Equation 4.3 specifies an exponential component to encourage the Configuration Finder (Section 4.2.1) to select higher bitrates that have lower PSuccess values than lower bitrates. Equation 4.3 caps the utility gain at $MaxVBR$, which is the user’s maximum desired video bitrate, to avoid streaming at extremely high bitrates where the returns on user QoE are not appreciable.

$$U(C) = \begin{cases} b^{C.VBR} & VBR \leq MaxVBR \\ b^{MaxVBR} & Otherwise \end{cases} \quad (4.3)$$

Conservative Equation 4.4 specifies a utility function for users who may prefer no interruptions and are willing to have a lower video bitrate [164]. We use our PSuccess models to identify the highest rate that we are confident that the link can support; this rate has a 1.0 PSuccess value and is referred to as $BW(l)$. This user preference limits the sending

rate on each link, which is referred to as $SR(l)$, to be less than the user-selected fractional threshold t of $BW(l)$. For example, if the highest rate with a PSuccess of 1.0 on link l is 1 Mbps, and $t = 0.5$, then the $SR(l)$ is limited to be 0.5 Mbps.

As shown in Equation 4.5, positive utility is given for each link's SR that is below $t \times BW(l)$. If a link's sending rate exceeds $t \times BW(l)$, then the utility score for that link is $-SR(l)$ to heavily discourage sending at rates that are greater than this threshold. Using this utility function reduces the potential for congestion as it encourages links to send at rates which we are confident that the links can support.

$$U(C) = \left(\sum_{l \in Links} Score(l) \right) \quad (4.4)$$

$$Score(l) = \begin{cases} SR(l) & SR(l) \leq t \times BW(l) \\ -SR(l) & Otherwise \end{cases} \quad (4.5)$$

Configuration Finder

The *Configuration Finder* module determines the *configuration*, which is defined as the sending rate on each link and the video bitrate that is targeted by the video encoder, that provides the highest expected utility. To calculate the expected utility, we must determine the Probability that the Video Frame Arrives (*PVideoFrameArrival*) on time since utility is gained only if the video frame can be reconstructed by its deadline. *PVideoFrameArrival* is calculated using the links' PSuccess functions and is the product of the links' PSuccess at their sending rates since all sent data must be received on time to reconstruct the video frame by its deadline.

The Configuration Finder searches through and evaluates configurations with different sending rates to determine the one that has the highest expected utility. There is a short time budget to accomplish this because the video encoder must be informed of the updated video bitrate to adapt the size of its generated frames to the changing network conditions. Although exhaustive algorithms may find the configuration with the maximum expected utility, they may require a significant execution time and cause the video encoder to continue to generate frames that either cause self-inflicted congestion when sent or do not fully utilize the available bandwidth. Therefore, we aim to find a new configuration within 20ms since we expect video frames to arrive every 40ms to allow the video encoder to adjust its bitrate. Furthermore, we budget this amount of time because the LTE schedules packets to be sent at this time interval [6].

To address the challenge of meeting the time budget for selecting the updated configuration, we reduce our search space by conducting a coarse-grained search through possible sending rates on each link. Conflux uses the configuration from this subset with the highest expected utility.

Determining the best configuration by analyzing the links' PSuccess functions is not possible when link quality information is unavailable or when all links are unable to deliver data on time. In these scenarios, the Configuration Finder uses a user-provided, default configuration. In our implementation, Conflux selects a low video bitrate and replicates video data across all links to ensure on-time video data arrival, to collect link quality information, and to prevent over-sending that can cause congestion for future video data.

4.2.2 Conflux's Data Plane

The Data Plane partitions the video data into packets, creates parity packets, and sends all packets on the available links according to the configuration. It consists of the sender-side buffer, the Packet Assignment Module, and the dejitter buffer. Figure 4.3 shows an overview of Conflux's data plane and the information exchanged with the control plane.

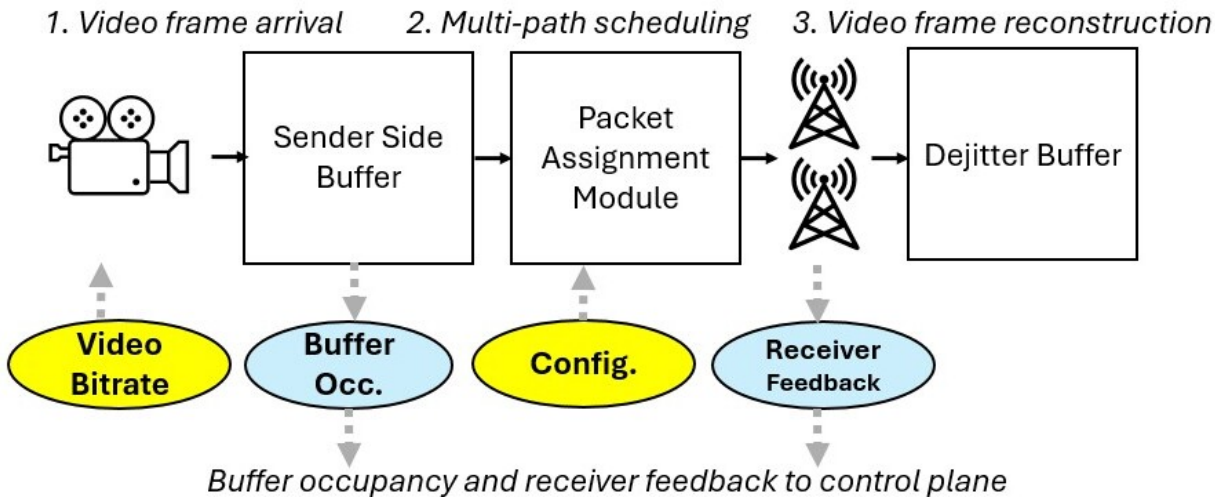


Figure 4.3: Overview of Conflux's Data Plane.

Sender-side Buffer

Conflux’s sender-side buffer holds the video data prior to sending to ensure that the links do not send more data than the configuration-specified amount. This is necessary when the video encoder generates a video frame that is larger than expected as sending this frame immediately can cause self-inflicted congestion. Unfortunately, retaining this video frame increases delay and may cause video data to miss its deadline. The sender-side buffer’s occupancy is reported to the control plane which lowers the video bitrate that is reported to the encoder. This lower rate is calculated as $\frac{VBR}{NumFramesInBuffer}$. There is also an additional 5% reduction for each consecutive instance that the sender-side buffer has more than one video frame. By reporting a lower video bitrate, Conflux drains the sender-side buffer and prevents late video data.

The Packet Assignment Module

The Packet Assignment Module (PAM) sends the sender-side buffer module’s data every 20 ms. We select this interval because it is lower than our frame inter-arrival time so that a frame is sent before the next frame arrives. Furthermore, it is a common LTE uplink scheduling period [107]. At the start of this 20 ms cycle, the PAM uses the configuration to determine the total number of packets (*TotalNum*) it can send during its current cycle. The PAM partitions the video data into packets and sends them according to the configuration.

Dejitter Buffer

As video packets arrive from the sender, the dejitter buffer at the receiver reconstructs the video stream from the packets. It reorders the video data according to the global sequence number and returns data in order by its deadline, skipping over any data that has not arrived on time. This avoids the head-of-line blocking for on-time video frames that are sent after the delayed frame, which can occur in protocols such as MPTCP and MPQUIC due to their in-order arrival guarantees.

4.3 Methodology

We evaluate Conflux by streaming video over various trace-driven, emulated network environments. In this section, we describe the traces that specify the available bandwidth of

our wireless links, our comparison systems, liveness requirements, and encoding settings.

4.3.1 Traces

We select Mininet [100] as our network emulation tool as it is commonly used within the networking community to evaluate networking and streaming systems. Our application streams video across two nodes on a Mininet network. Each node in the Mininet network has multiple network interfaces representing multiple links. We perform a trace-driven emulation that follows the methodology presented in Mao et al. [94] where a link’s available bandwidth changes once every five seconds. We use traces from publicly available datasets including Riiser et al. [118], Winstein et al. [145], and Mei et al. [97], and we describe them in turn.

Riiser et al. [118] measures the throughput achieved at the application layer when streaming video over 3G networks in a non-stationary environment that includes walking and commuting on various types of public transportation. As these traces provide the bandwidth measurement’s timestamp, the available bandwidth for this workload changes according to the time of bandwidth measurement. We select these traces because they have recently been used to evaluate many other systems such as [94, 172, 165] and because they exhibit low available bandwidth, up to 4 *Mbps*, which is characteristic of a challenging network environment for video streaming.

Winstein et al. [145] collects bandwidth measurements by saturating a link between the sender and the receiver and are recorded using the Mahimahi record-and-replay tool [106] that captures the arrival time of each packet. We use the traces that are found in the Mahimahi deployment. Winstein et al. collected 17 minutes of data by driving around the Boston area during rush hour. This measured available bandwidth is up to 38 *Mbps*.

Mei et al. [97] measures available bandwidth using TCP iperf, and the server logs TCP throughput. These measurements were collected at different times of day, and the duration of each trace ranges from 10000 *s* to 20000 *s*. These traces were collected while commuting on six different transportation routes, totaling seven traces where one route was repeated, in New York City. The maximum available bandwidth measured in this trace is 25 *Mbps*.

The aforementioned traces do not provide fine-grained delay measurements. As our evaluation setup requires link delay to be specified, we use findings from [15, 48, 59, 145] to determine our default link delay. Winstein et al. [145] observed the one-way delay in their measurements to be 20 *ms*. Other works have found their respective sender-to-receiver delays to be between 15-50 *ms*. The range in one-way delays between the different works is a result of different measurement environments and paths between the sender and

receiver. We use a 20 *ms* one-way delay to emulate a wireless network link that is not experiencing signal or data loss following the findings of [48, 145].

Finally, these traces also lack packet loss information. Works such as [15, 59] have found applications seldom experience lost packets because of underlying link-layer retransmission (ARQ) mechanisms and large buffers to conceal packet loss. Instead, packet loss is manifested at the application level as a temporary increase in latency. We emulate packet loss by periodically increasing latency temporarily, and this increase is specified for each experiment.

4.3.2 Metrics

To evaluate Conflux and its comparison systems' ability to stream live video, we primarily use the video stream's quality which is captured by the following metrics:

- **Peak-Signal-to-Noise Ratio (PSNR)**: A common metric to evaluate video quality [127, 147, 148, 156].
- **Video Multi-Method Assessment Fusion (VMAF)**: Developed by Netflix [105], it is a score between 0 to 100 where 100 indicates that the produced video is identical to the reference video.

We use the official library provided by Netflix to measure the VMAF and PSNR of Conflux and its competitors. We have observed that missing a complete video frame results in significant penalties for our metrics due to frame alignment de-synchronization [4]. This penalty affects Conflux and the comparison systems that are subject to delayed video data.

4.3.3 Videos

Our application streams several reference videos captured at 1080p at 25fps which are described in Table 4.2. We select videos that are live and have interesting events. These videos also have different lengths and profiles to investigate the video quality that Conflux and its comparison systems can obtain when streaming over a highly variable wireless network for different amounts of time. Figure 4.4 shows the relationship between video bitrate and VMAF quality and illustrates the obtainable video quality improvements as video bitrate increases.

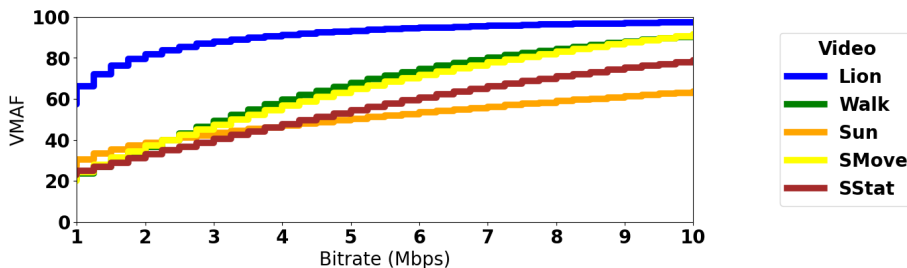


Figure 4.4: Video Bitrate v. Reference video VMAF (No Loss)

Video	Time (s)	Description
Lion	47	Video with slower motion
Sun	60	Panning over a landscape
SMove	60	High-motion during an extreme weather event
SStat	60	Stationary during an extreme weather event
Walk	120	Video with moderate motion

Table 4.2: Test Video Description

The lion [141] clip was gathered online; thus, it has been compressed so that it requires lower bitrate to retain its original quality. The remaining videos were captured on our device (iPhone 12), which is more characteristic of videos used in our expected deployment since they have not been edited or post-processed. Our evaluation primarily uses the Walk video.

4.3.4 Video Streaming Settings

A live video stream must meet the user’s latency requirements for the video stream to remain relevant to the user and for the user to have high QoE [91]. We refer to the user’s latency requirements as the *acceptable delay* which is the time from video capture to arrival at the receiver. Unless otherwise stated, our experiments use an acceptable delay of 500 *ms* because it is a widely used and accepted threshold for live video streaming [31, 10, 127]. Works such as [30] have found that 500ms delays result in humans having *fair* perception. We refer to *late video data* as video data that arrives past the user’s acceptable delay. This data is dropped from the video stream as the filmed event has passed. Incurring late data causes *incomplete frames* as the frames that are played are missing some of their data.

Following widely accepted industry recommendations to ensure that our evaluation is

representative of a typical deployment, we encode these videos using Adaptive BitRate (ABR) encoding with intra-refresh mode and the key frame interval set to the maximum value [2, 63].

4.3.5 Comparison Systems

Multi-homed, adaptive video bitrate protocols must schedule packets over multiple links and adapt the video bitrate as their links' conditions change. We compare Conflux with systems that use an oracle which provides the available bandwidth; this information is used to adapt the video bitrate. The quality of the resulting video stream is used to evaluate multi-homed transport protocols and schemes.

FRA-JSCC + Available Bandwidth Oracle (FRA and FRA-OPT)

FRA-JSCC performs joint-source channel coding and FEC. FRA-JSCC estimates bandwidth using PathChirp [117] to determine the video bitrate and sending rate on each link. We remove PathChirp and provide FRA-JSCC with each link's available bandwidth as it allows us to isolate and evaluate FRA-JSCC's core algorithm, which is video bitrate adaptation and packet scheduling. We evaluate two versions of FRA-JSCC. The first version (FRA) streams video over the network; thus, the resulting video may be missing data due to deadline violations. The second version's (FRA-OPT) video is obtained by encoding at the requested target bitrate; thus, it does not incur any deadline violations. We include this version because FRA-JSCC may experience deadline violations due to the encoder's limitations. The encoder may generate more data than expected, which we discuss in Section 4.5, and FRA-JSCC may temporarily send more data than the links' available bandwidth since its design does not include a sender-side buffer. Furthermore, in our evaluation of FRA-JSCC, we found that the assigned weights for their presented weighted round-robin scheduler may oversend data on a link. This causes congestion that results in video deadline violations. We remove this limitation in FRA-OPT; thus, its video quality results are FRA-JSCC's theoretical maximum.

MPTCP + Oracle Video Bitrate Adaptation (MPTCP)

We construct a live video streaming comparison system called *MPTCP Oracle* that uses MPTCP for multi-homed video data delivery. We select BLEST [45] as MPTCP's scheduler since it is designed for latency-sensitive workloads and is found to be the state of the

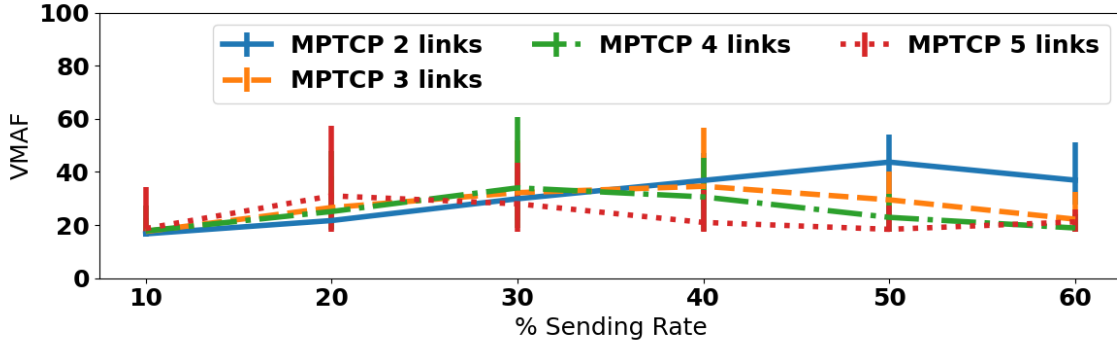


Figure 4.5: VMAF vs. %-tage Aggregate Available Bandwidth

art [92, 137]. MPTCP is a transport protocol and does not predict the available bandwidth. However, available bandwidth information is needed to adapt the video stream’s bitrate to changing network conditions. Therefore, this system uses an oracle that has knowledge of the total aggregate available to perform video bitrate adaptation.

We cannot directly use the total aggregate available bandwidth as the video bitrate as MPTCP may make scheduling decisions that result in late data since it does not know the available bandwidth on each path. We found that we can achieve higher video quality if the MPTCP Oracle streams at a fraction of the total aggregate available bandwidth. Figure 4.5 shows MPTCP Oracle’s video quality as the fraction of the total aggregate available bandwidth used for video streaming increases from 10% to 60%. Each line in Figure 4.5 shows the obtained video quality for a given number of links, which we refer to as a setting. Each setting is evaluated 150 times using our aforementioned traces.

In our experiments, MPTCP experiences head-of-line blocking, which increases the amount of delayed data and lowers video quality as a result. We find that as the sending rate approaches above 50% of the total available bandwidth, the five-link deployment experiences a significant increase in its 95%-tile delay (600ms). In this setting, MPTCP required at least four seconds to achieve excellent user perception. Thus, we select the streaming percentage, which is 40%, that can achieve our acceptable delay while not forgoing additional video quality.

