PH.D. DISSERTATION

AN EFFICIENT FRAMEWORK OF CONGESTION CONTROL FOR NEXT-GENERATION NETWORKS

by

Ihsan Ayyub Qazi

BSc (Honors) Computer Science and Mathematics

Lahore University of Management Sciences, Pakistan, 2005

Submitted to the Graduate Faculty of

the Department of Computer Science in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2010

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF COMPUTER SCIENCE

This dissertation was presented

by

Ihsan Ayyub Qazi

It was defended on

July 8, 2010

and approved by

Taieb Znati, Professor, University of Pittsburgh, USA

Craig Partridge, Chief Scientist, BBN Technologies, USA

Daniel Mosse, Professor, University of Pittsburgh, USA

Lachlan L. H. Andrew, Associate Professor, Swinburne University of Technology, Australia

Rami Melhem, Professor, University of Pittsburgh, USA

Dissertation Director: Taieb Znati, Professor, University of Pittsburgh, USA

PH.D. DISSERTATION

AN EFFICIENT FRAMEWORK OF CONGESTION CONTROL FOR NEXT-GENERATION NETWORKS

Ihsan Ayyub Qazi, PhD

University of Pittsburgh, 2010

The success of the Internet can partly be attributed to the congestion control algorithm in the Transmission Control Protocol (TCP). However, the tremendous growth in the range of bandwidth-delay products, Bit-Error Rates, and the increased diversity in applications has stressed TCP and the need for new transport protocol designs has become increasingly important.

Prior research has focused on the design of either end-to-end protocols (e.g., CUBIC) that rely on implicit congestion signals such as loss and/or delay or network-based protocols (e.g., XCP) that use precise per-flow feedback from the network. While the former category of schemes have performance limitations, the latter are hard to deploy, can introduce high per-packet overhead, and open up new security challenges. This dissertation explores the middle ground between these designs and makes four contributions. First, we study the interplay between performance and feedback in congestion control protocols. We argue that congestion feedback in the form of aggregate load can provide the richness needed to meet the challenges of next-generation networks and applications. Second, we present the design, analysis, and evaluation of an efficient framework for congestion control called Binary Marking Congestion Control (BMCC). BMCC uses aggregate load feedback to achieve efficient and fair bandwidth allocations on high bandwidth-delay networks while minimizing packet loss rates and average queue length. BMCC reduces flow completion times by up to 4x over TCP and uses only the existing Explicit Congestion Notification bits.

Next, we consider the incremental deployment of BMCC. We study the bandwidth sharing

properties of BMCC and TCP over different partial deployment scenarios. We then present algorithms for ensuring safe co-existence of BMCC and TCP on the Internet. Finally, we consider the performance of BMCC over Wireless LANs. We show that the time-varying nature of the capacity of a WLAN can lead to significant performance issues for protocols that require capacity estimates for feedback computation. Using a simple model, we characterize the capacity of a WLAN and propose the usage of the average service rate experienced by network layer packets as an estimate for capacity. Through extensive evaluation, we show that the resulting estimates provide good performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

My five years in graduate school have been one of the best periods of my life; filled with learning, self-discovery, and fun. There were several people who provided me support, care, and help during this time. Many of them had a profound impact on me as a researcher and as a person for which I am deeply indebted to them. Indeed, it is these people who made my stay in Pittsburgh all the more memorable.

First of all, I would like to thank my advisor Taieb Znati for his mentorship and support. He gave me a lot of freedom in my research and allowed me to work on problems of my interest. He always motivated me to aim high and set high standards for his students which helped me in becoming a better researcher. I am deeply grateful to him for allowing me to regularly visit my family during this time. Daniel Mosse was very generous in providing me with an enormous amount of his time and wisdom. I always thoroughly enjoyed the brain storming sessions we had both about research and life. He taught me how to maintain an attitude of a student despite becoming a senior researcher (in his case, a full professor). His passion for teaching and research showed me how academia can be exceedingly rewarding and great fun at the same time.

Lachlan Andrew has been a great collaborator and a mentor. He taught me the importance of being precise and also showed me how to think deeply about problems. His sharp questions and insightful comments forced me to think harder about my research and its limitations. I am thankful to him for suggesting me to look into packet marking schemes for conveying high resolution feedback. This lead to research which now forms Chapter 4 of this thesis. Craig Partridge taught me many of the right questions to ask about research. He taught me how to scope problems into manageable pieces and the importance of defending every number that appears in a research work. His sharp questions and deep insights helped me in appreciating the larger context of my work. I would like to thank him, Daniel Mosse, Lachlan Andrew, and Rami Melhem for serving in my

thesis committee.

Fahad Dogar was my apartment mate for four years and more importantly, my best friend (for nearly a decade now). He has been a great mentor, an excellent critic and a fantastic blackboard off which I'd frequently bounce off my ideas and dumb questions at times. During the not-so-great times of my studies, he was right there and provided excellent support. He was very careful in giving me the space to find my own ways of addressing issues that I encountered along the way. I don't have words to express my gratitude for the enormous amount of time, support, and advice he gave me whenever I needed it. The things we did outside our research activities made my stay all the more fun. The movies we saw, the frequent and rather long life deliberations we had, and the exciting cooking sessions we arranged (after which we would lie down and ask, "why did we eat soo much, again?") made my stay very memorable. He taught me about the importance of time management, doing things consistently well, and the usefulness of the "daily grind" in the life of a Ph.D. student. He always motivated me to strive for the best for which I deeply indebted to him. It is hard to imagine a better friend than Fahad. Thank you, Fahad.

I would like to thank Samir Sheikh for his great company at the start of my Ph.D. studies; Ali Arshad, for his amazing support, infinite car rides, and for making my stay in Pitt memorable; Asim Jamshed, for great discussions and for frequently providing me with a much needed break by visiting my office; Qurratulain Saeed, for her excellent support and her feedback on some of my papers; Ammar Baray, for being an excellent company. I'll miss our cricket sessions; Babar bhai, for great umpiring and for being a great friend; Sara Tahir, for always finding a way to have fun; Usman Khan, for being a lot of fun; Saad Nadeem, for always bringing a smile on my face with his interesting comments and stories; and Uzair Shuja, for being extremely helpful. Thank you and all other friends for your help, support, and company during the last five years.

It has been a pleasure to share the office space with my colleagues. I would like to thank Hammad for his useful feedback and support. I will miss our daily interactions. PJ was always there to provide feedback. Thank you for the interesting discussions on life and philosophy. I am grateful to Hammad, PJ, and Octavio for diligently proof-reading parts of my thesis. In addition, I would like to thank Ananda Gopalan, Hui Ling, Sherif Khattab, Mohammaed Aly, Mahmood Elhaddad, Subrata Acharya, Muhammad Hammoud, Hyunjin Lee, Muhammad Muhammad, Peter Djalaliev, Lory Al Moakar, and Mehmud Abliz for being excellent Department mates.

I would like to thank the Pitt staff for all their help and support. Kathleen o Connor amazed me with her management skills. She had a solution to everything :-) Kathleen Allport was wonderful in helping me with all the arrangements that needed to be done. Keena was very helpful in making sure I did the documentations right. I would also like to thank Karen, Nancy, Loretta, Terry, Bob Hoffman and the rest of the staff for their help during my stay at Pitt.

I am grateful to my undergraduate teachers for instilling in me the passion for networking research. Tariq Jadoon got me interested in computer networking through his course. Otherwise, I might well have been an economist :-), Zartash Uzmi inspired me to do networking research with his rather contagious passion and inquisitiveness at solving research problems.

My parents made great sacrifices in bringing me up and in providing me with good education. This thesis would not have been possible without the amazing support, patience, constant encouragement, care, and prayers of my family. No words can express my gratitude for them. I believe my mother deserves a doctorate for her patience and steadfastness during the last five years. My father's calmness and his great confidence in my abilities encouraged me to always go an extra mile in my research. My little brother, Zafar, has been a rock of support in my life. He provided me with the kind of support that one can only dream about. Raza bhai's (my older brother) smile and words of encouragement provided me with a lot of strength during my stay here. Zafar and Raza bhai made great efforts in making sure my mother was doing well during the time I could not be with her. I cannot imagine better siblings than Zafar and Raza bhai. I am indebted to them for their love and support. Finally, despite the importance of this education to me and my family, this can never be worth the time of my parents and siblings.

1.0 INTRODUCTION

1.1 MOTIVATION

The Internet is a global infrastructure for information exchange that has transformed the social, economic, and political aspects of our lives. One of the most crucial building blocks of the Internet is a mechanism for resource sharing and controlling congestion on the Internet. When end-hosts access a certain resource (such as a webpage from CNN, a video on YouTube, etc.,) on the Internet, it is important to ensure that they do not overwhelm network elements (such as routers), are able to efficiently utilize network resources, and achieve fairness in some agreed-upon sense. Today, congestion control for most of the traffic is provided by the Transmission Control Protocol (TCP) [1, 2, 3, 4].

The importance of congestion control was practically realized in the late 1980s when the Internet was first observed to have experienced a series of *"Congestion Collapses"*[1], which resulted in about a factor of thousand drop in throughput [2]. Later, a congestion control algorithm was retrofitted in the Transmission Control Protocol (TCP), the most dominant transport protocol on the Internet today, to avoid this situation. Since then, TCP has served the Internet well. However, TCP is now showing significant performance limitations and the need for new transport protocol designs has become increasingly important [5, 6, 7, 8, 9, 10, 11]

This need has arisen from TCP's inability to meet the challenges brought about by the tremendous growth in the range of link capacities, latencies, and Bit-Error Rates (BER) as well as due to increased diversity in applications and their requirements. Table 4 illustrates the increased heterogeneity in the characteristics of different links and networks in terms of capacity, latency, and BER

[1] A condition in which the network does little useful work due to severe congestion.

and Table 2 shows different types of applications and their requirements. We discuss these trends below:

Network	**Capacity**	**Latency**	**Bit-Error Rate**
Wired WANs [12, 13, 14, 15]	≈50Mbps-14Tbps	10ms-300ms	10^{-12}
Data Centers [9]	1Gbps-1Tbps	100μs-1ms	10^{-12}
Satellite Networks [5]	100kbps-155Mbps	250ms-1s	10^{-10}
802.11 WLAN/Mesh Networks [16, 17]	<1Mbps-600Mbps	1ms-200ms	$> 10^{-5}$
Wired LANs (e.g., Ethernet) [18]	10Mbps-10Gbps	<1ms	$\leq 10^{-12}$
Cellular Data Networks (e.g., 3G) [19]	384kbps-3Mbps	≈100ms-1s	10^{-5}

Table 1: Characteristics of various networks

- *Increased Heterogeneity in Link Characteristics*: Technological advancements and the increased diversity in networks has lead to link capacities ranging from kbps in multi-hop wireless mesh networks to Tbps in data centers, round-trip times that range from microseconds in data centers to about a second in Satellite networks, and BERs that range from 10^{-12} on optical fiber links to higher than 10^{-5} in wireless networks. For the Internet to continue to provide good performance in the future, transport protocols must have mechanisms to accommodate and leverage this diversity in networks and their characteristics (see Table 4).

- *Diversity in Applications*: Similarly, proliferation of new applications (e.g., Skype, Facebook, Flickr, Twitter) means that the Internet must be able to support a variety of application requirements and traffic workloads. For interactive applications, this means low latency, small jitter[2], and small throughput variations for good performance[3][23, 24] and for data center applications, that operate across a range of flow sizes, it means low latency for short flows, high

[2][20] argues that maintaining small jitter may not be a central requirement for networks as long as end-hosts have sufficient memory for buffering. Achieving low (bounded) delay and providing the needed average bandwidth are more important requirements. However, small jitter can reduce the memory requirements needed for buffering.

[3]A reader may ask if it is reasonable to put the requirements of interactive applications on congestion control

throughput for long flows, and high burst tolerance due to workflow patterns (see Table 2) [9].

Application Type	**Examples**	**Requirements**
Interactive	VoIP, Video Conferencing	Low latency, small jitter, and small throughput variations
Short Web Transfers ($<$100kB)	Google Search, Facebook	Short response times
Medium Sized Transfers (100KB$-$5MB)	YouTube, Flickr, Facebook Photos	Low latency
Large Transfers ($>$ 5MB)	Movie downloads, Software Updates	High throughput

Table 2: Requirements of different types applications

Moreover, future trends in technology indicate that this diversity is likely to increase as the Internet evolves to support a much richer set of applications than enabled today by the existing Internet. While such diversity is highly beneficial, it has also raised new challenges for TCP and next-generation transport protocols. We now discuss why these trends are problematic for TCP and argue that the need for new transport protocol designs is both important and urgent.

TCP Limitations: TCP becomes inefficient as capacity and/or delay increases [6, 7]. On Internet paths with large Bandwidth-Delay Product (BDP), TCP's additive increase of one packet per Round-Trip Time (RTT) means that flows take a long time to acquire any spare capacity. Since TCP allocates bandwidth inversely proportional to the RTT of flows, short RTT flows can grab most of the bandwidth when sharing a bottleneck with longer RTT flows [25]. TCP's reliance on

protocols such as TCP that favor reliability over timeliness. We think it is reasonable to do so because congestion control protocols are often used for carrying real-time traffic due to their reliability features, are safer for the Internet and may actually be preferred over UDP, which is often blocked by NATs and firewalls. [21] reports that more than 50% of the commercial streaming traffic is carried over TCP. Popular media applications such as Skype and Windows Media Services use TCP often due to the wide deployment of NATs and firewalls that block UDP traffic [22].

packet loss as the primary signal of congestion means that sources need to fill router buffers to obtain the signal. First, this leads to high average queue length, which increases the response time of all flows. Second, it leads to periodical increase in end-to-end delay which makes real-time applications like VoIP and Video Conferencing difficult to support on the Internet [23].

Moreover, TCP assumes that most packet losses inside the network are due to overflow of router buffers [2]. However, with the proliferation of wireless networks, buffer overflows are no longer the primary source of packet loss, instead bit errors, handoffs, multi-path fading, $etc.$, account for a significant proportion of lost packets [26, 27]. This can greatly limit throughput because responses to these losses need not be similar [28, 29, 30]. Further, the unguided exponential increases of TCP's slow-start algorithm has known efficiency and stability problems that have manifested itself in large BDP networks [31], wireless networks [32], and data centers (see Chapter 2 for a detailed discussion on TCP problems). [9].