Self-Clocking Rate Adaptation (Scream)

Scream [74] is a single-homed congestion control protocol that provides a target bitrate. Scream does not offer multi-homing for a single stream over multiple network interfaces,

and it does not specify a multi-homed scheduler for multi-stream support. Therefore, we provide Scream with a link that has the same amount of available bandwidth as the sum of the available bandwidth of all links given to our multi-homed systems. This comparison system represents Scream’s ability to stream video under perfect multi-homed scheduling. Our comparison system was adapted using Scream’s implementation provided by [123]. We evaluated various settings and found that setting the queue delay target to be 100ms and turning on Scream’s ECN feature to provide Scream with the highest VMAF.

Theoretical Optimal (TOpt)

TOpt’s achieved video quality is obtained by encoding videos at the total aggregate available bandwidth. These analytical results are a theoretical upper limit on the obtainable video quality.

4.4 Evaluation

In this Section, we evaluate the following Conflux User Preferences which we described in detail in Section 4.2.1:

- **Aggressive (CAgg):** This User Preference prioritizes video bitrate and models sigmoidal relationship between video bitrate and user QoE. Following Section 4.2.1’s Equation 4.3, the reward is 2^{VBR} .
- **Conservative (CCons90%):** This User Preference aims to avoid self-inflicted congestion by sending at 90% of the links’ estimated available bandwidth. Section 4.2.1 specifies its reward in Equation 4.4 where the threshold $t = 0.90$.
- **Conservative (CCons75%):** Same as CCons90% but the threshold $t = 0.75$ for Equation 4.4.

We compare Conflux with its comparison systems, investigate the differences between Conflux’s User Preferences, and characterize the effectiveness of Conflux’s recovery mechanisms.

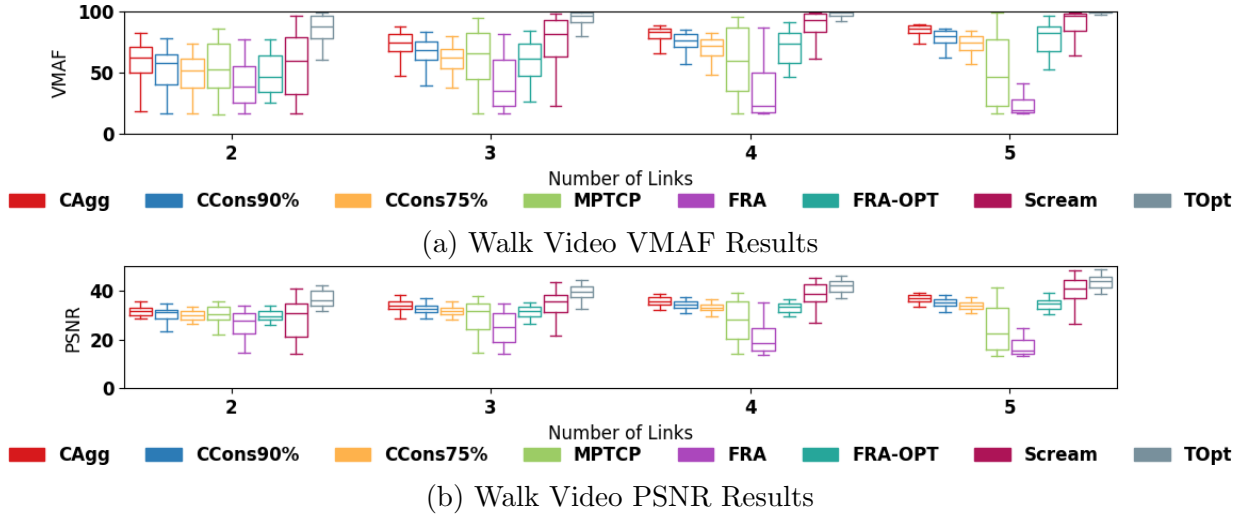


Figure 4.6: CDFs of video quality metrics for Walk video. Line towards the right is better.

4.4.1 Conflux vs. Comparison Systems

Figure 4.6a presents the CDFs for VMAF of 50 sessions of streaming the Walk reference video using the Mei et al. traces. We evaluate Conflux using up to five links as many devices support up to four links [34, 50, 89]. We compare Conflux User Preferences’ (Aggressive, Conservative 90%, Conservative 75%) achieved video quality with our comparison systems. Our results show that Conflux is able to scale as the number of links increase and often provides higher video quality than the non-optimal systems. VMAF values can be mapped to human perception categories of *excellent*, *good*, *fair*, *poor*, *bad* [79]. Our non-optimal comparison systems often achieve poor or bad video quality, and our results show that Conflux Aggressive achieves at least fair video quality in many of our experimental settings.

TOpt’s video quality is the theoretical maximum in our evaluation environment. Ideally, a system’s achieved VMAF in Figure 4.6a should be as close as possible to TOpt’s achieved VMAF. Figure 4.6a shows that Conflux Aggressive’s achieved VMAF is closest to TOpt, and this shows that it provides the highest video quality over all remaining comparison systems. Conflux Aggressive provides a median video quality that is 71% of TOpt at two links and 86% at five links. The difference between Conflux and TOpt narrows with more links because the total aggregate available bandwidth of four links is sufficient to achieve near-perfect (100) VMAF, and Conflux uses the fifth link to obtain higher video bitrate, resulting in higher video quality. The difference in video quality between Conflux and

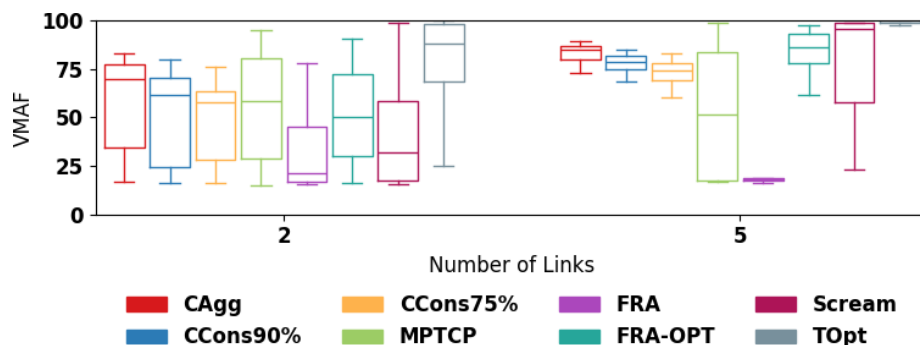
TOpt is due to Conflux encoding at a lower bitrate than TOpt. Conflux’s selects lower video bitrates since the video bitrate is adapted according to Conflux’s estimated available bandwidth, which requires time to determine.

Compared to FRA-JSCC, Conflux Aggressive consistently provides higher median video quality, between 61-336%, despite FRA-JSCC having perfect (oracle) knowledge of each link’s available bandwidth. We found that the assigned weights for the weighted round-robin packet scheduling resulted in over-sending on some links, leading to congestion and deadline violations. Furthermore, FRA-JSCC may also experience deadline violations due to encoder limitations which result in larger-than-expected video frames; we discuss these limitations in Section 4.5.

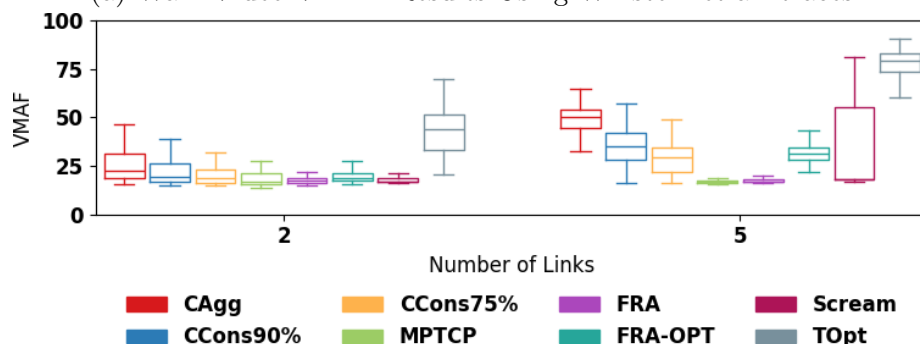
We remove these limitations and present FRA-JSCC’s theoretical maximum video quality as FRA-OPT in Figure 4.6a. When compared to FRA-OPT, we find that Conflux Aggressive is able to provide 34% higher median VMAF in the two-link environment. This is because FRA-JSCC uses parity data, between 24-25%, to ensure on-time video data arrival. Thus, FRA-JSCC forgoes additional video quality that can be obtained by using higher video bitrates than the ones selected. This improvement decreases to 21%, 13%, 4% for three, four and five links respectively because there is more available bandwidth to stream video.

The MPTCP Oracle achieves the highest VMAF of all the non-optimal, multi-homed comparison systems. Nonetheless, it offers lower median video quality than Conflux. As the number of links increase, the MPTCP Oracle’s scheduler has more chances to send at rates that are greater than the path’s available bandwidth. Consequently, the scheduler’s decisions may result in congestion that causes head-of-line blocking due to MPTCP’s in-order requirements. This causes the video stream to experience more late data, resulting in *poor* video quality in the five-link setting. Figure 4.6a illustrates MPTCP’s inability to scale its video quality with the number of links, and it shows that Conflux Aggressive provides an 18%, 13%, 37%, and 86% improvement in video quality over MPTCP Oracle for two, three, four, and five links respectively. This is because Conflux streams at a higher video bitrate and incurs less late data than the MPTCP Oracle.

We find that Conflux provides a 4%, -8%, -11%, and -11% improvement in the overall median video quality over Scream as the number of links increase from two to five in the high bandwidth environment. We find that the Scream comparison system drops video data from its RTP queue when the queue delay reaches a target threshold. As the number of links increase, a drop in available bandwidth in one link is more likely to be compensated by an increase in available bandwidth on another link. With perfect multi-homing available to Scream through the use of a single link, Scream is able to achieve



(a) Walk Video VMAF Results Using Winstein et al. traces



(b) Walk Video VMAF Results Using Riiser et al. traces

Figure 4.7: CDFs of video quality metrics for Walk video. Line towards the right is better.

higher VMAF than Conflux as the number of links increase. However, Scream also has a much longer tail than Conflux. This indicates that it may request and send too much video data, leading to congestion and delayed video data. As we have designed Conflux’s PSuccess function to be more conservative when increasing the video bitrate, Conflux has a much shorter tail in terms of video quality. Scream may experience more dropped data in a multi-homed environment since there is a greater chance that at least one path experiences congestion. This is also shown with other network environments.

Figure 4.6b shows that trends and results in video quality for VMAF also apply to the PSNR quality metric. PSNR is a quality metric that compares the difference in pixel values between the produced, streamed video and the reference video whereas VMAF measures video quality as perceived by humans [88]. As PSNR was widely used as a video comparison metric prior to the introduction VMAF and per industry recommendations [88], we include both metrics in our evaluation.

We also evaluate Conflux and its comparison systems using the Winstein et al. and

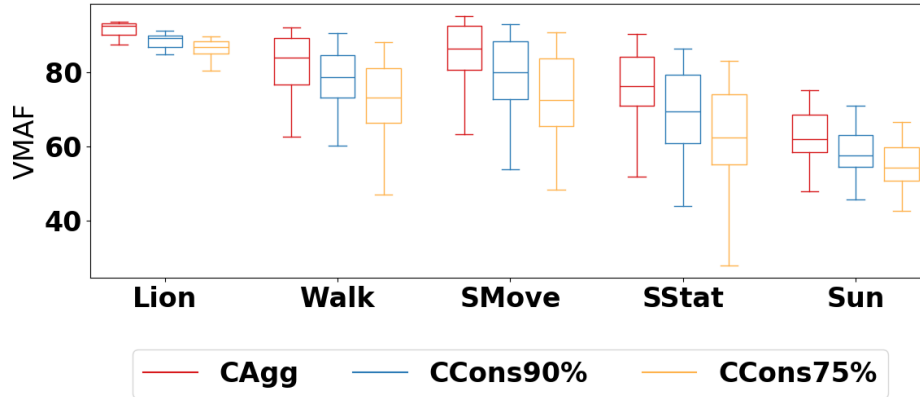


Figure 4.8: VMAF of Different User Preferences. Four-link Mei et al. environment

Riiser et al. traces for the two and five link settings. Figure 4.7a shows that Conflux Aggressive provides a 19% (two links) and 65% (five links) improvement over the best, non-optimal, multi-homed comparison system for the Winstein et al. environment, which incurs larger drops in available bandwidth than Mei et al. Scream is susceptible to the drops in available bandwidth, resulting in a median VMAF of 32, which is lower than Conflux’s median VMAF of 62 when there are two links. Furthermore, it has a significantly longer tail than Conflux. This supports our findings from our experiment that uses the Mei et al., or high bandwidth, traces.

Figure 4.7b shows that Conflux Aggressive can provide a 31% (two links) and 198% (five links) improvement in low-bandwidth environments such as the one found in the Riiser et al. traces. Because the links in this experiment have low bandwidth and because VMAF improvements are greater at low video bitrates (as shown in Figure 4.4), using additional links to obtain higher video bitrate results in large video quality improvements.

Figures 4.6 and 4.7 show that Conflux has lower maximum VMAF than MPTCP Oracle. This is due to the encoder generating less data than requested by Conflux, resulting in lower video bitrate and quality. We discuss impact of Conflux’s design and the interaction it has with the encoder in Section 4.5.

4.4.2 Conflux’s User Preferences

We now turn our attention Conflux’s various User Preferences that were first presented in Section 4.4.1. We loop our reference videos to four minutes in length to ensure that all video streams experience the same network conditions, and we stream over four links using

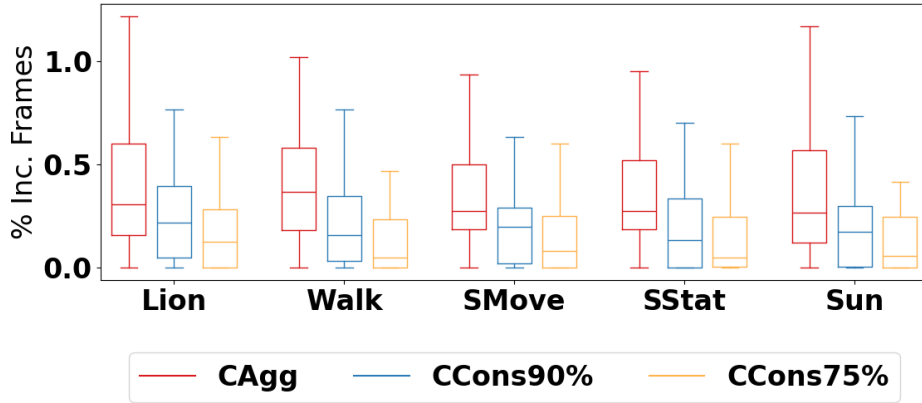


Figure 4.9: % Incomplete Frames of Different User Preferences. Four-link Mei et al. environment

our Mei et al. workload. Figure 4.8 shows that Conflux Aggressive consistently provides the highest video quality, where the median VMAF is between 62 and 92, for all of our reference videos since it uses nearly all of its available bandwidth for sending video data.

Although Conflux Aggressive provides the highest VMAF, there are cases where the Conservative User Preferences are more desirable. Specifically, the Conservative User Preferences have fewer incomplete frames, which is defined as a video frame that is missing bytes due to late data, than Aggressive. Figure 4.9 is a whisker plot of the percentage of incomplete frames experienced by Conflux’s User Preferences. It shows that Conservative 75% experiences the lowest percentage of incomplete frames, between 0.05% to 0.125% median percentage of incomplete frames, as it is incentivized to have a lower sending rate than the other User Preferences. Conflux Aggressive experiences the highest percentage of incomplete frames (0.28% to 0.39%) as it sends as much video data as possible and is the most susceptible to changing network conditions.

We previously evaluated Conflux and its comparison systems using traces with variable bandwidth to emulate video streaming over LTE networks. However, LTE links can also experience changing latency when signal loss occurs and packets are lost. The LTE network’s link layer handles packet loss through detection and retransmission; thus, users experience an increase in latency at the application level when signal loss occurs [15, 19]. Since our existing traces do not include latency measurements, we extend them with latency measurements to evaluate the impact of variable latency on video streaming. Our updated traces also use bandwidth measurements from Mei et al. [97] and periodic increases in one-way delay every 30s (Poisson distributed) for 1-2s (uniformly distributed). Our traces’ baseline one-way delay is 20ms, and we experiment with different increased

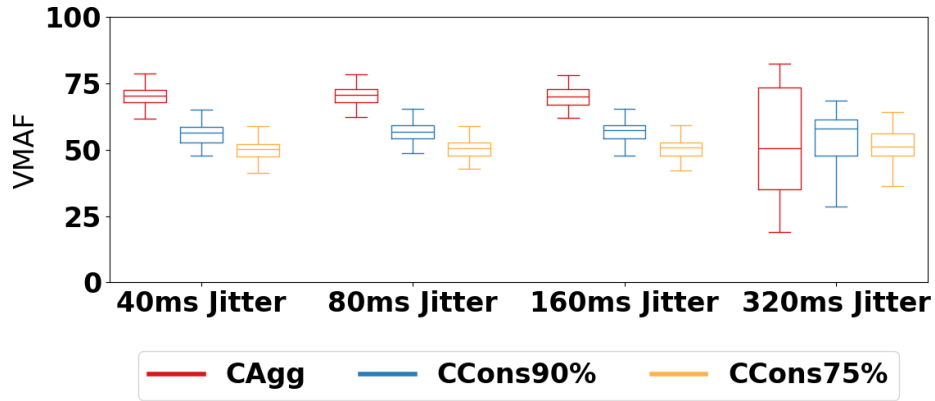


Figure 4.10: VMAF of Different User Preferences. Five links with periodic increases in latency and drops in available bandwidth.