1.1.1 Ideal Properties of a Congestion Control Protocol

There are several properties that we would like in an congestion control mechanism. These properties can placed into two categories: i) network and flow level properties and ii) properties related to ease of deployment and complexity of implementation. We now discuss these properties.

- *Efficiency*: Given enough traffic demand, a protocol should be able to maintain close to 100% link utilization across a range of link capacities and round-trip times. Of special interest are environments where the per-flow bandwidth-delay product is large. Indeed we are quickly moving towards an Internet where tens and hundreds of Gbps end-to-end paths would be common. This is evidenced by the fact that operators in Hong Kong (China) and Japan are already offering 1 Gbps broadband services to residential networks [33].

- *Fairness*: A congestion control protocol should be able to allocate the link bandwidth in a fair manner. Several definitions of fairness have been proposed in the literature [34]; the focus of this dissertation is on RTT fairness. RTT fairness allows flows with different round-trip times to achieve the same throughput. Achieving this, for example, will allow a flow traversing a Satellite link to compete with a flow directly connected to a high-speed backbone link with a

much smaller RTT.

- *Minimal Queuing Delay*: Ideally, a protocol should maintain low *average* queue length at all times. This is so because queued up packets increase latency for all flows.
  - *Average Flow Completion Times*: The diversity in applications indicates that the Internet traffic comprises of flows with a mix of transfer sizes (see Table 2). A protocol should be able to achieve better average flow completion times than TCP. The flow completion times of short transfers is typically bound by their round-trip times. Therefore, the key to reduce their transfer times is to reduce the queueing delay experienced by them, which can be significant portion of the RTT. For medium-sized and long transfers, quickly achieving a high throughput is important for reducing their flow completion times.

- *Negligible Loss Rate*: A protocol should induce negligible losses due to buffer overflows. These losses result in retransmissions which wastes network bandwidth. Moreover, large number of losses can create system-level bottlenecks that can impact performance [31].

- *Stability*: When networks face transient erratic behaviors caused by sudden increases in traffic (e.g., flash-crowds), a protocol should be able to detect this behavior and move to a stable operating point [34].

- *Resilience to End-User Misinformation*: Ideally, a protocol should not rely on truthful information from the senders (e.g., round-trip times, congestion window sizes) to achieve high performance as it opens up new ways for malicious users to cheat the system[4].

- *Easy to Deploy*: Deployment is an important problem that any practical congestion control protocol needs to address. For a congestion control protocol to perform correctly, the sender, the receiver, and the routers along the path have to comply with the assumptions of the protocol. Updating all of these components at once is a difficult task. Ideally, a protocol should be amenable to deployment in the current Internet architecture. By this we mean, it should not require changes in the IP header or the addition of a shim layer, can coexist with TCP with-

[4] A related property is *resilience to malicious ISPs*. This can be a hard property to satisfy when flows must traverse the malicious ISPs to reach certain destinations and/or sources.

out requiring complex router-level mechanisms, and can be incrementally deployed without hurting existing protocols in a significant way.

- *Low Router Complexity*: Internet routers are complex and expensive devices that run at very high speeds. Todays high-speed routers can spend less than a nanosecond on a packet before the packet is due to depart on the link. Hence, protocols that require the routers to maintain per-flow queues and classify every packet are considered complex [35]. Such protocols require more memory that increases the cost. Moreover, they perform a few memory accesses per packet, and this usually takes too long. Ideally, a scheme should be able to achieve the above properties without any per-flow information, state and queue, or additional per-packet computations in the routers.

In the last couple of decades, a lot of research has gone into addressing the limitations of TCP. However, prior research has focussed on two extreme points in the design space of congestion control protocols. At one end are end-to-end schemes (e.g., HighSpeed TCP [36], FAST [37], CUBIC [38]) that rely on packet loss and/or delay as signals of congestion. This reliance causes such schemes to introduce artificial packet losses and/or queuing at the bottleneck, which should be avoided in the first place. Research studies have shown that using such signals poses fundamental limitations in achieving high utilization and fairness while keeping low bottleneck queue and negligible packet loss rate in high BDP paths [39, 36]. On the other end reside network-based schemes (e.g., XCP [6], RCP [8]) that address these challenges by enforcing fairness and congestion control inside the network. In this category of schemes, routers compute precise *per-flow* feedback that is communicated explicitly to end-hosts (see Table 3 for a detailed comparison). Such schemes typically incur higher per-packet overhead, are hard to deploy in today's Internet as they require more bits for feedback than are available in the IP header such as XCP (128 bits), RCP (96 bits), and open up several security challenges as they critically rely on truthful information from the senders, such as their current round-trip times and congestion window sizes, to achieve good performance [35].

This dissertation explores the middle ground between these two design points and presents the design and implementation of an efficient framework for congestion control, called Binary Marking

Properties	**End-to-End Protocols**	**Network based Protocols**	**BMCC**
1) Efficient use of high bandwidth-delay links			
a) per-flow BDP is small	√	√	√
b) per-flow BDP is large		√	√
2) Fairness	√	√	√
3) Minimal queueing delay		√	√
4) Negligible loss rate		√	√
5) Stable	√	√	√
6) Resilience to end-user misinformation	√		√
7) Easy to deploy	√		√
8) Low router complexity			
a) No per-flow state or queue	√	√	√
b) No per-packet computation in routers	√		

Table 3: Properties achieved by end-to-end protocols, network-based schemes, and the Binary Marking Congestion Control (BMCC) protocol

Congestion Control (BMCC), that approximates the performance of network-based schemes with a much lower deployment barrier by using *aggregate load feedback* from the network (see Table 3). BMCC achieves efficient and fair bandwidth allocations on high BDP paths while maintaining low persistent queue length and negligible packet loss rates. Since BMCC maintains low average queue length, it considerably reduces the average completion times of flows. In comparison to TCP, this reduction is ≈4-5x. Using *aggregate* feedback (as opposed to *per-flow* feedback) helps in keeping router complexity low as they do not need to perform expensive per-packet operations [40]. Moreover, it preserves the end-to-end nature of congestion control. This means routers do not need to rely on truthful information from the senders to achieve the goals of congestion control.

We argue in this dissertation that it is fundamentally important that future transport protocols use richer feedback from the network. A richer feedback should convey the *degree* of congestion at the bottleneck as this allows transport protocols to employ scalable control laws in order to meet the goals of congestion control. Therefore, we use load[5] (the ratio of demand to capacity) as a congestion signal. This signal captures both the link utilization as well as the queue length at the bottleneck and thus provides an accurate characterization of congestion. Further, its scale-free nature allows it to be encoded using only few bits, which is helpful for deployment. BMCC's use of load factor as a congestion signal enables it to decouple loss recovery from congestion control. This facilitates distinguishing error losses from congestion related losses, which is important in wireless environments. Using extensive packet-level simulations, we assess the efficacy of BMCC and perform comparisons with several proposed schemes. We provide a detailed deployment path for BMCC and show how it can be incrementally deployed on the Internet. Moreover, we present methods for available capacity estimation in wireless networks (specifically, 802.11-based WLANs), that allow BMCC to achieve good performance in such networks.