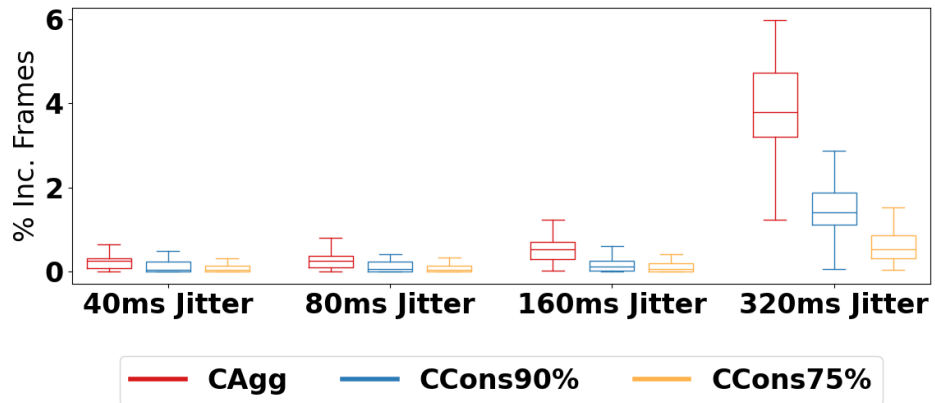


Figure 4.11: % Incomplete Frames of Different User Preferences. Five links with periodic increases in latency and drops in available bandwidth.

delay values including 40ms, 80ms, 160ms, 320ms (each direction) to emulate signal loss. Furthermore, as signal loss is often accompanied by a reduction in available bandwidth [42], we also reduce the available bandwidth by half when the latency increases.

Figure 4.10 shows the VMAF results, and Figure 4.11 shows percentage of incomplete frames when signal loss is emulated. The Aggressive User Preference has the highest VMAF and percentage of incomplete frames of all the User Preferences as it is incentivized to use as much of the available bandwidth as possible. This is beneficial when the jitter is sufficiently low so that Conflux is able to maintain a low percentage (less than 1%) of incomplete frames such that VMAF is not significantly affected. Once jitter reaches to

320ms, we find that the Aggressive User Preference experiences more delayed data than the Conservative User Preferences; thus, it has a lower VMAF. Because the Conservative User Preference does not attempt to saturate its links, it experiences fewer incomplete frames as is able to obtain a higher VMAF than the Aggressive User Preference.

4.5 Conflux’s Interaction with Encoders

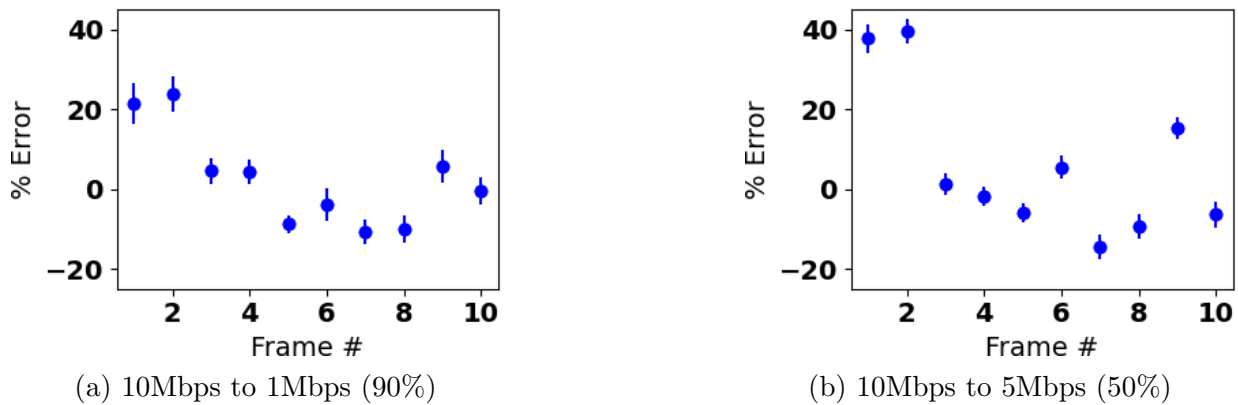


Figure 4.12: % encoder error (average and 95% confidence interval) of each subsequent frame after target bitrate is reduced as indicated in the subcaption. Frame # X indicates the X^{th} frame since the change in target bitrate. Positive % encoder error indicates that the produced frame is greater than expected.

Live video streaming applications such as Conflux use one-pass video encoding to meet their low latency requirements [73]. However, using one-pass encoding may not produce video data at the requested or target bitrate [82, 99]. The number of frames that the encoder requires to reach the target bitrate depends on the video content [124].

To characterize the encoder’s accuracy of generating video frames at the target bitrate, we analyze the encoder’s percentage error where a positive percentage error indicates that the encoder produced more data than requested. Figure 4.12a shows the average and 95% confidence intervals of the encoder’s percentage error between the targeted and produced video bitrate when the target video bitrate drops from 10Mbps to 1Mbps (90% drop). Similarly, Figure 4.12b shows the encoder’s percentage error when the target video bitrate drops from 10Mbps to 5Mbps (50% drop). Figure 4.12 shows that three frames must elapse before the encoder reaches the new target bitrate. These errors depend on the video

content or the current state of the encoder [73]. In our deployment which uses the Walk video, the encoder requires between 120 *ms* to reach the target bitrate.

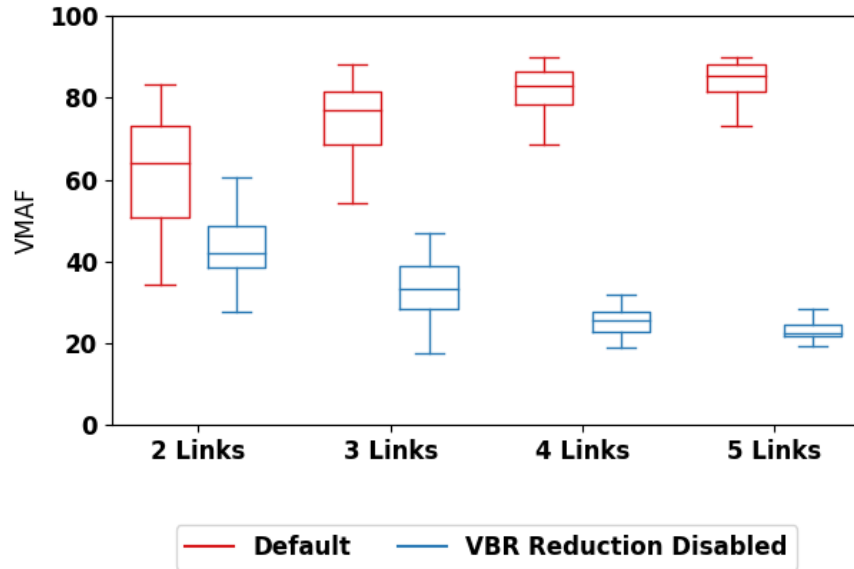
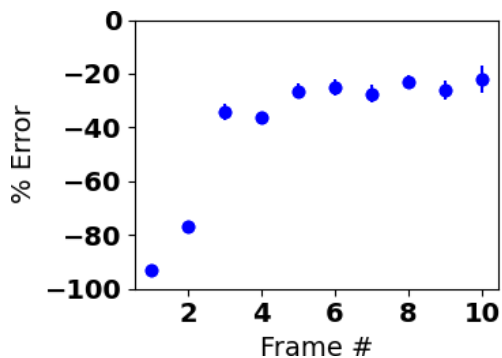


Figure 4.13: VMAF comparison of Conflux with is VBR Reduction Mechanism vs. Conflux with VBR Reduction Disabled.

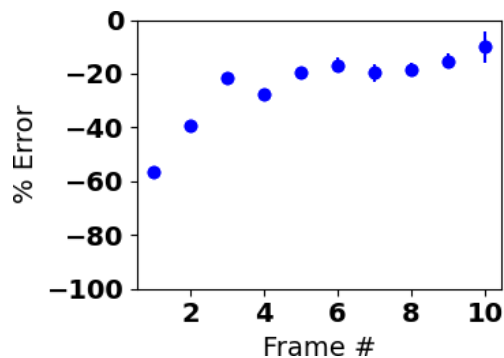
When the encoder produces more data than requested, the data may remain in the sender-side buffer for an extended period of time because Conflux limits the sending rate according to its Configuration (Section 4.2.2). By remaining in the sender-side buffer for an extended period time, data is more prone to late arrival since it has less time to reach the receiver.

Our approach to prevent video data from remaining and missing its deadline while in the sender-side buffer in Section 4.2.2 using Conflux’s buffer-size-based video bitrate reduction mechanism. We evaluate Conflux Aggressive with and without its video bitrate reduction mechanism using the Mei et al. [97] workload in the 2, 3, 4, and 5-link settings. This optimization reduces the video data’s median waiting time from 275 - 456 ms to 47 - 71 ms. A median waiting time of 47ms is acceptable because a frame can be sent before the next frame is generated. Figure 4.13 shows that using this optimization results in up to 259% higher video quality (VMAF) than if it was not included.

Conversely, we find that Conflux’s VMAF may also be limited due to the time required for the encoder to adjust to a higher video bitrate. Figure 4.14a and Figure 4.14b the percentage error when the requested video bitrate changes from 1Mbps to 10Mbps and



(a) 1Mbps to 10Mbps



(b) 5Mbps to 10Mbps

Figure 4.14: % encoder error (average and 95% confidence interval) of each subsequent frame after target bitrate increases from the amounts as indicated in the subcaption. Frame # X indicates the X^{th} frame since the change in target bitrate. Negative % encoder error indicates that the produced frame is smaller than expected.

5Mbps to 10Mbps respectively. We find that the resulting frame size can be as small as half of the requested frame size. These figures show that a significant number of frames must elapse before the percentage error is greater than -30%.

As the resulting video bitrate does not reach the requested video bitrate quickly, this results in foregone video quality as video bitrate is correlated with video quality metrics such as shown in Figure 4.4. Because the encoder requires time to adapt when Conflux lowers and re-adjusts the requested video bitrate in the event of congestion, Conflux does not obtain the optimal video bitrate.

4.6 Chapter Summary

In this chapter, we presented Conflux: A Multi-homed Adaptive Bitrate Protocol for On-Site Live Video Streaming. Conflux is a modular protocol that encapsulates link quality characterization, packet scheduling, and video bitrate adaptation into different, independent modules. We presented the PSuccess module that represents link quality as a function of sending rate to the probability that a link can deliver data on time at this rate. This abstraction allows Conflux to evaluate the trade-offs between video bitrate and on-time arrival. We introduced the User Preferences modules that specifies the user's preferred trade-off as a utility function. The links' PSuccess modules and the user's User Preference module are used as input to the Configuration Finder to find the video bitrate and the

links' sending rates that maximize the user's expected utility. This modular design allows Conflux to be extended to different types of network links and various types of users with minimal changes to the core framework.

Our evaluation showed that Conflux provided at least 13% improvement in VMAF over other non-optimal, multi-homed comparison systems for the Mei et al. workload. We illustrated that our various User Preferences were able to make their intended trade-offs between video bitrate. We characterized short-comings of the video encoder to show that single-pass encoders may not always produce video data at the requested rate. To address this, we introduced Conflux's video bitrate reduction mechanism that reports a lower target bitrate to avoid video data being queued in the sender-side buffer. We showed that this is necessary to ensure that video data does not expire in the sender-side buffer.

Chapter 5

Recovery Mechanisms in Conflux

Introducing recovery for lost or delayed packets containing video data is important in ensuring that the video stream does not experience disruptions when link quality degrades or when a connection is lost. This is because having 2% delayed data can already halve video quality metrics such as PSNR [139]. Determining the impact of recovery mechanisms is important because using them comes at the cost of using available bandwidth that could have been used for obtaining a higher video bitrate.

In this chapter, we introduce recovery mechanisms to Conflux to increase the amount of video data that arrives on time when links experience congestion or when links drop packets in network environments that do not have underlying retransmission mechanisms. In designing recovery mechanisms, we consider retransmission and using Forward Error Correction (FEC) to ensure that the video data arrives on time at the receiver.

Packet retransmission in the multi-homed environment can be effective at preventing delayed video data when a link experiences congestion since there may be alternate paths that are able to deliver the video data on time. Retransmitting delayed data is preferable to sending redundant data as bandwidth is used only when packets are delayed, and this is beneficial for cost-sensitive users who do not want to expend additional bandwidth unnecessarily. For retransmission to be effective, the timeout should be long enough to avoid incorrectly identifying the link as being congested while being short enough so that there is sufficient time for the retransmitted data to arrive by its deadline. Furthermore, there should be sufficient available bandwidth to send retransmitted and new video data.

An alternative to retransmission is FEC. FEC constructs parity data from the original data, and all data is sent at the same time. With FEC, not all data is required to arrive at the receiver for the original data to be reconstructed. We integrate FEC into Conflux and

use cm256 [135] which is a Cauchy MDS Block Erasure Codec. We partition our video data into multiple packets and use the cm256 library to generate parity packets. The primary challenge in using FEC is determining the ratio between the amount of original data and parity data. Using an insufficient amount of FEC may result in delayed video data, and sending too many parity packets results in unnecessary bandwidth usage. FEC is an especially effective strategy to use when user deadlines are short compared to the available links' delay and when retransmission is not possible. Ideally, FEC should be used when it increases video data's probability of on-time arrival by improving the user's expected utility. Alternatively, some users may prefer to specify a minimum, fixed redundancy level which is helpful when there are sudden and unexpected link failures such that the link's PSuccess model is inaccurate before the link fails.

We also explore using Conflux in network environments that do not have underlying link-layer retransmission mechanisms. In our previous chapter, we designed and evaluated Conflux for network environments that have underlying retransmission mechanisms (LTE); thus, packets experience significant increases in latency if link quality deteriorates. In this chapter, we evaluate Conflux in network environments that experience packet loss according to the Gilbert-Elliot model and the random loss model. We show that Conflux is able to deliver video data on time and is able to achieve good user perception in many cases. This demonstrates that Conflux can be used in other network environments in addition to LTE.

Our chapter contents are as follows: Section 5.1 introduces recovery mechanisms to Conflux. We describe our methodology for evaluating Conflux's recovery mechanisms in Section 5.2. Section 5.3 evaluates Conflux's recovery mechanisms and shows that there are limited opportunities for retransmission, that minimum levels of redundancy can help protect against delayed data, and that Conflux can deliver data in environments that experience packet loss.

5.1 Conflux Extensions to Support Recovery

In this section, we describe Conflux's architectural extensions to support retransmission and FEC. We specify how the Configuration Finder uses the PSuccess function to estimate the probability that a video frame arrives and the changes made to the Packet Assignment Module to support retransmission. Furthermore, with the inclusion of recovery mechanisms, we can now enforce minimum levels of redundancy for users; thus, we introduce new User Preferences that incentivize a minimum level of redundancy.

5.1.1 FEC-Enabled Configuration Finder

With the introduction of FEC, the Configuration Finder now determines level of redundancy, in addition to the video stream’s video bitrate, that provides the greatest expected utility for the end user. In Conflux, data redundancy is provided using a (n,k) block FEC code. In this code, the video data (i.e., a video frame) is divided into k fixed-sized blocks, which we refer to as the original blocks. The original blocks are used to generate $n - k$ redundant blocks that contain parity information. A total of n blocks are sent; however, only k blocks need to arrive to reconstruct the video data. As all blocks are sent over the multiple, available links, the video stream is able to tolerate link failures and congestion as video or parity data can arrive on other links.

In our implementation, we use cm256 [135] which is a Cauchy MDS Block Erasure Codec. We select this codec because it is computationally fast, and this is important as Conflux must create a video frame’s original and parity blocks before the next video frame arrives. We use 512-byte blocks, and we discuss our packet size selection in the following subsection (Section 5.1.2).

To determine if redundancy improves the user’s expected utility, the Configuration Finder requires the probability that the video frame arrives on time for different redundancy levels. In Chapter 4, the probability that the video frame arrives on time was simply the product of all links’ PSuccess functions at their respective sending rate since all data must arrive on time for the video frame to be reconstructed. With the introduction of FEC, the Configuration Finder must determine the probability for all possible packet-arrival combinations that can reconstruct the video frame since not all sent packets are required to arrive. To accomplish this, we introduce *PVideoFrameArrival* as the probability that the video frame is reconstructed by the user’s deadline at a specific video bitrate and redundancy level.

We use each link’s PSuccess function to calculate the probability that each packet arrives, and this probability is used to calculate the probability that the video frame arrives (*PVideoFrameArrival*). We require the probability of arrival for each individual packet for calculating *PVideoFrameArrival* since not all packets are required to arrive on time. We define *PPacketArrival(P)* to be Packet P ’s probability of on-time arrival. *PPacketArrival(P)* is defined in Equation 5.1 where Packet P is sent on Link L_i . Equation 5.1 uses P ’s offset and size to calculate the input sending rate to L_i ’s PSuccess function. The amount of time used in this calculation, referred to as *SendInterval*, is the time between sending data. If P is the last packet scheduled on its link, *PPacketArrival(P)* (abbreviated as *PPA(P)*) is the same as the link’s *PSuccess*.

i	$PSuccess(i, 1)$	$PExact(i, 1)$	$PSuccess(i, 2)$	$PExact(i, 2)$
0	1.0	0.1	1.0	0.01
1	0.9	0.1	0.99	0.01
2	0.8	0.1	0.98	0.23
3	0.7	0.7	0.75	0.75

Table 5.1: Example showing Link 1’s and 2’s probability values that at least i packets arrive (PSuccess) and exactly i packets arrive (PExact)

k	$PVideoFrameArrival$ for different values of k
1	0.999
2	0.997
3	0.972
4	0.866
5	0.761
6	0.525

Table 5.2: $PVideoFrameArrival$ for different values of k when $n = 6$

$$PPA(P) = PSuccess_{Link_i} \left(\frac{\sum_{j=1}^P PacketSize(Packet_j)}{SendInterval} \right) \quad (5.1)$$

The probability of arrival for each packet depends on the assumption that if a packet is delayed due to congestion, packets that are sent after the delayed packet are also delayed. This assumption is generally true in LTE networks because packets sent after a delayed packet are likely queued behind the delayed packet as the LTE’s Radio Link Control performs retransmission and reordering [64]. Using this assumption, Conflux’s packet scheduling decision, and the links’ PSuccess functions, we enumerate the different, possible ways that packets can arrive across the available links to calculate $PVideoFrameArrival$.

Calculating $PVideoFrameArrival$ requires the number of packets that are sent on each link and each link’s PSuccess function. Conflux schedules packets on links that provide the greatest $PPacketArrival$ value. This simplifies the number of Configurations to search through as the Configuration Finder does not need to consider all possible combinations of packet scheduling decisions.

We show how $PVideoFrameArrival$ is calculated. We begin with Table 5.1 that shows the PSuccess values of two links (Link 1 and Link 2) for three packets each. The PSuccess

values are used to calculate the probability that exactly i packets arrive on each link (i.e., $PExact(i, LinkID)$). $PVideoFrameArrival$ is calculated using these probabilities, and $PVideoFrameArrival$ is shown in Table 5.2 for different values of k when $n = 6$. The Configuration Finder searches through and evaluates configurations with different n and k values, and Conflux selects the configuration that provides the highest expected utility, which is calculated as $PVideoFrameArrival \times U(C)$ where $U(C)$ is the reward.

There is a short time budget to determine the n and k values that maximizes the user’s expected utility because the video encoder must be informed of the updated video bitrate to adapt the size of its generated frames to the changing network conditions. Although exhaustive algorithms may find the configuration with the maximum expected utility, they may require more time than is available. As a new video frame is generated every 40 ms, we budget half of that time to find a new configuration to allow the video encoder to adjust its bitrate. Furthermore, we budget this amount of time because the LTE schedules packets to be sent at this time interval [6].

To address the challenge of meeting the time budget for selecting the updated configuration, we reduce our search space by conducting a coarse-grained search through possible values of n and k where n is the number of packets with $PPA(P) > 0$. Conflux uses the configuration from this subset with the highest expected utility.

Determining the best configuration by analyzing the links’ PSuccess functions is not possible when link quality information is unavailable or when all links are unable to deliver data on time. In these scenarios, the Configuration Finder uses a user-provided, default configuration. In our implementation, Conflux selects a low video bitrate and replicates video data across all links to ensure on-time video data arrival, to collect link quality information, and to prevent over-sending that can cause congestion for future video data.

5.1.2 Blocksize Selection

We select 512 to be our blocksize for FEC, and each block forms a packet. Although 512 is smaller than standard packet sizes, we have found that the additional overhead from using this packet size does not affect video quality as measured using VMAF.

Using a smaller packet or block size results in additional overhead in partitioning video data, creating redundant data, processing statistics for PSuccess, and determining the configuration. This additional overhead can result in video data being sent later than the intended 20ms interval and a stale PSuccess function. As a result, the achieved video quality suffers, as shown in Figure 5.1, for small packet sizes. From these results, we select 512 bytes as our packet size in our remaining experiments.

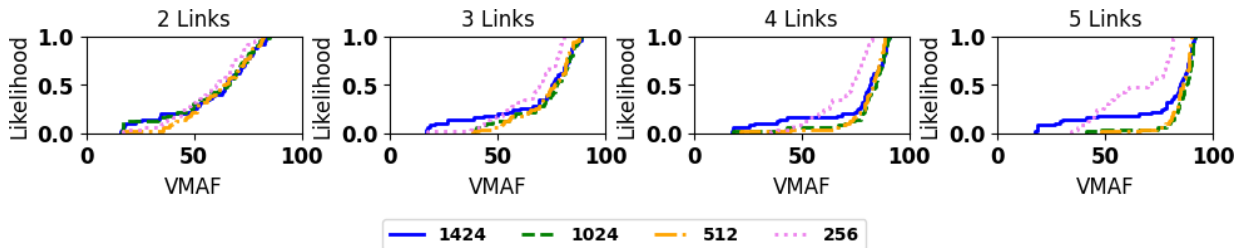


Figure 5.1: CDF of VMAF for Conflux with varying packet sizes

Figure 5.1 shows the resulting video quality of Conflux Aggressive in the Mei et al. [97] environment, which was first presented in Section 4.3, for varying packet sizes that include: 256, 512, 1024, and 1424 (such that the total packet size is 1460 when the 36-byte header is included). We find that 512 and 1024 provide the highest video quality. Given that there is no significant difference between 512 and 1024, we select 512 because it supports finer-grained granularity for FEC.

5.1.3 Packet Retransmission

Designing a retransmission mechanism requires that we determine how much available bandwidth should be used for packet retransmission and when to retransmit a packet. Unfortunately, we expect that retransmission in Conflux to have few benefits because there is often insufficient bandwidth for both retransmitted and new video data since we are saturating the links with new video data from the encoder. We evaluate the opportunities for retransmission and its effectiveness in Section 5.3.1 and find that they are limited in cases where links are saturated to maximize utility.

One possible solution to ensure sufficient bandwidth for sending retransmitted and new video data is to reserve available bandwidth for retransmission. However, this results in less new video data being sent; thus, the resulting video quality is equivalent to sending parity data with the reserved bandwidth. The advantage in reserving bandwidth for retransmission instead of sending parity data immediately is lower bandwidth usage as only packets that are expected to be delayed are retransmitted. A drawback to this solution is that it is likely less effective than FEC at improving on-time video data arrival since retransmitted packets have less time to arrive at the receiver.

An alternate solution is to prioritize retransmission instead of sending video data that has not been transmitted (e.g., the next frame, or remnants of the current frame if the encoded provided too much data). However, this causes unsent video data to have less

time to arrive at the receiver or expire in the sender-side buffer as the video frame was encoded earlier and at a higher bitrate that did not account for the need of additional bandwidth to retransmit data. One potential method to ensure that there is a video frame whose size is equal to or less than the target bitrate is to use multiple encoders to generate these frames so that they are available when needed. However, this comes at an additional cost (e.g., hardware and computation). In this thesis, we do not explore this solution.

Conflux’s Retransmission Mechanism

The Packet Assignment Module (PAM) implements Conflux’s retransmission mechanism. It determines the unacknowledged packets and the links that they can be resent on and still arrive on time. A delayed packet can be retransmitted on any link that has sufficiently low delay and the ability to deliver the packet by its deadline. If all available links cannot deliver the unacknowledged packet on time, the unacknowledged packet is dropped at the sender to avoid expending additional bandwidth.

We use a packet’s lack of acknowledgment by its timeout to determine if it is a candidate for retransmission. The timeout value should be sufficiently high such that the link is most likely experiencing congestion. Furthermore, the timeout value should also be sufficiently low so that delayed packets can still arrive on time if they are resent. We select a retransmission timeout value of four times the link’s Round Trip Time (RTT) as this is a common threshold used by many congestion controls protocol to identify congestion. As packets have deadlines, retransmission is not useful when the deadline is within four times the RTT. Users whose deadlines are shorter than this threshold should use FEC instead of retransmission to prevent video data from arriving past its deadline.

PAM first resends unacknowledged packets that can arrive on time given current link conditions. PAM sends the packets that should be retransmitted prior to sending unsent, new video data packets. This is because packets sent at the beginning of a series of packets are less likely to experience additional queueing delay in the link’s network buffers that is the result of congestion. Furthermore, PAM only retransmits packets if there is available bandwidth to send both the unacknowledged packets and new (unsent) video data. We choose to prioritize new video data because we do not want to introduce additional delays to the new video data that can negatively affect future frames. We prioritize unsent video data because unacknowledged data may still arrive at the receiver; thus, retransmitting data may be unnecessary.

Retransmission is advantageous over FEC when the user has limited budget for sending data and its maximum desired video bitrate is lower than the total available bandwidth.

Unfortunately, even when the estimated total bandwidth is greater than the user’s maximum target bitrate, Conflux does not send more data than the user-specified maximum. This is because the Configuration Finder limits the search of video bitrates and sending rates on each link according to its User Preference that specifies this maximum. Consequently, the Scheduler’s rate control feature limits the sending rate on each link according to the bitrate specified by the Configuration, leaving little to no available bandwidth for retransmission.

One method to approximate and allow the use of additional available bandwidth for retransmission is to omit the limit for the video bitrate when searching for the Configuration and subsequently report the user’s maximum video bitrate to the encoder if the found video bitrate is higher. This allows Conflux to send at higher bitrates so that there is more available bandwidth to retransmit packets if necessary. This change allows us to approximate the improvement in video quality and the overhead of retransmission. We evaluate the impact of using additional available bandwidth for retransmission in Section 5.3.1

Because of the delay requirements of our target application and because of existing ARQ mechanisms in our expected deployment environment, we largely focus on the use of FEC at maximizing the user’s expected utility. Future work includes characterizing the probability of requiring retransmission, characterizing the improvement in expected utility for retransmission, and evaluating its impact on the user’s expected utility when evaluating a Configuration.

5.1.4 User Preferences

Although Conflux already considers using redundancy with other utility functions, delay-sensitive users can specify a utility function that enforces a minimum amount of redundancy to ensure on-time video data arrival. This is desirable for users who do not want to experience any delayed video data even in the presence of sudden, unexpected drops in available bandwidth.

Equation 5.2 enforces a user-specified level of redundancy by only granting a reward if the proposed configuration meets this level. The amount of redundancy is specified as the *Coding Rate* (CR), which is defined as the ratio of original data sent to total data sent. In Conflux, this is the ratio between $C.VBR$ and $C.SR$. Equation 5.2 specifies the Redundant utility function, and $f(C)$ is the reward if the configuration meets the user’s redundancy requirements. The reward is the sum of the links’ sending rates that send below their available BW minus a penalty for links that send above the predicted available bandwidth where the penalty is $BW(l) - SR(l)$.

$$U(C) = \begin{cases} f(C) & \frac{C.VBR}{C.SR} \leq CR \\ -inf & \text{Otherwise} \end{cases} \quad (5.2)$$

5.1.5 Extensions to Feedback Messages to Handle Packet Loss

As introduced in Section 4.2.1, the PSuccess function models the link’s ability to deliver data on time at its input sending rates. This function is constructed based on latency measurements (i.e., if the latency of a recent period is greater than the minimum latency of a longer period). Consequently, packets that are dropped prior to being received do not have latency measurements.

Conflux uses periodic feedback messages (sent once every 10ms) called *Extended Acknowledgment Messages* that contain latency information for the last epoch. Dropped packets, at a low enough rate such that other packets are received in this window, do not affect the latency measurements in this epoch since these measurements come from the packets that have arrived during this epoch. In an environment without packet loss, the extended acknowledgment message serves as a cumulative acknowledgment message. However, in an environment with packet loss, we include a list of unacknowledged packets in the extended acknowledgment message. This allows Conflux to determine which packets to resend.

The Extended Acknowledgment Message specifies the range of packet sequence numbers for its epoch. Therefore, if an Extended Acknowledgment Message is lost, then all packets in the message’s epoch are treated as unacknowledged and are resent by PAM. We do not expect that Extended Acknowledgment Messages to be lost often due to the ARQ mechanisms of our expected deployment environment and because the path that these messages take is not the bottleneck.

5.2 Methodology for Evaluating Recovery Mechanisms

We evaluate the effectiveness of Conflux’s recovery mechanisms and its impact on preventing delayed video data by streaming video over various trace-driven, emulated network environments. In this section, we describe the traces that specify the available bandwidth of our wireless links and Conflux’s User Preferences.

5.2.1 Traces

Similar to Chapter 4, we select Mininet [100] as our emulation tool and use traces from Mei et al. [97] for bandwidth measurements and Winstein et al. [145] for one-way delay of 20 *ms*.

Modeling Packet Loss

The Mei et al. [97] traces also lack packet loss information. Works such as [15, 59] have found that applications seldom experience lost packets because they are concealed with underlying link-layer retransmission (ARQ) mechanisms and large buffers. In LTE networks, packet loss is manifested at the application level as a temporary increase in latency [19, 59]. We emulate packet loss by periodically increasing latency temporarily by 40ms to 320ms each way, and this increase is specified for each experiment.

To emulate network environments, we use the Gilbert-Elliot loss model and random packet loss with varying loss rates. The Gilbert-Elliot model is commonly used to describe burst errors and packet loss in wireless channels. This model comprises *good* and *bad* states, the transition probabilities between the two states (i.e., P_{gb} is the transition between the good to the bad state, and P_{bg} is the transition between the bad to the good state), and the packet loss rates of each state. We denote P_{good} as the probability of losing a packet in the good state, and P_{bad} as the probability of losing a packet in the bad state. We select our P_{gb} , P_{bg} , P_g , and P_b values based on existing literature [19, 44], and we evaluate Conflux using these values to understand Conflux’s ability to deliver data on time in different network and loss environments. As packet loss without link layer retransmission is not common for LTE links, we do not limit ourselves to LTE packet loss models.

In our first Gilbert-Elliot-based loss model, we use the findings of Bocharova et al. [19], and set the values of the Gilbert-Elliot loss model to: $P_{gb} = 0.0489$, $P_{bg} = 0.1505$, $P_g = 0.0511$, and $P_b = 0.3075$. These parameters were determined by fitting traces that were collected on a vehicular testbed with multiple subjects that send monitoring packets to a server once every 50-100ms.

In our second Gilbert-Elliot-based loss model, our parameters are: $P_{gb} = 0.0223$, $P_{bg} = 0.2533$, $P_g = 0.0046$, and $P_b = 0.991$. Feng et al. [44] determined these parameters by fitting their packet loss measurements which were collected by sending UDP packets between two mobile devices. In this workload, if a link enters a bad state, nearly all packets are lost.

In addition to the Gilbert-Elliot-based loss models, we evaluate Conflux in environments that experience random packet loss. In this model, packets that are lost do not depend

on the link’s current state, and the probability that a packet is lost does not depend if the previous packet sent on the link was lost. We change the packet loss rate every five seconds, and we evaluate Conflux with a wide range of packet loss rates.

5.2.2 Metrics

To evaluate Conflux’s ability to stream live video in different network environments, we use the following metrics:

- **Video Multi-Method Assessment Fusion (VMAF)**: Developed by Netflix [105], it is a score between 0 to 100 where 100 indicates that the produced video is identical to the reference video.
- **Percentage of Incomplete Frames**: This measures the percentage of video frames that are missing some or all of their video data.

5.2.3 Videos and Settings

Our application streams the Walk reference video that has been looped for four minutes and was captured at 1080p at 25fps on our device (iPhone 12). This video was first introduced in our evaluation in Section 4.3. Similarly, we use the same video streaming settings as found in Chapter 4. Our acceptable delay is also 500ms, and we use Adaptive BitRate (ABR) encoding with intra-refresh mode and the key frame interval set to the maximum value [2, 63].

5.2.4 User Preferences

We focus on the following User Preferences that use the available links at different capacities.

- **AggNoRec**: This is the Aggressive version of Conflux presented in Chapter 4; thus it does not have any recovery features. We use this version as a baseline to show the advantages of using recovery mechanisms. This uses a reward function of 2^{VBR} .
- **CCons90%**: The Conservative User Preference that sends at 90% of the link’s available bandwidth.

- **Agg**: This is the Aggressive User Preference with a reward function of $f(C) = 2^{VBR}$. This version includes the use of retransmission, and redundancy is used if it is able to improve the user’s expected utility.

CCons90% and Agg are User Preferences in the version of Conflux that includes recovery mechanisms, which we refer to as *Conflux+Rec*. Comparing these User Preferences with AggNoRec allows us to characterize the impact that retransmission and FEC have on our evaluation metrics.

Furthermore, with the introduction of FEC, users can also specify their minimum level of redundancy. We can also specify their reward function to indicate additional preferences regarding link usage. We evaluate the following User Preferences:

- **CRed10%**: This is described in Section 5.1.4 where the reward is the sum of the links’ sending rates that send below their available bandwidth and a penalty for links that send above their available bandwidth. This User Preference incentivizes using at least 10% redundancy and sending at lower rates, and it is designed for significantly loss-adverse users.
- **CRedAgg10%**: This User Preference requires at least 10% of redundancy to obtain the reward. The underlying reward function is $f(C) = 2^{VBR}$, which is the same as Agg. The motivation for this User Preference is to allow us to determine if the Aggressive User Preference can benefit from redundancy.
- **CRedAgg25%**: Uses the same underlying reward function as CRedAgg10%, but requires at least 25% redundancy.

5.3 Evaluation

In this section, we evaluate Conflux’s recovery mechanisms, which includes retransmission and FEC, and extensions for network environments without link-layer ARQ. We also explore Conflux’s Retransmission mechanism by investigating the frequency that it is required and by showing that reserving available bandwidth by reporting a lower video bitrate can improve video quality.