1.2 THESIS STATEMENT

The thesis of this dissertation is that "***congestion information in the form of aggregate load feedback can provide the richness needed to achieve the performance requirements of next-generation networks and can enable new designs to meet the challenges brought about by the diversity in networks and applications. Moreover, we can measure this congestion feedback using minimal overhead while achieving a high level of accuracy.***"

The above thesis statement raises several research challenges. At a high-level, the biggest challenge is: *"If we were to design a congestion control protocol, that uses aggregate load as a congestion signal, from scratch, how would we design it? and can such a protocol meet the requirements of next-generation networks and applications?* We systematically address this challenge by providing answers to the following questions:

[5] or load factor. We use these terms interchangeably

1. What granularity of aggregate load feedback is needed to achieve optimal performance? How does performance vary as a function of the accuracy of feedback?
2. What window increase/decrease policies should be in place to ensure efficient and fair bandwidth allocations on high BDP networks while keeping low queues and near-zero packet drop rates?
3. How can we efficiently convey aggregate load feedback from the routers back to the sources without hurting end-to-end performance and at the same time achieve a high level of accuracy
4. What are the deployments challenges for protocols that use aggregate load feedback as a congestion signal? How can they coexist with protocols such as loss-based TCP without introducing significant complexity inside the network for fair sharing between them? How would such protocols operate in scenarios where some devices do not provide the desired load feedback?
5. How can we estimate aggregate load in wireless networks such as Wireless LANs where capacity changes over time due to variations in channel conditions and the level of contention due to other wireless nodes?

1.3 CONTRIBUTIONS AND NOVELTY

This thesis makes four contributions.

First, we are the first to formalize and present a clear understanding of the interplay between performance and accuracy of congestion feedback. We show that in order to achieve efficient bandwidth allocation, 3-bit load feedback suffices. While 2-bit feedback schemes are far from optimal, performance follows the law of diminishing returns when more than 3-bits are used. Prior works only considered 1-bit and 2-bit feedback schemes but we provide a general treatment of the role of performance and feedback. We further show that backing off as a function of the severity of congestion can lead to better congestion responsiveness and faster fairness convergence. Recent studies in data center networks highlight the importance of this property [9]. For this purpose, we show that three additional bits are sufficient. Such an understanding will help in the design and development of new protocols using aggregate load as feedback.

Second, we design a framework for congestion control called Binary Marking Congestion Control (BMCC) that uses only the existing two ECN bits to convey high resolution (up to 16-bit in our current implementation) congestion feedback information. We achieve this with the help of a packet marking scheme called Adaptive Deterministic Packet Marking (ADPM), that was proposed in prior work but never used in a practical congestion control protocol. We then use the insights from our previous work to design a congestion control protocol that uses ADPM. We present analytical models of convergence to fairness and efficient bandwidth allocations and study the impact of ADPM on the resulting protocol. The results show that BMCC achieve high utilization and fairness on large BDP networks while maintaining low persistent queue length and negligible packet loss rate. Moreover, it outperforms existing schemes such as SACK, XCP, VCP and in some cases RCP in terms of average flow completion times. This research has already contributed to the development of new protocols that use less bits to obtain more information about the congestion state [41, 42]. We believe it will further our understanding of protocols that rely on marking schemes that tradeoff accuracy for delay.

Third, we study the problem of capacity estimation in 802.11-based WLANs in order to compute accurate load feedback. While capacity is fixed in wired networks, it is not the case in wireless networks where capacity changes over time due to variations in channel conditions and the level of contention due to other nodes in the wireless neighborhood. We present a simple model for characterizing the available capacity of wireless nodes and then propose the use of average service rate experienced by network layer packets as an estimate for available capacity. We showed that under a variety of wireless settings, the estimates are robust and result in good performance. The proposed use of these estimates can be extended to other wireless networks such as multi-hop wireless mesh networks.

Finally, we study the bandwidth sharing properties of BMCC and TCP over different kinds of bottlenecks. We show that TCP flows can starve BMCC flows when sharing a BMCC-enabled bottleneck whereas the converse holds true when they share a non-BMCC bottleneck. To address the former case, we present simple router algorithms that prevent starvation and allow fairer bandwidth sharing between BMCC and TCP. These algorithms have applicability beyond that of BMCC. For the latter case, we propose mechanisms for BMCC flows to detect the bottleneck type (which we

show is feasible) and shift to TCP mode. We believe such switching has a useful role to play in the migration towards more efficient congestion control.

1.4 THESIS ORGANIZATION

The rest of the dissertation is organized as follows:

Chapter 2 provides a background on TCP and discusses the related work. This is followed by Chapter 3, which presents the design considerations for congestion control protocols and studies the interplay between performance and congestion feedback. Chapter 4 presents the design of the Binary Marking Congestion Control (BMCC) protocol, presents analytical models that predict and provide insights into the convergence properties of the protocol, and presents detailed simulations results. Chapter 5 addresses deployment challenges of BMCC and proposes simple router algorithms to prevent starvation and improve bandwidth sharing between BMCC and TCP. Chapter 6 investigates the performance of BMCC over 802.11-based wireless LANs and proposes the usage of average service rate of network layer packets as estimates for available capacity. It then presents a detailed simulation evaluation of the proposed mechanism. In Chapter 7, we summarize the contributions of this dissertation, discuss the limitations of this work, present possible future directions of this work, and finally offer concluding remarks.

2.0 BACKGROUND AND RELATED WORK

In this chapter, we provide background on TCP, discuss problems with it and do a brief survey of the related work. In particular, we survey works in the area of congestion control protocol design for (a) high bandwidth-delay product networks, and (b) wireless networks. We start by providing a few definitions.

2.1 DEFINITIONS

Here are some definitions that we use throughout the dissertation.

1. **Flow:** A flow is a sequence of packets that share the same source and destination IP addresses as well as the corresponding port numbers.

2. **Round-Trip Time (RTT):** The round-trip time of a flow is the total time taken by the network to deliver both a packet from a flow's sender to its receiver, and the corresponding acknowledgement to the sender.

3. **Congestion Window:** The congestion window is the maximum number of packets (or bytes) that a flow is permitted to send into the network in a given RTT.

4. **Receiver Window:** The receiver window is the maximum number of packets (or bytes) that can be buffered by the *receiver* at a given point in time.
5. **Window Size:** A flow's window size is the minimum of the congestion and receiver window sizes.

6. **Flow Rate:** A flow's rate is the ratio of its window size to its RTT.

7. **Bottleneck Link:** A link l is said to be the bottleneck link for source r if the link is fully utilized and r has the largest flow rate among all sources using link l [34].

8. **Max-Min Fairness:** Max-min fairness is a fairness criteria that aims at maximizing the minimum throughput of the flows that share a single bottleneck. When flows have infinite demand, this implies equal throughput for all flows. If a flow's demand is less than its fair share, the flow gets the minimum of its demand and its fair share. The extra bandwidth is divided equally among the other sources. Many protocols (e.g., TCP and its variants) try to approximate max-min fairness. We refer to these approximations using the general term "fairness" [34].

2.2 CONGESTION CONTROL ON THE INTERNET

2.2.1 Transmission Control Protocol (TCP)

TCP provides an end-to-end, reliable, byte-oriented service to the applications. To prevent senders from overwhelming the receivers, TCP employs flow control whereas in order to avoid overwhelming the network, it uses congestion control. In this section, we focus on the congestion control algorithm used by TCP.