5.3.1 Impact of Retransmission

We investigate the frequency that retransmission is required and how often there is sufficient bandwidth to retransmit delayed video data. We measure the following for every Conflux $20ms$ send interval (i.e., time that Conflux sends video data - see Section 4.2.1):

- *NumUnackedPackets* : The number of packets that are unacknowledged by their deadline.
- *NumMaxRetransmitPackets* : The number of packets that Conflux can retransmit in its $20ms$ send interval. This does not consider the links' delay.
- *NumResentPackets* : The number of packets that are resent. We note that *NumResentPackets* can be less than *NumMaxRetransmitPackets* when a link's delay is too high such that retransmitted packets cannot arrive on time.

We first evaluate Conflux using the same network environment first presented in Section 4.3.1 that follows Mei et al. [97] bandwidth measurements for two to five links. We find that Conflux only requires retransmission for less than 1% of the video stream for all link configurations. This shows that retransmission is often unnecessary for typical workloads. Secondly, we find that when retransmission is needed, there is insufficient bandwidth. Of the very few times that require retransmission, there is insufficient bandwidth 88% (or $NumMaxRetransmitPackets < NumUnackedPackets$) of the time (average of all test cases over 2 to 5 links). We find that can send an average of 9% of the data that require retransmission ($NumResentPackets/NumUnackedPackets$). This is because retransmission is required when link quality degrades and the available bandwidth drops. As video frames are currently being encoded at the rate prior to the link degradation, there is likely insufficient bandwidth to send the new video frame and packets requiring retransmission.

The Cost and Benefits of Retransmission

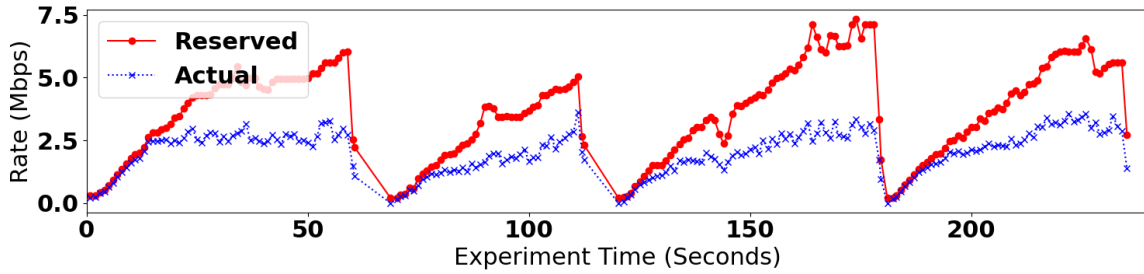
Retransmission can be beneficial for preventing data that is currently in transmission (e.g., queued in the network) or that is part of the next video frame from being late. Unfortunately, there are limited opportunities for retransmission since we are saturating the link. However, one scenario where retransmission is beneficial is when a user has a video bitrate limit due to budget constraints, and available bandwidth is reserved for retransmission. Conflux's model does not include a reward for reserving available bandwidth for retransmission. However, we can approximate this feature and evaluate the benefits of

retransmission by limiting the reported video bitrate to the encoder after finding the Configuration. To evaluate the benefits of retransmission we introduce the following settings:

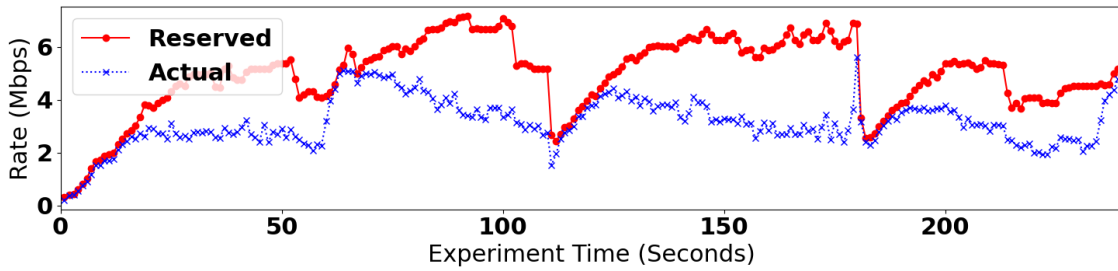
- **Max Video Bitrate - No Resends (NoRetr.):** We artificially limit the reported video bitrate to the encoder and disable retransmission. This reference setting is used as a baseline to illustrate the benefits of retransmission.
- **Max Video Bitrate - Allow Resends (Retr.):** We artificially limit the reported video bitrate to the encoder and enable retransmission. We do not limit the video bitrate or sending rate when searching for the configuration. This setting approximates the video quality that can be achieved if Conflux reserves available bandwidth for retransmission.

We disable FEC for both versions. We illustrate the video quality when the video bitrate is limited to 5Mbps. We select a video bitrate that could allow us to achieve fair video quality for our Walk reference video.

We evaluate Conflux in a controlled network environment to measure impact of reserving available bandwidth for retransmission. We have found that Conflux often did not require retransmission in our aforementioned (Mei et al., Riiser et al., and Winstein et al.) workloads and that the overall impact of retransmission is limited. Thus, we construct a workload where the available bandwidth on links are relatively high and stable for a long period of time before a link quality degradation event which illicit the need for retransmission. In this workload, stable links have between 6 to 6.3Mbps (changes once every five seconds) so that they can support the video stream and retransmitted data. We also designated one link to be unstable, and this link also has 6 to 6.3Mbps of available bandwidth; however, its available bandwidth drops to 1Mbps and its latency increases to 500ms once every 60 seconds (Poisson distributed) for 1s. We evaluate our two settings using two links.



(a) Unstable Link



(b) Stable Link

Figure 5.2: Comparison of Reserved and Actual sending rates for system with retransmission

Figure 5.2 shows a timeseries of the reserved and actual sending rate of each link (stable and unstable). Figure 5.2a shows how Conflux is able to reduce the sending rate when the drop in available bandwidth occurs, and Figure 5.2b shows how it is able to make use of the stable link for resends and new video data.

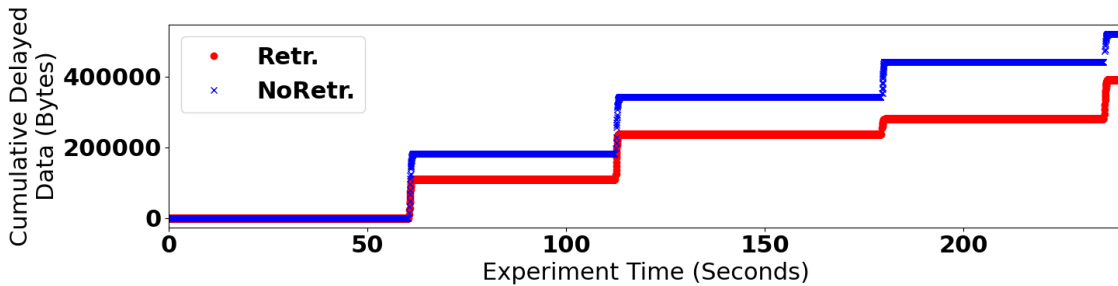


Figure 5.3: Cumulative delayed data of the video stream

Figure 5.3 shows the impact of retransmission by indicating the video stream’s cumulative delayed data as time progresses. Each step that appears in this timeseries graph occurs due to a congestion event in the unstable link. As the video stream progresses, the gap between using retransmission and omitting retransmissions grows.

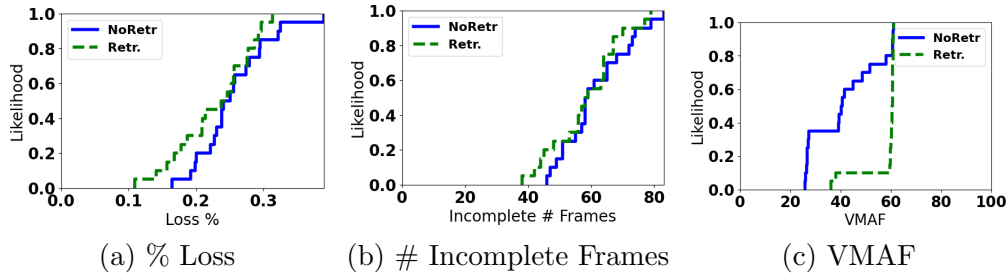


Figure 5.4: Comparison of Retransmission and No Retransmission

We repeat the test 20 times and show the resulting CDF of the delayed data (loss) percentage (Figure 5.4a), number of incomplete frames (Figure 5.4b) and resulting VMAF (Figure 5.4c). We find that using retransmission can improve the median video quality from 40 to 60 by reducing the number of incomplete frames and delayed data percentage. This illustrates that retransmission is beneficial even in network environments that have underlying ARQ mechanisms.

5.3.2 Recovery Mechanisms in LTE Workloads

We first show that Conflux+Rec can lower the percentage of incomplete frames by using FEC. FEC is effective even when application-level packet loss does not occur due to link-layer retransmission mechanisms that conceal channel errors. Instead of causing an application to experience loss, these channel errors are manifested as drops in available bandwidth and increases in latency which may lead to delayed video data. The delayed video data can result in visual artifacts that increase a viewer’s perceptual annoyance, which is measured as a mean annoyance value [126].

We measure the impact of Conflux+Rec in network environment that emulates channel errors. We use traces from the Mei et al. [97] for bandwidth measurements; however, since these traces do not include latency measurements, we extend them with latency measurements to evaluate the impact of variable latency, which we refer to as jitter. These links experience jitter from 20ms to one of 40ms, 80ms, 160ms, and 320ms every 30 seconds

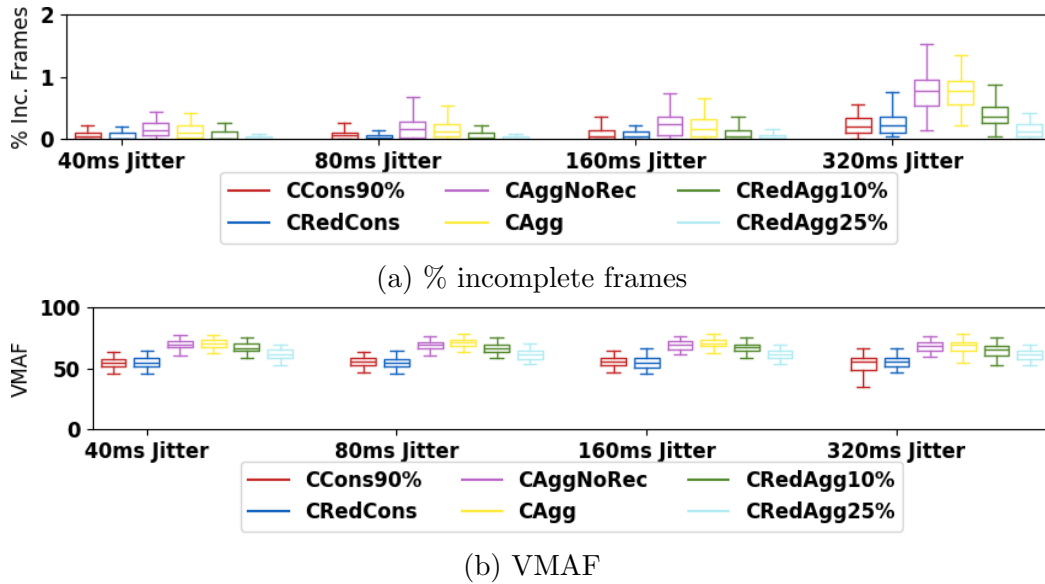


Figure 5.5: Available bandwidth follows Mei et al. workload and jitter is introduced every 30 seconds for 1 - 2 seconds

for 1 - 2 seconds following a Poisson distribution. Because signal loss is often accompanied by a reduction in available bandwidth [42, 19], we also reduce the available bandwidth by half when latency increases.

Figure 5.5a shows a boxplot of the percentage of incomplete frames for 50 test cases. This figure shows that using 25% redundant data (CRedAgg25%) provides the lowest median percentage (0.13%) of incomplete frames over all settings followed by the conservative User Preferences, CCons90% (0.22%) and CRedCons (0.23%). This illustrates that redundancy is effective at protecting the video stream from incomplete frames that result in visual artifacts. This is because if a link experiences quality degradation, redundant packets containing video data may still arrive on-time on other links. Conflux’s conservative User Preferences (CCons90% and CRedCons) are also less susceptible to delays that cause incomplete frames than the aggressive User Preferences (CAggNoRec and CAgg have 0.77% and 0.77% incomplete frames) because they do not saturate the available links. Thus, this result shows that sending at lower rates is also an option for cost-sensitive users who wish to avoid visual artifacts. However, sending at a lower rate forgoes additional video quality when compared to using FEC in conjunction with an aggressive User Preference.

Figure 5.5b shows that saturating the link and using redundancy is preferable to sending less data when video quality is considered. This figure shows that CRedAgg25% provides

higher median VMAF than CCons90% and CRedCons. CRedAgg25% provides between 61-62 VMAF, and the conservative User Preferences have a median VMAF of 55 in all settings. Therefore, users who have the available bandwidth budget should fully utilize the link and send redundant data (CRedAgg25%) to minimize the percentage of incomplete frames while obtaining higher video quality than the conservative User Preferences (CCons90% and CRedCons).

The trade-off in video quality from using redundancy is also shown in Figure 5.5b where we compare using different redundancy levels when aggressively sending data on the available links (CAggNoRec, CAgg, CRedAgg10%, CRedAgg25%). As expected, using redundancy lowers video quality because the available bandwidth is used for sending redundant data instead of additional video data from selecting a higher bitrate. Using redundancy reduces the median VMAF from 69-70 (CAgg) to 65-67 (CRedAgg10%) and 61-62 (CRedAgg25%). Nonetheless, some users may prefer lower video bitrates over visual artifacts [126]. These users should select CRedAgg10% or CRedAgg25% to avoid visual artifacts.

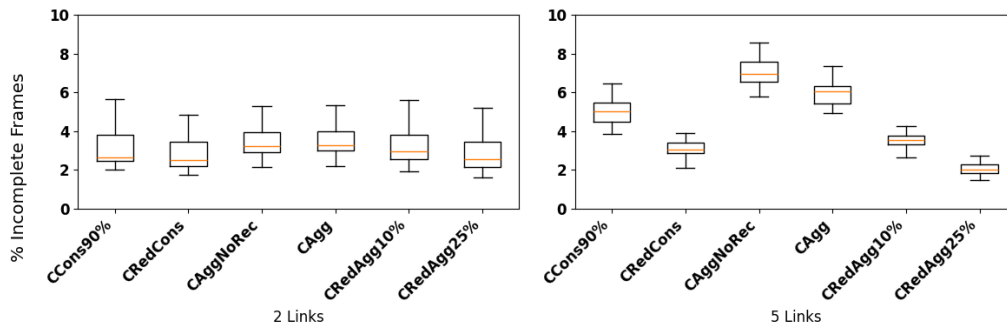


Figure 5.6: % incomplete frames in workload with significant drops in available bandwidth inspired by Fouladi et al. [46]

We find that the aforementioned differences between our presented User Preferences are more pronounced in workloads that exhibit significant drops in available bandwidth which emulate temporary, extreme link failure. We evaluate Conflux+Rec using a workload that is inspired by Fouladi et al. [46] that emulates this environment. In this environment, the available bandwidth temporarily drops to 0.1Kbps for up to 1s (using our Mei et al. [97] traces) every 30s (Poisson distributed).

Figure 5.6 shows the percentage of incomplete frames for our different recovery mechanisms for the two-link and five-link environments. As the number of available links increase, the probability of at least one link failure also increases. Nonetheless, having more links

(five links) is advantageous over having fewer links (two links) because a single link failure has a smaller impact on the total aggregate available bandwidth for video streaming when there are more links. Furthermore, using more links provides more alternate available paths for the video stream when a link fails. Since there is only one alternate path and since a large percentage of the aggregate available bandwidth is reduced when a link fails in the two link environment, there are limited benefits to using FEC.

In the five-link environment, using No Recovery results in 7% incomplete frames, and adding ARQ reduces this to 6% (CAgg). Including redundancy further lowers the percentage of incomplete frames to 4% and 2% for 10% and 25% redundancy respectively. Therefore, users who wish to avoid visual artifacts that cause an increase in the user’s mean annoyance value would benefit from using more redundant data (i.e., CRedAgg25%).

5.3.3 Environments with Application-Level Packet Loss

Although Conflux and Conflux+Rec are designed for environments that do not exhibit application-level packet loss such as LTE, they can also be used in network environments that do not have underlying link-layer retransmission. In these more general network environments, packets can be dropped at any point between the sender and the receiver, and dropped packets are not retransmitted by any other component in the network. Consequently, applications using these networks experience packet loss due to dropped packets.

We made further changes to make Conflux more robust in these general network environments. The primary change is including unacknowledged packet sequence numbers in the Extended Acknowledgment Message so that dropped packets are retransmitted to reduce the amount of delayed or missing video data the on-site video streaming application experiences. We refer to this updated version as *Loss-Aware Conflux*.

We evaluate Loss-Aware Conflux in network environments that exhibit packet loss. Our network environment also follows the bandwidth measurements of Mei et al. and we use a one-way delay of 20ms. Because the Mei et al., environment does not include packet loss rates due to LTE’s underlying ARQ mechanisms, we experiment with various packet loss models that we specify in each experiment.

We show that Loss-Aware Conflux is resilient to application-level packet loss and can provide good perception to users in a variety of loss models. We find that specifying a minimum amount of redundancy on top of aggressively sending video data can provide a low percentage of incomplete frames. In the model presented by Bocharova et al. [19], loss can occur whether or not the link is in a good (5% loss) or bad state (30%). Figure 5.7 shows the percentage of incomplete frames and that using a redundant User Preference can

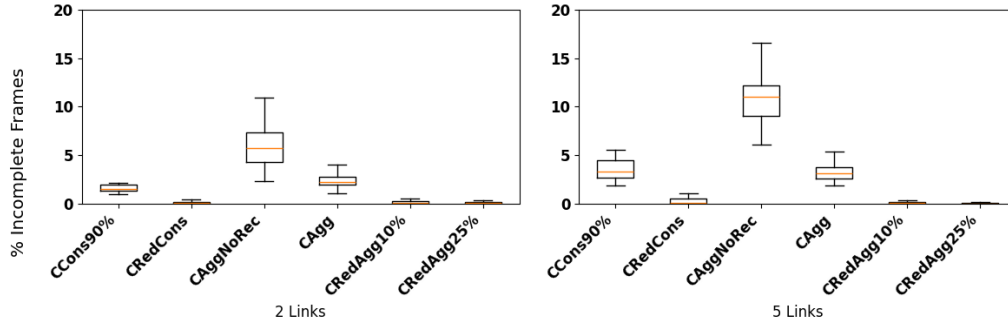


Figure 5.7: % incomplete frames for Loss-Aware Conflux in the Bocharova et al. [19] packet loss model

significantly lower this percentage. In the five-link environment, we can reduce the median percentage of incomplete frames from 11% (CAggNoRec - no recovery mechanisms) to 0.02% (CRedAgg25%). We find similar trends between our presented User Preferences in the Feng et al. [44] loss environment which is shown in Figure 5.8.

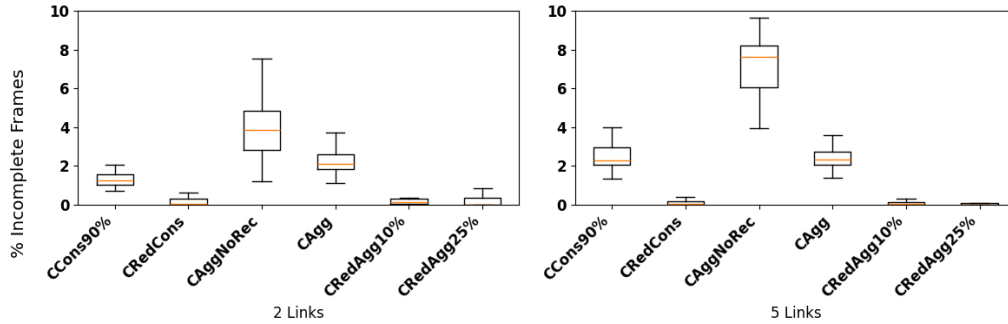


Figure 5.8: % incomplete frames for Loss-Aware Conflux in the Feng et al. [44] loss model

Although packet loss in wireless networks is often modeled using the Gilbert-Elliot loss model, we also evaluate Loss-Aware Conflux in the presence of random packet loss [54, 33, 45] to illustrate Loss-Aware Conflux’s robustness in a variety of network environments. Figure 5.9b shows the achieved video quality of our user preferences as the loss percentage increases from 0%, to 0-1%, 1-2%, and 2-5%. This figure shows that enforcing a minimum level of redundancy results in a low percentage of incomplete frames. The median percentage of incomplete frames is between 0.0% - 0.3% for CRedAgg10% and is up to 59% for CAgg. Our results show that enforcing redundancy in network environments that drop

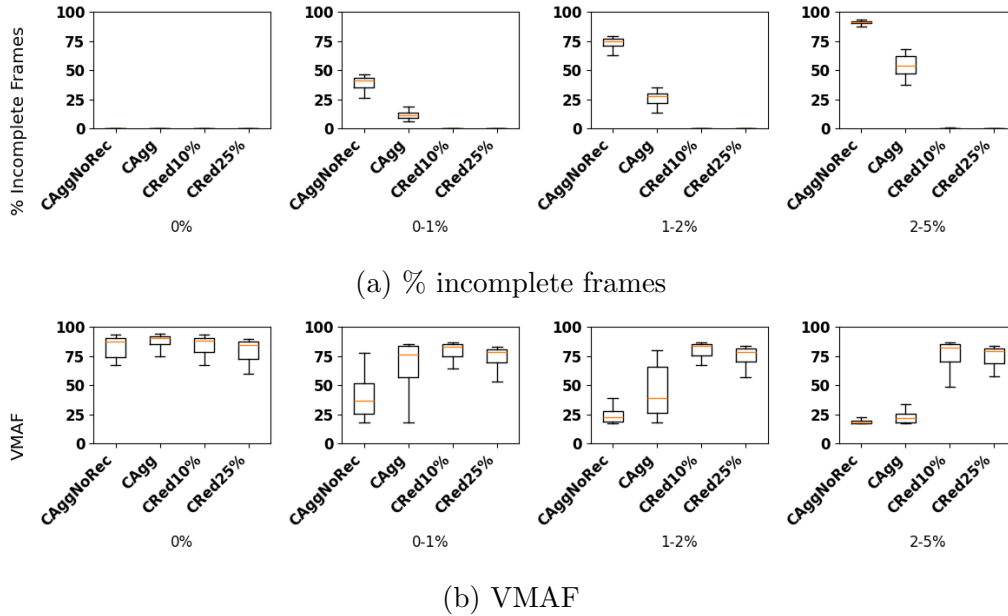


Figure 5.9: Loss-Aware Conflux in a random loss environment

packets is effective at preventing lost data, and it is necessary because our PSuccess model does not capture packet loss. Furthermore, we find that CAgg is unable to further reduce its percentage of incomplete frames because there is often insufficient available bandwidth for retransmission after sending video data, which we discussed in Section 5.3.1.

Conflux’s Redundant User Preferences also achieve high video quality in environments that drop packets. Figure 5.9b shows that CRedAgg10% and CRedAgg25% achieve between 78–88 VMAF regardless of the percentage loss on each link. This is contrasted with CAggNoRec and CAgg whose median video quality degrade from 88 and 90 (VMAF) to 18 and 22 (VMAF) respectively when the percentage loss on each link increases to 2–5%.

In summary, FEC is beneficial in reducing the percentage of incomplete frames in all network environments, regardless of the presence of underlying link-layer retransmission. Loss-Aware Conflux is robust in these environments with the use of Redundant User Preferences that specify at least 10% redundancy, and it can achieve good user perception even in challenging network environments.

5.4 Chapter Summary

In this chapter, we presented retransmission and FEC extensions to Conflux, thereby introducing Conflux+Rec and Loss-Aware Conflux. These extensions enabled Conflux to be robust in a variety of network environments which it was not originally designed for.

These extensions required changes to Conflux’s Configuration Finder to evaluate the probability that the video data can be reconstructed. This was accomplished by modelling the probability of on-time arrival for each packet so that the Configuration Finder can calculate the probability that at least k of n packets arrive. This extension introduces the ability for Conflux’s users to specify a minimum amount of redundancy based on their own preferences. Furthermore, the Conflux’s Packet Assignment Module was extended to retransmit packets that have not been acknowledged within four round-trip times. Finally, we introduced negative acknowledgments in the extended acknowledgment message to create Loss-Aware Conflux to provide better support when streaming in lossy network environments. Because of Conflux’s modular design, the required changes were localized to their respective modules.

Our evaluation showed the limitations of using retransmission in the Conflux framework and presented the results of an extension that prioritized retransmission and showed that it is able to reduce the percentage of incomplete frames and improve video quality. Furthermore, we showed that specifying a minimum level of redundancy in Conflux+Rec’s User Preferences can protect the video stream against delayed video data in challenging network environments. Finally, we showed that Loss-Aware Conflux can also obtain good video quality in network environments that do not retransmit delayed data by using a user preference that uses at least 10% redundancy.

Chapter 6

Conclusion

On-site live video streaming for news casting is an important application that many use and benefit from daily; thus, providing high user QoE is important. This application must provide high quality and timely video to achieve high user QoE, and multi-homing and video bitrate adaptation are techniques that enable this application to achieve these goals. This is because multi-homing can enable applications to stream at high video bitrates through bandwidth aggregation. Furthermore, it can also offer lower latency by using alternate, non-congested paths. Video bitrate adaptation can also improve user QoE by adapting the video bitrate to changing link conditions so that the video stream does not experience delays due to congestion. This is especially important when the expected deployment environment (i.e., wireless LTE) experiences fluctuating available bandwidth.

Solving the multi-homed, adaptive live streaming problem is challenging due to link heterogeneity; poor choices in multi-homed packet scheduling and video bitrate adaptation may result in worse video quality than if a single link was used. There are many subproblems and research areas to consider when solving this challenge, and these include: (1) accurate bandwidth measurement, (2) link quality prediction, (3) multi-homed scheduling, and (4) live video bitrate adaptation. This thesis addressed each of these challenges and presented a modular approach to solving the multi-homed, adaptive live video streaming for news casting problem.

6.1 Contributions

Firstly, this thesis introduced the PacketBurst bandwidth estimation technique that removes the inaccuracies in packet-train bandwidth estimations caused by kernel interrupt

scheduling. This technique identifies and removes the affected packets from the bandwidth estimation calculation. When using six MTU-sized packets to estimate available bandwidth, the PacketBurst technique experiences an 2.75% error whereas packet trains experience a 21.6% error when the available bandwidth is 20Mbps. This contribution enables applications to obtain accurate bandwidth estimates from short packet trains and improves all packet-dispersion bandwidth estimation techniques. This is important when the application’s logic relies on bandwidth estimates.

Secondly, we presented our framework and technique for using Machine Learning (ML) to predict link quality. We used traces collected by our industrial partner to identify attributes that best predict link instability, which we defined as a significant increase in latency, or drop in available bandwidth or packet arrival. Using these traces, we identified periods of instability and constructed examples to train and evaluate many statistical ML models. We have found that using decision trees and k-nearest neighbour techniques predicted at least 40% of link instability events. This contribution is important because it demonstrated the possibility of using statistical ML models to predict link instability. Our work differentiates itself from others as it goes beyond predicting the available bandwidth and focuses on events that can have detrimental impacts on the video stream. The result of this work can be used by multi-homed transport protocols to avoid unstable links and by adaptive video bitrate protocols to lower the video bitrate.

Thirdly, we presented Conflux: an adaptive video bitrate protocol for on-site live video streaming in the multi-homed environment. We introduced a modular design that separates link quality characterization from multi-homed scheduling and video bitrate adaptation. This allowed Conflux to use an arbitrary number and different types of links, and to cater to a variety of user preferences. We characterized link quality as a PSuccess function that maps a sending rate to the probability that the link is able to deliver data on time. The probabilistic model introduced by PSuccess characterized the trade-offs between incurring congestion and obtaining a higher video bitrate. The impact of Conflux’s design and implementation resulted in its ability to achieve higher video quality than its competitors. Our evaluation showed that Conflux achieved at least a 13% improvement in video quality against the MPTCP comparison system despite MPTCP having knowledge of the total aggregate available bandwidth.

Finally, we introduced recovery mechanisms such as retransmission and redundancy (i.e., FEC) to Conflux. We found that the benefits of retransmission are limited since there is often insufficient available bandwidth for both new video data and retransmitted data when limiting the sending rate on each link according to the user’s preference. We demonstrated that using redundancy improved on-time video data arrival at the cost of using additional bandwidth even when there are underlying retransmission mechanisms.

To accomplish this, we characterized the expected improvement by using the `PSuccess` function to model the probability of on-time arrival for each individual packet which is then used to estimate the probability that the video data can be reconstructed. This allowed Conflux to quantify the trade-offs of using redundant data and to select the redundancy level that provides the highest expected user utility.

6.2 Future Work

In this section, we describe future work. We describe our future work in link quality characterization and how we can account for more complex user requirements such as service level agreements. We also outline how to handle unequal frame importance and rate mismatch in the encoder. Finally, we describe extensions to Conflux that explicitly characterize the benefits of reserving available bandwidth for retransmission over using FEC and that include packet loss in the `PSuccess` model.

6.2.1 Directions in Link Quality Characterization

We presented an abstraction that modeled link quality as a function that takes in a sending rate as input and produces the probability that the link can sustain the input sending rate (i.e., the `PSuccess` function). As new types of networks such as 5G and 6G are introduced, there are opportunities to model these new networks using our probability abstraction. There are also many opportunities to develop the `PSuccess` function such as using bandwidth measurements, link instability predictions, or capturing temporal trends such as the expected rate of link instability events.

Although 5G network providers may provide network slicing to guarantee resources (e.g., available bandwidth) for users within the mobile core, network slicing does not necessarily guarantee the available bandwidth between the user and the cell tower (gNodeB), which may be the bottleneck link within the 5G. This is because the link between the user and gNodeB may be affected by physical factors. Therefore, it is still beneficial to model link instability so that an adaptive video bitrate protocol has a signal to lower the video bitrate when a reduction in the available bandwidth cannot be addressed by network slicing.

We also showed that machine learning can be used to predict sudden increases in latency, application-level packet loss, and drops in available bandwidth. These predictions can also be used to update the `PSuccess` function. Future work includes determining how

to use these classifications within the PSuccess function. For example, we envision a hybrid method that uses these classifications in conjunction with a congestion signal to determine the range of affected sending rates in the PSuccess function and the magnitude of the change in the probability of success.

Improvements to the PSuccess function will result in Conflux being able to make better trade-offs in terms of the sending rate and redundancy levels to use.

6.2.2 Development of User Preferences

Over the duration of the video stream, a user may want to make different trade-offs. For example, if the user has experienced many delayed frames already, it may prefer to send at a lower rate than what it was originally sending at to ensure that the viewer does not continue to experience visual artifacts. One solution to capture this preference is to introduce a state-based design where the utility function changes depending on the current, achieved video stream. This type of User Preference may be desirable for users who have specific service-level agreements (e.g., maximum number of artifacts per minute). This direction of video bitrate adaptation requires the development of an abstraction or framework that allows users to translate their service level agreements into the corresponding state machine or user preferences.

6.2.3 Frame Awareness

In this thesis, we did not differentiate frame types as we used intra-refresh encoding which results in I frames being amortized over multiple frames. Consequently, each frame was roughly of equal importance and equal size. However, users may select different encoding settings depending on their use case. Consequently, one future research direction is to address unequal frame importance. One solution is the schedule high priority packets or frames on links with the PSuccess function that indicates greater stability or to use redundancy for only the high priority packets. Within the Conflux framework, this could be addressed with utility functions / User Preferences that are specific to the frame type.

6.2.4 Handling Rate Mismatch in the Encoder

This thesis illustrated the challenges when using a single-pass encoder in conjunction with rate-based control on all the available links. Our solution to this challenge was to reduce

the video bitrate so that the encoder could reach the target video bitrate more quickly. Future work includes exploring other techniques such as pacing (delay sending additional video packets to avoid congesting links) or sending additional packets immediately and risk congestion.

An alternate method to handle rate mismatch in the encoder is to account for the encoder over-shooting its target bitrate within the Conflux model. In Conflux, we presented *PVideoFrameArrival* which is defined as the probability that the generated video frame arrives given the links' PSuccess function, the target bitrate, and the coding rate. We can integrate the encoder's rate mismatch in the probability that the generated video frame arrives on time, and this would model the probability of the encoder either over-shooting or under-shooting. However, predicting encoder behaviour is currently an open problem for single-pass encoding [99]. Thus, more research in this area needs to be completed prior to integrating encoder inaccuracies into Conflux.

6.2.5 Explicit Considerations for Retransmission and Packet Loss

Conflux+Rec and Loss-Aware Conflux did not explicitly consider the trade-offs for reserving available bandwidth for packet retransmission or FEC to protect against packet drops. Future work involves developing the User Preferences to capture the benefits of reserving available bandwidth for retransmission. Moreover, we showed that Loss-Aware Conflux was able to effectively stream video in network environments that do not have link-layer retransmission. Loss-Aware Conflux accomplished this with its retransmission and FEC recovery mechanisms, and did not build in packet loss into the PSuccess function. Future work includes capturing packet loss rates in PSuccess so that we can accurately calculate the probability that the video frame can be reconstructed at the receiver and determine the exact amount of redundancy to use when there is packet loss.

6.3 Concluding Remarks

In conclusion, multi-homed, adaptive on-site live video streaming for news casting is an important application that many depend on to receive news in order to be informed and make decisions. The challenges found in this application are unique and different from other video streaming applications such as video conferencing or on-demand video; and unfortunately, this application has largely been overlooked by the research community. This thesis presented solutions to these challenges, and this is especially important as

bandwidth demands for mobile streaming continue to increase such that they cannot be satisfied by a single link.

This thesis presented techniques to improve bandwidth estimation, link quality forecasting, and a framework for reasoning about the various trade-offs that a protocol can make when streaming adaptive, latency-sensitive video in the multi-homed environment. Furthermore, this thesis has demonstrated the viability of bitrate adaptation in the multi-homed environment for latency-sensitive data. As our presented system is not constrained to predetermined types of network links, number of links, or user requirements, we hope that it can be used as a foundation for designing multi-homed and adaptive transport protocols for latency-sensitive data. The modular design of our system encourages the adoption of new ideas that focus on the individual challenges pertaining to multi-homed, adaptive live streaming such as link quality characterization, video bitrate adaptation, and redundancy policies. This is important as networks evolve and as applications become more demanding as only individual components need to be redesigned.

The effective use of multiple, heterogeneous resources is challenging, especially when the inclusion of one resource can lead to worse performance. Therefore, it is important to have useful abstractions that enable a protocol to decide how to use its available resources or if a resource should even be used. This is particularly important when additional resources enable new, alternative solutions to our problem. For example, multi-homing introduces the possibility of replicating the video stream over multiple links, which was not possible in the single-homed environment. Having a suitable abstraction, such as probability, enables the creation of a general protocol that can evaluate the utility of different decisions. General protocols are advantageous as they do not need to be redesigned when the characteristics of a resource changes or when additional resources are added, thereby making them scalable. With the introduction of our probability abstraction and our video streaming model, we hope that the work in this thesis encourages the design and development of useful abstractions to create general and scalable protocols.