A TCP source maintains a sliding window called congestion window or $cwnd$, which indicates its current belief about the number of packets that the network can safely handle. TCP increases $cwnd$ after every new acknowledgement[1] until it detects a packet loss, upon which, TCP decreases $cwnd$, which in turn reduces the load on the network. TCP detects packets losses by two mechanisms. First, when a packet is sent, it initializes a timer. If no acknowledgement is received within the timeout interval, the packet is assumed to be lost. Second, when out-of-order packets are received by TCP receivers, they send acknowledgements for the last in-order packet received. When sources receive three duplicate acknowledgements, it assumes that a packet was lost and

[1]The receipt of an acknowledgement indicates that a packet has just left the network.

retransmits a packet[2].

TCP uses two algorithms for dynamically changing $cwnd$, namely, *Slow-Start* and *Congestion Avoidance*. In Slow-Start, sources increase $cwnd$ by one Maximum Segment Size (MSS) for each new acknowledgment received, which results in the window doubling after each window's worth of data is acknowledged[3]. With this exponential increase, $RTT \cdot log_2 W$ seconds time is required to reach a window of size W. A connection enters Slow-Start when starting up or on experiencing a packet retransmission timeout, and exits Slow-Start when it detects a packet loss or when the congestion window has reached a dynamically computed threshold, $ssthresh$. More specifically, $ssthresh$ is set to half of the current congestion window when packet loss was detected. TCP exits Slow-Start to enter the Congestion Avoidance phase, where it continues to probe for available bandwidth, but more cautiously than in Slow-Start.

In the Congestion Avoidance phase, TCP uses the Additive Increase and Multiplicative Decrease (AIMD) algorithm to probe for network bandwidth [2, 45]. With AIMD, TCP increases its window by one packet every round-trip time until it experiences a packet drop. Upon detecting a packet loss, TCP reduces its $cwnd$ by half.

2.3 PERFORMANCE LIMITATIONS OF THE TRANSMISSION CONTROL PROTOCOL (TCP)

TCP has a number of well-known problems. We enumerate them here with examples wherever appropriate.

1. TCP becomes inefficient as capacity and/or delay increases [7, 4, 46, 35, 39, 47]. On Internet paths with large BDP or "pipe size", TCP's additive increase of one packet per Round-Trip Time (RTT) means that flows take a long time to acquire any spare capacity. For instance, given 1500-byte packets and a 100 ms RTT, filling a 10 Gbps pipe would take at least 1.6 hours between packet drops.

[2]Note that this assumes that the network does not sufficiently reorder packets. With multi-path routing and network coding mechanisms such as COPE [43] that can significantly reorder packets, this is likely to result in problems [44].

[3]This is when the receiver sends an acknowledgment for every data packet.

2. TCP relies on packet loss feedback to adapt its sending rate. The end-to-end loss probability needs to be impractically small for a TCP flow to be able to sustain a large equilibrium window, making it hard for high-speed connections to obtain large throughput. A simple model for the steady-state TCP throughput [48], X (bytes/sec), for large bulk transfers (in the AIMD phase), with loss probability p, is

$$X = \frac{MSS \cdot K}{RTT \cdot \sqrt{p}} \tag{2.1}$$

where MSS is the maximum segment size, p is the loss probability, and K is a constant that depends on the loss pattern (e.g., random, periodic), and the whether the acknowledgements are delayed or not. It can be easily derived from the above model that in order to sustain a throughput of 10 Gbps with a RTT of 100 ms, the loss probability cannot be more than 1 in 5 billion packets.

3. TCP assumes that packet losses inside the network occur due to congestion (or overflow of router buffers). With the widespread adoption of 802.11 wireless LANs, multi-hop wireless mesh networks, etc., congestion can no longer be assumed as the only source of packet loss; instead bit errors, hand-offs, multi-path fading, etc., account for a significant proportion of lost packets [49, 29, 30]. TCP forces senders to respond to all these kinds of losses in the same manner. This results in degraded performance [28].

4. TCP flows achieve throughput proportional to $1/RTT^z$, where $1 \leq z \leq 2$ [50]. This property can result in high level of unfairness. For instance, consider two TCP flows sharing a common bottleneck with RTTs of 10 ms and 200 ms, respectively. The above behavior implies that the short RTT flow can obtain at least 20 times and up to 400 times higher throughput than the large RTT flow.

5. With TCP's slow start algorithm, the congestion window is doubled every RTT. This exponential growth, without improved feedback from the networks, results in a large number of packet losses within one RTT, especially on high BDP paths. This can badly affect ACK-clocking and cause TCP timeouts. [31] reports that the huge number of packet losses frequently forces TCP to experience long black-outs (sometimes more than 100 seconds), either caused by very

long timeouts or by system-related bottlenecks caused by operations such as freeing up a large number of packet buffers, e.g., SKBs in Linux, within a short period of time. Router buffers larger than one BDP can seriously aggravate this problem.

6. Finally, TCP deliberately fills up any amount of buffering available at the bottleneck router. Extra buffers mean extra delay, which increases the response time of flows. In addition, it introduces large variations in delay and throughput, which is not amenable to real-time applications [23].

2.4 CONGESTION CONTROL PROTOCOLS FOR LARGE BANDWIDTH-DELAY PRODUCT NETWORKS

Since the pioneering work of Van Jacobson [2] and Chiu and Jain [45], a lot of research effort has gone into the design of fair and efficient congestion control protocols for high-speed and long-distance networks. These works can broadly be placed into three categories: end-to-end (e2e) scheme with implicit feedback [47, 51, 36, 37, 52, 38, 52, 53, 54], e2e schemes with explicit feedback [55, 49, 56, 57, 58, 10] and network-based schemes [59, 6, 8]. We will now discuss prior work in each of these categories.

2.4.1 End-to-End Congestion Control Protocols with Implicit Feedback

The basic idea behind these schemes is to treat the network as a black box and infer congestion via implicit signals such as packet loss and/or delay. However, these signals are also performance metrics that protocols aim to minimize. Heavy reliance on these signals implies that protocols can only respond to congestion once it has become bad enough to overflow network buffers, or at least to form significant standing queues. Moreover, packet loss is a binary signal of congestion which only indicates if there is congestion (packet loss) or no congestion (no packet loss). This forces the sources to be conservative in their increase policy and aggressive in their decrease policy. While delay is a multi-bit signal, it is hard to measure reliably [60, 61, 62]. Research studies have shown

that using such signals poses fundamental limitations in achieving high utilization and fairness while keeping low bottleneck queue and negligible packet loss rate in high BDP paths [36, 39].