References

- [1] 4G. wikipedia.org. Accessed: 10 July 2025. [Online.] Available: <https://en.wikipedia.org/wiki/4G/>.
- [2] CBR/ABR/VBR: the 3 encoding mode. cd.textfiles.com. Accessed: 5 June 2025. [Online.] Available: <http://cd.textfiles.com/amigaplus/lesercd16/Sound/Lame-3.91/doc/html/modes.html>.
- [3] Control interfaces. kernel.org. Accessed: 10 July 2025. [Online.] Available: <https://www.kernel.org/doc/Documentation/networking/timestamping.txt>.
- [4] Handling of Frame Drops Due to Packet Loss in VMAF Evaluation. github.com. Accessed: 10 July 2025. [Online.] Available: <https://github.com/Netflix/vmaf/issues/1366>.
- [5] Internet Speed Comparison Chart — What’s a Good Internet Speed? electronicshub.org. Accessed: 10 July 2025. [Online.] Available: <https://www.electronicshub.org/internet-speed-comparison-chart/>.
- [6] LTE eNodeB Scheduler. techplayon.com. Accessed: 5 June 2025. [Online.] Available: <https://www.techplayon.com/lte-enodeb-scheduler-and-different-scheduler-type/>.
- [7] Ahmed Abd, Tarek Saadawi, Myung Lee, et al. LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol. *Computer Communications*, 27(10):1012–1024, 2004.
- [8] Najah Abu-Ali, Abd-Elhamid M Taha, Mohamed Salah, and Hossam Hassanein. Up-link scheduling in LTE and LTE-advanced: Tutorial, survey and evaluation framework. *IEEE Communications surveys & tutorials*, 16(3):1239–1265, 2013.

- [9] Muninder Adavelli. How Many Videos Are Uploaded to YouTube Every Day in 2025? techjury.net. Accessed: 10 July 2025. [Online.] Available: <https://techjury.net/blog/live-streaming-statistics/#gref>.
- [10] Samira Afzal, Vanessa Testoni, Christian Esteve Rothenberg, Prakash Kolan, and Imed Bouazizi. A holistic survey of wireless multipath video streaming. *arXiv preprint arXiv:1906.06184*, 2019.
- [11] Mark Akselrod, Nico Becker, Markus Fidler, and Ralf Lübben. 4G LTE on the road-what impacts download speeds most? In *Proceedings of the 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–6. IEEE, 2017.
- [12] Taner Arsan. Review of bandwidth estimation tools and application to bandwidth adaptive video streaming. In *Proceedings of High Capacity Optical Networks and Emerging/Enabling Technologies*, pages 152–156. IEEE, 2012.
- [13] Venkat Arun and Hari Balakrishnan. Copa: Practical delay-based congestion control for the internet. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation*, pages 329–342. USENIX, 2018.
- [14] Arjun Balasingam, Manu Bansal, Rakesh Misra, Kanthi Nagaraj, Rahul Tandra, Sachin Katti, and Aaron Schulman. Detecting if LTE is the bottleneck with Burst-Tracker. In *Proceedings of the 25th Annual International Conference on Mobile Computing and Networking*, pages 1–15. ACM, 2019.
- [15] Nico Becker, Amr Rizk, and Markus Fidler. A measurement study on the application-level performance of LTE. In *Proceedings of the 2014 IFIP Networking Conference*, pages 1–9. IEEE, 2014.
- [16] Abdelhak Bentaleb, May Lim, Mehmet N Akcay, Ali C Begen, Sarra Hammoudi, and Roger Zimmermann. Toward one-second latency: Evolution of live media streaming. *IEEE Communications Surveys & Tutorials*, 2025.
- [17] Abdelhak Bentaleb, Christian Timmerer, Ali C Begen, and Roger Zimmermann. Bandwidth prediction in low-latency chunked streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 7–13. ACM, 2019.
- [18] Francesco Beritelli, Giacomo Capizzi, Grazia Lo Sciuto, Christian Napoli, and Francesco Scaglione. Rainfall estimation based on the intensity of the received signal

in a LTE/4G mobile terminal by using a probabilistic neural network. *IEEE Access*, 6:30865–30873, 2018.

- [19] Irina Bocharova, Boris Kudryashov, Maben Rabi, Nikita Lyamin, Wouter Dankers, Erik Frick, and Alexey Vinel. Characterizing packet losses in vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(9):8347–8358, 2019.
- [20] Yuanlong Cao, Qinghua Liu, Guoliang Luo, Yugen Yi, and Minghe Huang. PR-MPTCP+: Context-aware QoE-oriented multipath TCP partial reliability extension for real-time multimedia applications. In *Proceedings of the 2016 Visual Communications and Image Processing*, pages 1–4. IEEE, 2016.
- [21] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and design of the Google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–12. ACM, 2016.
- [22] Robert L Carter and Mark E Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical report, Citeseer, 1996. Available: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=edd3cf31fd1914e9bbe060c8713cabe02fa5a4a1>.
- [23] C Casetti and W Gaiotto. Westwood SCTP: Load balancing over multipaths using bandwidth-aware source scheduling. In *Proceedings of the IEEE 60th Vehicular Technology Conference*, volume 4, pages 3025–3029. IEEE, 2004.
- [24] Liane Cassavoy. How Fast Is 4G LTE Wireless Service. lifewire.com. Accessed: 10 July 2025. [Online.] Available: <https://www.lifewire.com/how-fast-is-4g-wireless-service-577566>.
- [25] Gregory Cermak, Margaret Pinson, and Stephen Wolf. The relationship among video quality, screen resolution, and bit rate. *IEEE Transactions on Broadcasting*, 57(2):258–262, 2011.
- [26] Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee. A tale of three videoconferencing applications: Zoom, webex, and meet. *IEEE/ACM Transactions on Networking*, 2022.
- [27] Sharon Choy, Joochan Lee, Bernard Wong, and Khuzaima Daudjee. Conflux: A multi-homed adaptive bitrate protocol for on-site live video streaming. *IEEE Transactions on Networking*, 2025. © 2025 IEEE. Reprinted with permission.

- [28] Sharon Choy and Bernard Wong. Evaluating machine learning techniques for predicting link instability in wireless networks to support live video streaming. In *2023 19th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 161–168. IEEE, 2023. © 2023 IEEE. Reprinted with permission.
- [29] Sharon Choy and Bernard Wong. Obtaining accurate bandwidth estimations for the internet of things. In *2023 6th Conference on Cloud and Internet of Things (CIoT)*, pages 104–111. IEEE, 2023. © 2023 IEEE. Reprinted with permission.
- [30] Carlos Cortés, Pablo Pérez, Jesús Gutiérrez, and Narciso García. Influence of video delay on quality, presence, and sickness in viewport adaptive immersive streaming. In *Proceedings of the 12th International Conference on Quality of Multimedia Experience*, pages 1–4. IEEE, 2020.
- [31] Dash Industry Forum. Report on Low Latency DASH. dashif.org. Accessed: 5 June 2025. [Online.] Available: <https://dashif.org/docs/Report%20on%20Low%20Latency%20DASH.pdf>.
- [32] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Skype video responsiveness to bandwidth variations. In *Proceedings of the 18th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 81–86. ACM, 2008.
- [33] Quentin De Coninck and Olivier Bonaventure. Multipath QUIC: Design and evaluation. In *Proceedings of the 13th Conference on Emerging Networking EXperiments and Technologies*, pages 160–166. ACM, 2017.
- [34] Dejero. EnGo 3. dejero.com. Accessed: 5 June 2025. [Online.] Available: <https://www.dejero.com/products>.
- [35] Dejero. Products. dejero.com. Accessed: 5 June 2025. [Online.] Available: <https://www.dejero.com/products>.
- [36] Deutsche Telekom AG. Multipath DCCP. multipath-dccp.org. Accessed: 10 July 2025. [Online.] Available: <https://multipath-dccp.org/references.html/>.
- [37] Scikit Learn Developers. sklearn.feature_selection.selectkbest. scikit-learn.org. Accessed: 10 July 2025. [Online.] Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html.

- [38] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Converge: QoE-driven multipath video conferencing over WebRTC. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 637–653. ACM, 2023.
- [39] Codé Diop, Guillaume Dugué, Christophe Chassot, and Ernesto Exposito. QoS-oriented MPTCP extensions for multimedia multi-homed systems. In *The 26th International Conference on Advanced Information Networking and Applications Workshops*, pages 1119–1124. IEEE, 2012.
- [40] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure? In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Society*, volume 2, pages 905–914. IEEE, 2001.
- [41] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions on Networking*, 12(6):963–977, 2004.
- [42] Ayman Elnashar and Mohamed A El-Saidny. Looking at LTE in practice: A performance analysis of the LTE system based on field test results. *IEEE Vehicular Technology Magazine*, 8(3):81–92, 2013.
- [43] Epitome. Bandwidth Aggregation Software. epitomesolutions.in. Accessed: 5 June 2025. [Online.] Available: https://www.epitomesolutions.in/bandwidth_aggregation_software.php.
- [44] Jingyun Feng, Zhi Liu, and Yusheng Ji. Wireless channel loss analysis-a case study using wifi-direct. In *Proceedings of the 2014 International Wireless Communications and Mobile Computing Conference*, pages 244–249. IEEE, 2014.
- [45] Simone Ferlin, Özgü Alay, Olivier Mehani, and Roksana Boreli. BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks. In *Proceedings of the 2016 IFIP Networking Conference*, pages 431–439. IEEE, 2016.
- [46] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation*, pages 267–282. USENIX, 2018.

- [47] Aditya Ganjam, Faisal Siddiqui, Jibin Zhan, Xi Liu, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. C3: Internet-scale control plane for video quality optimization. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation*, volume 15, pages 131–144. USENIX, 2015.
- [48] Johan Garcia, Stefan Alfredsson, and Anna Brunstrom. Delay metrics and delay characteristics: A study of four Swedish HSDPA+ and LTE networks. In *Proceedings of the 2015 European Conference on Networks and Communications*, pages 234–238. IEEE, 2015.
- [49] Frode Guribye and Lars Nyre. The changing ecology of tools for live news reporting. *Journalism Practice*, 11(10):1216–1230, 2017.
- [50] Haivision. Haivision Air. haivision.com. Accessed: 5 June 2025. [Online.] Available: <https://www.haivision.com/products/air-mobile-transmitter/>.
- [51] Haivision. MoJoPro. haivision.com. Accessed: 5 June 2025. [Online.] Available: <https://www.aviwest.com/products/mojopro-series/>.
- [52] Haivision. MoJoPro. haivision.com. Accessed: 10 July 2025. [Online.] Available: <https://iperf.fr/>.
- [53] Bo Han, Feng Qian, Lusheng Ji, Vijay Gopalakrishnan, and NJ Bedminster. MP-DASH: Adaptive video streaming over preference-aware multipath. In *Proceedings of the 12th Conference on Emerging Networking EXperiments and Technologies*, pages 129–143. ACM, 2016.
- [54] Jiangping Han, Kaiping Xue, Yitao Xing, Peilin Hong, and David SL Wei. Measurement and redesign of BBR-based MPTCP. In *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, pages 75–77. ACM, 2019.
- [55] Mahdi Hemmati, Bill McCormick, and Shervin Shirmohammadi. QoE-aware bandwidth allocation for video traffic using sigmoidal programming. *IEEE MultiMedia*, 24(4):80–90, 2017.
- [56] Christian Herglotz, André Kaup, Stéphane Coulombe, and Ahmad Vakili. Power-efficient video streaming on mobile devices using optimal spatial scaling. In *Proceedings of the 2019 IEEE 9th International Conference on Consumer Electronics*, pages 233–238. IEEE, 2019.

- [57] Ningning Hu and Peter Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, 2003.
- [58] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, pages 225–238. ACM, 2012.
- [59] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. An in-depth study of LTE: effect of network protocol and application behavior on performance. *ACM SIGCOMM Computer Communication Review*, 43(4):363–374, 2013.
- [60] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *ACM SIGCOMM Computer Communication Review*, 44(4):187–198, 2015.
- [61] C. Huitema. Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP). datatracker.ietf.org. Accessed: 10 July 2025. [Online.] Available: <https://datatracker.ietf.org/doc/html/rfc3605>.
- [62] Per Hurtig, Karl-Johan Grinnemo, Anna Brunstrom, Simone Ferlin, Özgü Alay, and Nicolas Kuhn. Low-latency scheduling in MPTCP. *IEEE/ACM Transactions on Networking*, 27(1):302–315, 2018.
- [63] IBM. Internet connection and recommended encoding settings. support.video.ibm.com. Accessed: 10 July 2025. [Online.] Available: <https://support.video.ibm.com/hc/en-us/articles/207852117-Internet-connection-and-recommended-encoding-settings>.
- [64] Telesystem Innovations. LTE in a Nutshell: Protocol Architecture. frankrayal.com. Accessed: 10 July 2025. [Online.] Available: <https://frankrayal.com/wp-content/uploads/2017/02/LTE-in-a-Nutshell-Protocol-Architecture.pdf>.
- [65] Janardhan R Iyengar, Paul D Amer, and Randall Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006.

- [66] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *Proceedings of the 2002 Passive and Active Measurements Workshop*. Citeseer, 2002.
- [67] Cyriac James, Emir Halepovic, Mea Wang, Rittwik Jana, and NK Shankaranarayanan. Is Multipath TCP (MPTCP) Beneficial for Video Streaming over DASH? In *Proceedings of the 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 331–336. IEEE, 2016.
- [68] Haiqing Jiang, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Tackling bufferbloat in 3G/4G networks. In *Proceedings of the 2012 Internet Measurement Conference*, pages 329–342. ACM, 2012.
- [69] Junchen Jiang, Rajdeep Das, Ganesh Ananthanarayanan, Philip A Chou, Venkata Padmanabhan, Vyas Sekar, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, Renat Vafin, et al. Via: Improving internet telephony call quality using predictive relay selection. In *Proceedings of the ACM SIGCOMM 2016 Conference*, pages 286–299. ACM, 2016.
- [70] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. CFA: A practical prediction system for video QoE optimization. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*, pages 137–150. USENIX, 2016.
- [71] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive. In *Proceedings of the 8th Conference on Emerging Networking EXperiments and Technologies*, pages 97–108. ACM, 2012.
- [72] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation. In *14th USENIX Symposium on Networked Systems Design and Implementation*, pages 393–406. USENIX, 2017.
- [73] Minqiang Jiang and Nam Ling. Low-delay rate control for real-time H.264/AVC video coding. *IEEE Transactions on Multimedia*, 8(3):467–477, 2006.
- [74] I Johansson and M Westerlund. Self-Clocked Rate Adaptation for Multimedia. ietf.org. Accessed: 5 June 2025. [Online.] Available: <https://www.ietf.org/archive/id/draft-johansson-ccwg-rfc8298bis-screamv2-00.html>.

- [75] The kernel development community. Timestamping. docs.kernel.org. Accessed: 10 July 2025. [Online.] Available: <https://docs.kernel.org/networking/timestamping.html>.
- [76] Srinivasan Keshav. Packet-pair flow control. *IEEE/ACM Transactions on Networking*, pages 1–45, 1995.
- [77] Ramin Khalili, Nicolas Gast, Miroslav Popovic, and Jean-Yves Le Boudec. MPTCP is not Pareto-optimal: Performance issues and a possible solution. *IEEE/ACM Transactions On Networking*, 21(5):1651–1665, 2013.
- [78] Hendrik Knoche, Hermann G De Meer, and David Kirsh. Utility curves: Mean opinion scores considered biased. In *Proceedings of the 7th International Workshop on Quality of Service*, pages 12–14. IEEE, 1999.
- [79] Alexander Kruglov. Interpretation of objective video quality metrics. elecard.com. Accessed: 5 June 2025. [Online.] Available: https://www.elecard.com/page/article_interpretation_of_metrics.
- [80] Nicolas Kuhn, Emmanuel Lochin, Ahlem Mifdaoui, Golam Sarwar, Olivier Mehani, and Roksana Boreli. DAPS: Intelligent delay-aware packet scheduling for multipath transport. In *Proceedings of the 2014 IEEE International Conference on Communications*, pages 1222–1227. IEEE, 2014.
- [81] Kevin Lai. Packet Pair Technique. usenix.org. Accessed: 10 July 2025. [Online.] Available: https://www.usenix.org/legacy/publications/library/proceedings/usits01/full_papers/lai/lai_html/node2.html.
- [82] ZG Li, Wen Gao, Feng Pan, SW Ma, Keng Pang Lim, GN Feng, Xiao Lin, Susanto Rahardja, HQ Lu, and Yan Lu. Adaptive rate control for H.264. *Journal of Visual Communication and Image Representation*, 17(2):376–406, 2006.
- [83] Yeon-sup Lim, Erich M Nahum, Don Towsley, and Richard J Gibbens. ECF: An MPTCP path scheduler to manage heterogeneous paths. In *Proceedings of the 13th Conference on Emerging Networking EXperiments and Technologies*, pages 147–159. ACM, 2017.
- [84] DeviceAtlas Limited. Dual sim smartphone usage. <https://deviceatlas.com/blog/dual-sim-smartphone-usage>, 2022. Accessed: 2024-11-27.