Traditional loss-based versions of TCP such as Tahoe [2], Reno [63], NewReno [64, 65], and SACK [66] use AIMD with $\alpha = 1$ pkt/RTT and $\beta = 0.5$. This causes inefficiencies on high BDP paths. Scalable TCP [67] uses MIMD as opposed to AIMD. While MIMD improves efficiency, it may not converge to a fair bandwidth allocation, especially, in the presence of synchronous feedback as shown in [45]. Highspeed TCP [36] uses a generalized AIMD where the linear increase factor and multiplicative decrease factor are adjusted by a convex function of the current congestion window size. Below a certain cutoff value, HSTCP uses the same factors as standard TCP. BIC [52] adds a binary search phase to standard TCP for probing the available bandwidth in a logarithmic manner. In CUBIC TCP [38] (default in Linux from 2.6.19), the window growth function is a cubic function of the elapsed time since the last congestion event; a similar strategy is employed by HTCP [68]. FAST [37] uses queueing delay as a congestion signal and improves on TCP Vegas's AIAD policy with a proportional controller. PCP [53] chooses the sending rate of a flow by estimating the available bandwidth at the bottleneck. The spare bandwidth is estimated by measuring the inter-arrival time of a sequence of probe packets [69]. However, this requires accurate timers and small jitter. While PCP performs well in lightly loaded links, it is unclear how PCP's performance and stability properties vary under high load. CTCP [54] uses loss and delay for congestion window growth. LTCP [47] layers congestion control of two scales for high speed, large RTT networks. DCCP [70] provides a framework for implementing congestion control protocols without reliability. PERT [71] uses delay to emulate the behavior AQM/ECN from the end-hosts.

2.4.2 Network-based Congestion Control Protocols

In network-based schemes, congestion control and fairness are enforced inside the network [6, 8, 59, 72]. The routers compute detailed, per-flow feedback based on the estimates of link capacity, number of ongoing flows, and the average round-trip time of flows. Per-flow state such as the round-trip time, congestion window size and/or rate of flows, etc., is carried in the flows' packets to the routers for feedback computation, which is then sent back to the sources for adjusting

their transmission rates. Feedback computation typically requires a number of floating operations, which introduces higher per-packet overhead on routers than end-to-end schemes [73].

Network-based schemes achieve high performance while overcoming many limitations of pure end-to-end schemes. However, they have a very steep deployment path in today's Internet. Carrying per-flow state from the senders and the feedback information from the routers requires additional fields either in an IP option, a TCP option [74] or modified header [6], or a shim layer [75]. Therefore, universal deployment is a big challenge for these protocols because many routers are configured to drop packets containing IP options and IP payloads may be encrypted. Moreover, such schemes exhibit inter-operability issues with standard TCP traffic. Since many new protocols aim to avoid queueing, they are likely to be starved by TCP flows which continue to increase their rate until the buffer overflows. It is often proposed that this problem be solved by having separate queues for packets which do and don't support the new protocol [75]. However, that raises complex management issues and unnecessarily increases the cost.

RCP is a network-based scheme in which each router assigns a single rate to all flows passing through it. Determining a single rate, however, requires an accurate estimate of the number of ongoing flows, a difficult task considering the dynamic nature of the Internet [8]. XCP regulates the sending rate by making routers send precise window increment/decrements in feedback to each flow [6]. ATM ABR service, previously, also proposed explicit rate control, however, unlike XCP, ABR protocols usually maintain per-flow state at the switches and are essentially rate-based [59].

2.4.3 End-to-End Congestion Control Protocols with Explicit Feedback

To approximate the performance of network-based schemes but with a lower deployment barrier and low per-packet overhead, limited feedback-based schemes such as TCP+AQM/ECN [76, 77, 78, 55], VCP [56], MLCP [57] have been proposed. These schemes use few bit(s) of explicit feedback from the network to aid end-hosts in making congestion control decisions. Such protocols typically require modifications at the end-hosts with incremental support from the routers.

With TCP+AQM/ECN, routers use Active Queue Management (AQM) schemes such as RED [77], REM [76], and PI [78] and ECN [55] to mark packets. Such schemes use one bit of explicit

feedback for signalling whether there is congestion or not. Since the feedback is highly imprecise, sources are forced to be conservative in their increase policy and aggressive in their decrease policy. Moreover, AQM schemes operate without requiring changes to TCP sender control laws. While this provides benefit of deployment, it limits how much performance can be achieved from such schemes. In the case of 2-bit schemes (such as VCP), Qazi et al. [57] recently showed that they converge unnecessarily slowly, and that quantizing to at least 16 levels is required to achieve near-optimal convergence speed.

2.5 CONGESTION CONTROL PROTOCOLS FOR WIRELESS NETWORKS

While network capacity is fixed in wired networks, it is not the case in wireless networks. The nature of the wireless medium implies that sharing also takes place at lower layers. This results in dynamically varying network capacity that depends on channel conditions and traffic load due to other nodes in the neighborhood, etc. Moreover, congestion in wireless networks exhibits strong location dependency i.e., different nodes in a congested neighborhood locally perceive different degrees of congestion [79, 80, 81].

A number of protocols have been proposed for achieving efficient and fair bandwidth allocation in wireless networks. These schemes can be placed into two broad categories: (1) schemes that use local heuristics (such as queue length and the average number MAC retransmissions) to estimate congestion in the wireless neighborhood [82, 83, 84, 32], (2) schemes that determine neighborhood congestion in a distributed manner by sharing local congestion information with other nodes in the neighborhood [80, 85, 81].

LRED uses a exponential weighted moving average of the number of retransmissions at the MAC layer as a measure of congestion while marking packets in a manner similar to RED [**?**, 77]. ATP is a rate-based congestion control scheme that involves explicit rate feedback to the sources from the network [32]. In ATP, a flow receives the maximum of the weighted average of the sum of the queueing and transmission delay at any node traversed by the flow. ATP uses the inverse of this delay as the sending rate of a sender. NRED [81] identifies a subset of flows which share

channel capacity with flows passing through a congested node. It regulates flow rates by estimating a neighborhood queue size and by using RED [77]-style marking on packets in this queue. [80] presented two congestion control protocols, WCP and WCPCap that outperformed TCP in well-known problematic topologies. WCP uses AIMD, RED with ECN support and congestion sharing to achieve better performance than TCP. whereas WCPCap uses the precise neighborhood capacity estimates guide the source sending rates. Another line of work uses back-pressure hop-by-hop congestion control [86].

Developing reliable capacity estimation algorithms is arguably the most important performance concern for protocols that rely on a capacity estimate for feedback computation. On this front, [87] proposed to use interference graphs given network topology and traffic demands and showed that the problem can be seen as a multi-commodity flow problem. However, they do not model the MAC and instead assume that packet transmissions can be finely scheduled across links. Such models tend to over-estimate the performance of 802.11 networks. [88] uses a theoretical model of 802.11 DCF and solves a centralized optimization problem for determining rate allocations to satisfy efficiency and/or fairness objectives. However, it is unclear how effectively it can be used for congestion control purposes because solving the proposed optimization problem operates at much longer timescales (in the order to few minutes) than typical RTT of flows. With increasing heterogeneity of wireless devices based on standards such as IEEE 802.15.4 (Zigbee), IEEE 802.16 (WiMAX), etc., model-based approaches would require an accurate analytical model for every new standard and the computation and communication overhead may be too high to be useful. Ideally, capacity estimation should be independent of any link layer technology.

One way of solving this problem is to design a customized transport protocol for the wireless medium and use connection splitting at the gateways [89]. While this may be a plausible approach to take it isn't clear if this is the most desirable solution. Another approach is to devise algorithms for tracking the changing network capacity [80, 90, 88]. This will allow all protocols to be useable in wireless networks. In this dissertation, we take the latter approach.