- [85] Chang Liu, Jean Tourrilhes, Chen-Nee Chuah, and Puneet Sharma. Voyager: Revisiting available bandwidth estimation with a new class of methods—decreasing-chirp-train methods. *IEEE/ACM Transactions on Networking*, 30(4):1717–1732, 2022.
- [86] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. Rate adaptation for adaptive HTTP streaming. In *Proceedings of the 2nd Multimedia Systems Conference*, pages 169–174. ACM, 2011.
- [87] Tong Liu, Jiangyu Pan, and Ye Tian. Detect the bottleneck of commercial 5G in China. In *Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pages 941–945. IEEE, 2020.
- [88] Zoe Liu. Making Sense of PSNR, SSIM, VMAF. visionular.ai. Accessed: 10 July 2025. [Online.] Available: <https://visionular.ai/vmaf-ssim-psnr-quality-metrics/>.
- [89] LiveU. Live U Solo Pro. liveu.tv. Accessed: 5 June 2025. [Online.] Available: <https://www.liveu.tv/products/create/liveu-solo/>.
- [90] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 2021 Internet Measurement Conference*, pages 229–244. ACM, 2021.
- [91] Sharat Chandra Madanapalli, Alex Mathai, Hassan Habibi Gharakheili, and Vijay Sivaraman. ReCLive: Real-time classification and QoE inference of live video streaming services. In *Proceedings of the 2021 IEEE/ACM 29th International Symposium on Quality of Service*, pages 1–7. IEEE, 2021.
- [92] Imtiaz Mahmud, Tabassum Lubna, and You-Ze Cho. Performance evaluation of MPTCP on simultaneous use of 5G and 4G networks. *Sensors*, 22(19):7509, 2022.
- [93] Tarun Mangla, Nawanol Theera-Ampornpant, Mostafa Ammar, Ellen Zegura, and Saurabh Bagchi. Video through a crystal ball: Effect of bandwidth prediction quality on adaptive streaming in mobile environments. In *Proceedings of the 8th International Workshop on Mobile Video*, pages 1–6. ACM, 2016.
- [94] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the ACM SIGCOMM 2017 Conference*, pages 197–210. ACM, 2017.

- [95] Shiwen Mao, Dennis Bushmitch, Sathya Narayanan, and Shivendra S Panwar. MRTP: a multiframe real-time transport protocol for ad hoc networks. *IEEE Transactions on Multimedia*, 8(2):356–369, 2006.
- [96] John D McCarthy, M Angela Sasse, and Dimitrios Miras. Sharp or smooth? comparing the effects of quantization vs. frame rate for streamed video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 535–542. ACM, 2004.
- [97] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifan Han, Feng Li, and Jin Li. Realtime mobile bandwidth prediction using LSTM neural network and Bayesian fusion. *Computer Networks*, 182:107515, 2020.
- [98] Bob Melander, Mats Bjorkman, and Per Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Proceedings of the IEEE 2000 Global Telecommunications Conference*, volume 1, pages 415–420. IEEE, 2000.
- [99] Loren Merritt and Rahul Vanam. Improved rate control and motion estimation for H.264 encoder. In *Proceedings 2007 IEEE International Conference on Image Processing*, volume 5, pages V–309. IEEE, 2007.
- [100] Mininet Project Contributors. Mininet. mininet.org Accessed: 5 June 2025. [Online.] Available: <http://mininet.org/>.
- [101] Dimitar Minovski, Niclas Ogren, Christer Ahlund, and Karan Mitra. Throughput prediction using machine learning in LTE and 5G networks. *IEEE Transactions on Mobile Computing*, 2021.
- [102] Abubakar Muhammad Miyim and Abubakar Wakili. Performance evaluation of LTE networks. In *Proceedings of the 2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*, pages 1–6. IEEE, 2019.
- [103] Ricky KP Mok, Xiapu Luo, Edmond WW Chan, and Rocky KC Chang. QDASH: a QoE-aware DASH system. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 11–22. ACM, 2012.
- [104] Hyeonjun Na, Yongjoo Shin, Dongwon Lee, and Joohyun Lee. LSTM-based throughput prediction for LTE networks. *ICT Express*, 2021.

- [105] Netflix. Netflix/vmaf - perceptual video quality assessment algorithm. github.com. Accessed: 5 June 2025. [Online.] Available: <https://github.com/Netflix/vmaf>.
- [106] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. Mahimahi: Accurate record-and-replay for http. In *Proceedings of the 2015 USENIX Annual Technical Conference*, pages 417–429. USENIX, 2015.
- [107] K P. How LTE Stuff Works? . howltestuffworks.blogspot.com. Accessed: 5 June 2025. [Online.] Available: <http://howltestuffworks.blogspot.com/2014/06/scheduling-request-procedure.html/>.
- [108] Qiuyu Peng, Anwar Walid, Jaehyun Hwang, and Steven H Low. Multipath tcp: Analysis, design, and implementation. *IEEE/ACM Transactions on Networking*, 24(1):596–609, 2014.
- [109] Coia Prant. x264. code.videolan.org. Accessed: 5 June 2025. [Online.] Available: <https://code.videolan.org/videolan/x264/>.
- [110] Ravi Prasad, Constantinos Dovrolis, Margaret Murray, and KC Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network*, 17(6):27–35, 2003.
- [111] Darijo Raca, Ahmed H Zahran, Cormac J Sreenan, Rakesh K Sinha, Emir Halepovic, Rittwik Jana, and Vijay Gopalakrishnan. On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges. *IEEE Communications Magazine*, 58(3):11–17, 2020.
- [112] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. How hard can it be? designing and implementing a deployable multipath TCP. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation*, pages 399–412. USENIX, 2012.
- [113] Reza Rassool. VMAF reproducibility: Validating a perceptual practical video quality metric. In *Proceedings of the 2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–2. IEEE, 2017.
- [114] Devdeep Ray, Connor Smith, Teng Wei, David Chu, and Srinivasan Seshan. Sqp: Congestion control for low-latency interactive video streaming. *arXiv preprint arXiv:2207.11857*, 2022.

- [115] Paul Regnier, George Lima, and Luciano Barreto. Evaluation of interrupt handling timeliness in real-time linux operating systems. *ACM SIGOPS Operating Systems Review*, 42(6):52–63, 2008.
- [116] Ubuntu Manpage Repository. tc. manpages.ubuntu.com. Accessed: 10 July 2025. [Online.] Available: <https://manpages.ubuntu.com/manpages/xenial/man8/tc.8.html>.
- [117] Vinay Joseph Ribeiro, Rudolf H Riedi, Richard G Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Proceedings of the 2003 Passive and Active Measurements Workshop*. ACM, 2003.
- [118] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 114–118. ACM, 2013.
- [119] Sydney Roy. What Is Video Bitrate (And What Bitrate Should You Use)? (Update). wowza.com. Accessed: 5 June 2025. [Online.] Available: <https://www.wowza.com/blog/what-is-video-bitrate-and-what-bitrate-should-you-use>.
- [120] Yusuf Sani, Darijo Raca, Jason J Quinlan, and Cormac J Sreenan. SMASH: A supervised machine learning approach to adaptive video streaming over HTTP. In *Proceedings of the 12th International Conference on Quality of Multimedia Experience*, pages 1–6. IEEE, 2020.
- [121] H. Schulzrinne et al. RTP: A Transport Protocol for Real-Time Applications. data-tracker.ietf.org. Accessed: 10 July 2025. [Online.] Available: <https://datatracker.ietf.org/doc/html/rfc3550>.
- [122] scikit-learn developers. sklearn.feature_selection.rfe. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html. Accessed: 2024-11-27.
- [123] Scream Contributors. Ericssonresearch / scream. github.com. Accessed: 5 June 2025. [Online.] Available: <https://github.com/EricssonResearch/scream>.
- [124] Mehdi Semsarzadeh, Mohsen Jamali Langroodi, and Mahmoud Reza Hashemi. An adaptive rate control for faster bitrate shaping in x264 based video conferencing. In *Proceedings of the 2010 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–4. IEEE, 2010.

- [125] Satadal Sengupta, Niloy Ganguly, Sandip Chakraborty, and Pradipta De. Hotdash: Hotspot aware adaptive video streaming using deep reinforcement learning. In *Proceedings of the 2018 IEEE 26th International Conference on Network Protocols*, pages 165–175. IEEE, 2018.
- [126] Alexandre F Silva, Mylene CQ Farias, and Judith A Redi. Perceptual annoyance models for videos with combinations of spatial and temporal artifacts. *IEEE Transactions on Multimedia*, 18(12):2446–2456, 2016.
- [127] Varun Singh, Saba Ahsan, and Jörg Ott. MPRTP: multipath considerations for real-time media. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 190–201. ACM, 2013.
- [128] Cong-Xi Song, Biao Han, and Jin-Shu Su. 4D-MAP: Multipath Adaptive Packet Scheduling for Live Streaming over QUIC. *Journal of Computer Science and Technology*, 39(1):159–176, 2024.
- [129] Wei Song and Dian W Tjondronegoro. Acceptability-based QoE models for mobile video. *IEEE Transactions on Multimedia*, 16(3):738–750, 2014.
- [130] SonicWall. Wireless: SNR, RSSI and Noise basics of wireless troubleshooting. sonicwall.com. Accessed: 10 July 2025. [Online.] Available: <https://www.sonicwall.com/support/knowledge-base/wireless-snr-rssi-and-noise-basics-of-wireless-troubleshooting/180314090744170/>.
- [131] Randall Stewart and Christopher Metz. SCTP: new transport protocol for TCP/IP. *IEEE Internet Computing*, 5(6):64–69, 2001.
- [132] Thomas Stockhammer. Dynamic adaptive streaming over HTTP: standards and design principles. In *Proceedings of the 2nd ACM Multimedia Systems Conference*, pages 133–144. ACM, 2011.
- [133] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the 2003 Internet Measurement Conference*, pages 39–44. ACM, 2003.
- [134] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the ACM SIGCOMM 2016 Conference*, pages 272–285. ACM, 2016.

- [135] Chris Taylor et al. cm256. github.com Accessed: 5 June 2025. [Online.] Available: <https://github.com/catid/cm256>.
- [136] Restream Team. 48 live streaming stats you should know in 2024. restream.io. Accessed: 5 June 2025. [Online.] Available: <https://restream.io/blog/live-streaming-statistics/>.
- [137] Neha Rupesh Thakur and Ashwini S Kunte. Analysing schedulers of multipath TCP in diverse environment. In *Proceedings of the 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 1337–1340. IEEE, 2021.
- [138] Guibin Tian and Yong Liu. Towards agile and smooth video adaptation in dynamic HTTP streaming. In *Proceedings of the 8th Conference on Emerging Networking EXperiments and Technologies*, pages 109–120. ACM, 2012.
- [139] Franco Tommasi, Valerio De Luca, and Catiuscia Melle. Packet losses and objective video quality metrics in h. 264 video streaming. *Journal of Visual Communication and Image Representation*, 27:7–27, 2015.
- [140] TVU Networks. Live Streaming App For Mobile Video Broadcasting. tvunetworks.com. Accessed: 5 June 2025. [Online.] Available: <https://www.tvunetworks.com/products/tvu-anywhere/>.
- [141] Videvo. Lioness walking towards camera. videvo.net. Accessed: 5 June 2025. [Online.] Available: <https://www.videvo.net/video/lioness-walking-towards-camera/463060/>.
- [142] Tobias Viernickel, Alexander Froemmgen, Amr Rizk, Boris Koldehofe, and Ralf Steinmetz. Multipath quic: A deployable multipath transport protocol. In *Proceedings of the 2018 IEEE International Conference on Communications*, pages 1–7. IEEE, 2018.
- [143] Vislink. Mobile Viewpoint. mobileviewpoint.com. Accessed: 10 July 2025. [Online.] Available: <https://www.mobileviewpoint.com/>.
- [144] Shibo Wang, Shusen Yang, Xiao Kong, Chenglei Wu, Longwei Jiang, Chenren Xu, Cong Zhao, Xuesong Yang, Jianjun Xiao, Xin Liu, et al. Pudica: Toward near-zero queuing delay in congestion control for cloud gaming. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation*, pages 113–129. USENIX, 2024.

- [145] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation*, pages 459–471. USENIX, 2013.
- [146] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath TCP. In *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*. USENIX, 2011.
- [147] Jiyan Wu, Bo Cheng, Chau Yuen, Yanlei Shang, and Junliang Chen. Distortion-aware concurrent multipath transfer for mobile video streaming in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 14(4):688–701, 2014.
- [148] Jiyan Wu, Yanlei Shang, Jun Huang, Xue Zhang, Bo Cheng, and Junliang Chen. Joint source-channel coding and optimization for mobile video streaming in heterogeneous wireless networks. *EURASIP Journal on Wireless Communications and Networking*, 2013(1):1–16, 2013.
- [149] Jiyan Wu, Rui Tan, and Ming Wang. Energy-efficient multipath TCP for quality-guaranteed video over heterogeneous wireless networks. *IEEE Transactions on Multimedia*, 21(6):1593–1608, 2018.
- [150] Jiyan Wu, Chau Yuen, Bo Cheng, Ming Wang, and Junliang Chen. Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 15(9):2345–2361, 2015.
- [151] Jiyan Wu, Chau Yuen, Ming Wang, and Junliang Chen. Content-aware concurrent multipath transfer for high-definition video streaming over heterogeneous wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 27(3):710–723, 2015.
- [152] Junhao Wu, Wang Yang, and Lihuan Hui. An MPQUIC-based frame-granularity transmission mechanism for adaptive video streaming. In *Proceedings of the IEEE 2023 Global Communications Conference*, pages 2686–2691. IEEE, 2023.
- [153] Siyuan Xiang, Lin Cai, and Jianping Pan. Adaptive scalable video streaming in wireless networks. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 167–172. ACM, 2012.

- [154] Yufeng Xiao, Shanzhi Chen, Xin Li, and Yuhong Li. A new available bandwidth measurement method based on self-loading periodic streams. In *Proceedings of the 2007 International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1904–1907. IEEE, 2007.
- [155] Min Xing, Siyuan Xiang, and Lin Cai. A real-time adaptive algorithm for video streaming over multiple wireless access networks. *IEEE Journal on Selected Areas in Communications*, 32(4):795–805, 2014.
- [156] Changqiao Xu, Tianjiao Liu, Jianfeng Guan, Hongke Zhang, and Gabriel-Miro Muntean. CMT-QA: Quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 12(11):2193–2205, 2012.
- [157] Changqiao Xu, Wei Quan, Athanasios V Vasilakos, Hongke Zhang, and Gabriel-Miro Muntean. Information-centric cost-efficient optimization for multimedia content delivery in mobile vehicular networks. *Computer Communications*, 99:93–106, 2017.
- [158] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. Video telephony for end-consumers: Measurement study of google+, ichtat, and skype. In *Proceedings of the 2012 Internet Measurement Conference*, pages 371–384. ACM, 2012.
- [159] Kaiping Xue, Jiangping Han, Dan Ni, Wenjia Wei, Ying Cai, Qing Xu, and Peilin Hong. DPSAF: Forward prediction based dynamic packet scheduling and adjusting with feedback for multipath TCP in lossy heterogeneous networks. *IEEE Transactions on Vehicular Technology*, 67(2):1521–1534, 2017.
- [160] Fan Yang, Qi Wang, and Paul D Amer. Out-of-order transmission for in-order arrival scheduling for multipath TCP. In *Proceedings of the IEEE 2014 International Conference on Advanced Information Networking and Applications Workshops*, pages 749–752. IEEE, 2014.
- [161] Wang Yang, Jing Cao, and Fan Wu. Adaptive Video Streaming with Scalable Video Coding using Multipath QUIC. In *Proceedings of the IEEE 2021 International Performance, Computing, and Communications Conference*, pages 1–7. IEEE, 2021.
- [162] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. *ACM SIGCOMM Computer Communication Review*, 45(4):325–338, 2015.

- [163] Chaoqun Yue, Ruofan Jin, Kyoungwon Suh, Yanyuan Qin, Bing Wang, and Wei Wei. Linkforecast: Cellular link bandwidth prediction in lte networks. *IEEE Transactions on Mobile Computing*, 17(7):1582–1594, 2017.
- [164] Kai Zeng, Hojatollah Yeganeh, and Zhou Wang. Quality-of-experience of streaming video: Interactions between presentation quality and playback stalling. In *Proceedings of the IEEE 2016 International Conference on Image Processing*, pages 2405–2409. IEEE, 2016.
- [165] Liekang Zeng, Haowei Chen, Daipeng Feng, Xiaoxi Zhang, and Xu Chen. A3D: Adaptive, accurate, and autonomous navigation for edge-assisted drones. *IEEE/ACM Transactions on Networking*, 2023.
- [166] Jia Zhao, Jiangchuan Liu, Cong Zhang, Yong Cui, Yong Jiang, and Wei Gong. MPTCP+: Enhancing adaptive HTTP video streaming over multipath. In *Proceedings of the 2020 IEEE/ACM 28th International Symposium on Quality of Service*, pages 1–6. IEEE, 2020.
- [167] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. Xlink: QoE-driven multipath QUIC transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM Conference*, pages 418–432. ACM, 2021.
- [168] Chao Zhou, Chia-Wen Lin, and Zongming Guo. mDASH: A markov decision-based rate adaptation approach for dynamic HTTP streaming. *IEEE Transactions on Multimedia*, 18(4):738–751, 2016.
- [169] X Zhu et al. Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media. datatracker.ietf.org. Accessed: 5 June 2025. [Online.] Available: <https://datatracker.ietf.org/doc/html/rfc8698>.
- [170] Yutong Zhu, Li Song, Rong Xie, and Wenjun Zhang. Sjt4k video subjective quality dataset for content adaptive bit rate estimation without encoding. In *Proceedings of the 2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–4. IEEE, 2016.
- [171] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K Sinha. Can accurate predictions improve video streaming in cellular networks? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 57–62. ACM, 2015.

- [172] Xutong Zuo, Jiayu Yang, Mowei Wang, and Yong Cui. Adaptive bitrate with user-level QOE preference for video streaming. In *Proceedings of the 2022 IEEE International Conference on Computer Communications*, pages 1279–1288. IEEE, 2022.