3.0 UNDERSTANDING THE INTERPLAY BETWEEN PERFORMANCE AND FEEDBACK

In this chapter, we study the interplay between performance and feedback in congestion control protocols. We first discuss the key factors that affect the design of congestion control protocols and compare several congestion signals and sender control laws. Using aggregate load as the congestion signal, we then analyze the tradeoff between performance and feedback. Our results show that 3-bit feedback is sufficient for achieving near-optimal rate convergence to an efficient bandwidth allocation. While the performance gap between 2-bit and 3-bit schemes is large, gains follow the *law of diminishing returns* when more than 3 bits are used. Further, we show that using multiple back-off factors enables the protocol to adjust its fairness convergence rate, rate variations and responsiveness to congestion based on the degree of congestion at the bottleneck. Based on these insights, we design the Multi-Level feedback Congestion control Protocol (MLCP).

3.1 DESIGN CONSIDERATIONS FOR CONGESTION CONTROL PROTOCOLS

Two factors play a defining role in the design of congestion control protocols: (a) *congestion signal* (e.g., packet loss, delay, load), which characterizes the state of congestion at the bottleneck and (b) *sender control laws* (e.g., Additive Increase Multiplicative Decrease), that define the responses to the congestion signal(s). In this section, we discuss how TCP's congestion signal and control laws give rise to several TCP issues shown in Figure 1. In addition, we discuss the strengths and limitations of several other congestion signals and sender control laws.

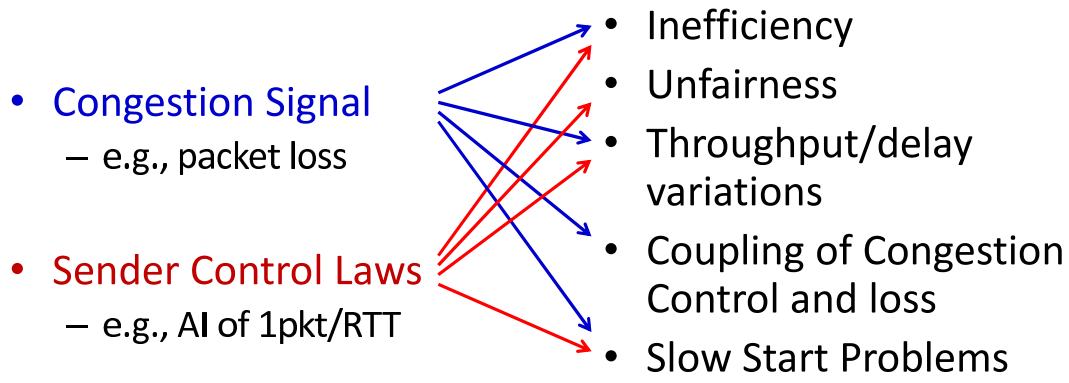


Figure 1: Impact of the congestion signal and sender control laws on TCP performance.

3.1.1 Congestion Signals

TCP uses *packet loss* as a signal of congestion. One reason for its usage is that it is simple to obtain and need not be explicitly communicated. However, it has several limitations when used as a congestion signal. First, packet losses (due to congestion) occur when router buffers overflow. Hence, to obtain such a signal, the network needs to be driven to a point of extreme congestion which should be avoided in the first place[1]. Second, packet losses and one bit marks are binary signals of congestion. They do not indicate the *degree* of congestion at the bottleneck. This forces the sources to be conservative in their increase policy because they are unsure if the network is close to congestion or away from it, and aggressive in their decrease policy because when the signal is received, the network is highly congested. These characteristics contribute to TCP's inefficiency and the variations it introduces in throughput and delay. Third, usage of packet loss as a congestion signal couples congestion control with loss recovery, which makes different kinds

[1]To avoid such a situation, routers have to employ algorithms (such as RED [77]) that characterize congestion in overload and either drop packets or set bits in the passing packets to indicate incipient congestion.

of losses difficult to discern. This forces sources to respond in the same manner to all kinds of losses when the responses should be different. Four, it causes senders to employ algorithms for inferring/detecting losses. For example, TCP uses three duplicate acknowledgements as an indication of packet loss, in addition to using timeouts in other cases. While this heuristic works fine in many situations but with the increased use of load balancing by ISPs, multi-path routing, and mechanisms such as network coding, packet re-ordering may become a rule rather than an exception, in which case, this heuristic would need to be revisited. A different congestion signal may help in minimizing packet losses in which case it reduces the problems associated with this heuristic. Finally, packets losses are undesirable because they waste bandwidth due to the need for retransmissions. However, it is important to note that in a statistically multiplexed data network such as the Internet, packet losses are unavoidable in some situations.

Some versions of TCP use *queueing delay* as a signal of congestion [51, 37]. Its usage is based on the observation that when queues start building at the bottleneck, it increases the sender observed RTT. Hence, this increase in RTT can be used to guide the sender towards a target throughput. Delay provides much more fine grained information about congestion than packet loss. Moreover, it can be obtained using sender-only mechanisms[2]. However, there are few challenges with using delay as the primary congestion signal. First, delay only characterizes the state of congestion in the overload region i.e., when queues start building up. Since low load and high load regions are not represented, increasing too fast in these regions can cause too many packet losses and slow increase may lead to inefficiency. Second, it is hard to reliably measure forward path queueing delay due to several sources of noise e.g., delayed ACKs, delays due to complex scheduling mechanisms such as in DSL/Cable links, etc. Third, it requires each flow to have a few packets in the queue which implies that the buffer requirement increases linearly with the number of flows. Finally, it is hard for delay based protocols to compete fairly with loss based schemes. Other versions of TCP use the the time elapsed between two congestion events to guide the sending rate [68, 38]. The algorithms assumes that the longer it takes between two congestion events to occur, higher is the BDP of the path. Therefore, sources should increase their congestion window sizes at a higher rate. While this is true on high BDP, wired paths, in WLANs and multi-hop

[2]Note that TCP already employs delay as a signal in the form of timeouts during times of extreme congestion or when when a path for packets does not exist.

wireless mesh networks, where the bit-error rates are much higher, this may result in sending more data than what the network can handle due to the need for higher number of retransmissions in the wireless medium.

Available bandwidth, which is the difference between link capacity and the input traffic rate, is another congestion signal used by some protocols [6, 8]. It accurately captures the state of the bottleneck, but it typically requires large number of bits to be encoded. This makes their deployment, given the current Internet, challenging. Moreover, it can be hard to reliably measure available bandwidth in wireless networks, such as static multi-hop wireless mesh networks, where capacity changes highly dynamically [91, 88, 80].

Load factor, defined as the ratio of traffic demand to capacity, is a scale-free parameter, unlike available bandwidth [92, 56]. It indicates the degree of congestion at the bottleneck and decouples congestion control from loss recovery. Moreover, load factor can be encoded using few bits. This is important for compatibility with IP, which offers only a limited number of bits for explicit congestion notification. While raw load factor values hide the absolute available capacity, its rate of change can be used to provide this information. One issue that is common between available bandwidth and load factor is that they need an estimate of the available capacity, which can be hard to obtain e.g., in wireless networks [88, 80]. Due to the richness and the scale-free nature of load factor, we use it as a congestion signal in our work.

3.1.2 Sender Control Laws

TCP uses AIMD, with fixed parameters, in the congestion avoidance mode. Additive increase of 1 pkt/RTT does not scale with the BDP of the path leading to inefficiencies [6]. Multiplicative decrease by half upon every congestion event introduces large variations in throughput and delay [23]. Simply increasing the AI parameter value, without improved feedback, does not work because it would result in high packet loss rate in low BDP networks. Similarly, increasing the MD parameter, would lead to higher average queueing delays for all flows.

To overcome efficiency problems, we need control laws that scale gracefully with the BDP of the path while being backward compatible with TCP. One option is to use AIMD with parameters

that depend on the load factor. While better than the case of fixed parameters, it is difficult to come up with parameters that would work across a range of bandwidth-delay products because load factor doesn't expose the absolute bandwidth. Another option is to use a scalable control law such as MI, which results in exponential growth and scales well with capacity and/or delay and augment it with AIMD to provide fairness. We take this approach and use MIAIMD as the sender control policy [56][3]. Moreover, in order to achieve RTT-fair rates, control laws need to be scaled so that long RTT flows send more traffic in order to compensate for their longer RTT.

3.2 FEEDBACK ANALYSIS

Every protocol that uses load factor as a signal of congestion must consider three important issues (1) *How many bits to use for carrying load-factor information?* (2) *What transition points to choose for each symbol?* (3) *What actions should end-hosts take based on the received signal?* In this section, we address these issues in detail.

The number of bits used in representing the feedback signal impacts the preciseness of congestion information. This, in turn, determines how conservative a source may need to be in order to compensate for the loss of information. However, having large number of bits in representing load-factor information is not necessarily desirable. On one hand, increasing the number of bits is likely to increase the overhead caused by the need to process and respond to different levels of congestion. On the other hand, it leads to a more precise estimation of the level of congestion and, therefore, a more accurate response from the sources. Hence, the goal is to determine the number of congestion levels that provide the best trade-off between performance improvements and the number of bits used in the feedback.

The performance metrics likely to be affected by the preciseness of the feedback signal include (1) rate of convergence to high utilization and (2) rate of convergence to fairness. The analysis of these metrics is used to derive the *optimal* number of congestion levels.

[3]Note that [56] uses MIAIMD with fixed parameters

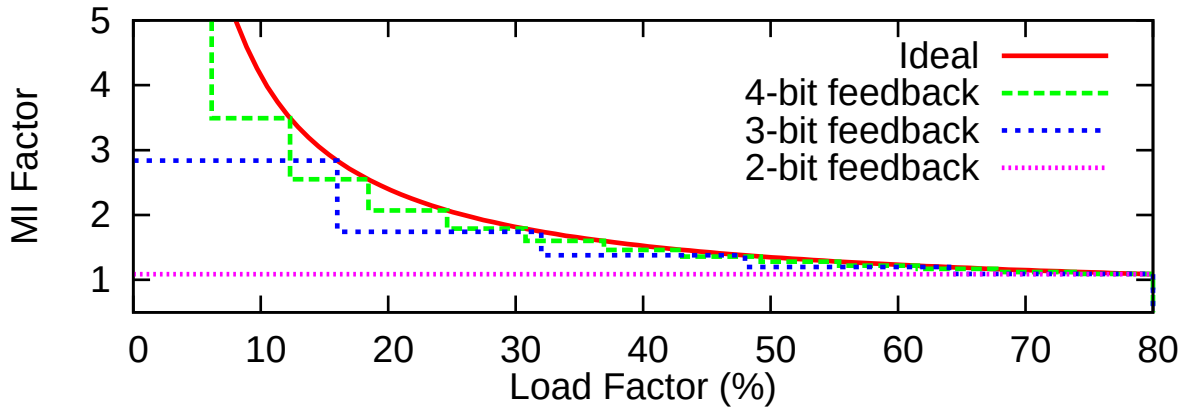


Figure 2: Comparison of MI factors of the ideal protocol with 2-bit, 3-bit and 4-bit feedback schemes

3.2.1 Rate of Convergence to High Utilization

Window-based congestion control protocols often use MI to converge exponentially to high utilization. However, *stable* protocols often require the magnitude of the MI factor to be proportional to the available bandwidth at the bottleneck [6, 56]. In the context of load-factor based congestion control protocols, this translates into requiring the MI factor to be proportional to $1 - \sigma$, where σ is the load factor at the bottleneck. We, therefore, define the MI gain function of the ideal, stable, load factor based congestion control protocol as follows.

$$\xi(\sigma) = \kappa \cdot \frac{1 - \sigma}{\sigma} \tag{3.1}$$

where $\kappa = 0.35$ is a stability constant. The stability result presented in Section 3.6 shows that congestion control protocols whose MI gains are upper-bounded by the above function, are indeed stable. It should be noted that the actual MI factor is given by $1 + \xi(\sigma)$ [56].

Figure 2 shows the MI factors used by the ideal protocol[4] along with 2-bit, 3-bit and 4-bit feed-

[4] a protocol that uses the exact load factor values

back schemes. The goal of the protocol designer is to closely match the MI gain curve of the ideal protocol using as few bits as possible. The more congestion levels the feedback signal represents, the more aggressive the sources can be due to higher MI factors. If the number of congestion levels is small, sources would have to make a conservative assumption about the actual load factor value at the bottleneck, forcing them to use small MI gains. To compare the performance of schemes using different representations of the network load levels, we examine their speed of convergence for achieving efficient bandwidth allocations.

To quantify the speed of convergence, we compute the time required to achieve a given target utilization $U_t \in [0, 1]$ (80% in our case[5]. When the system utilization, U_s, is less than U_t, each flow applies MI with a factor that depends on (1) l, the number of congestion levels used by the scheme and (2) the load factor interval (or utilization region) in which the system is operating. Suppose that a given scheme divides the target utilization region (i.e., $[0, U_t]$) into $l_0, l_1, l_2, .., l_l$ levels, where $l_0 = 0$, the size of each interval $[l_{i-1}, l_i]$ (referred to as interval i) is $s = U_t/l$ and $l_i = l_{i-1} + s$. The MI factor applied during interval i is given by $m_i = 1 + \xi(l_i)$. Note that the upper limit of an interval determines the MI factor. The reason is when $U_s \in [l_{i-1}, l_i]$, l_i is an upper-bound on system utilization and since U_s can lie anywhere in the interval, a flow must assume it to be l_i to avoid using a larger MI factor than allowed by Eq. 3.1.

Consider a single flow with an initial congestion window size of x_0 KB. Suppose that the BDP of the path of the flow is $k = C \cdot RTT$ and the system utilization is l_{i-1}. When the system utilization becomes l_i, the congestion window of a flow must be equal to $x_i = k \cdot l_i, \forall i \geq 1$. Therefore,

$$x_{i-1} \cdot (m_i)^{r_i} = x_i \tag{3.2}$$

where r_i is the number of RTTs required to achieve utilization l_i given that the system started at l_{i-1}. This implies that the amount of time required to complete interval i is

$$r_i = log_{m_i}(x_i/x_{i-1}). \tag{3.3}$$

[5]The value of U_t presents a tradeoff between achieving high utilization and convergence to fairness. To balance these requirements, we set U_t to 80%, the same value is also used in [56].

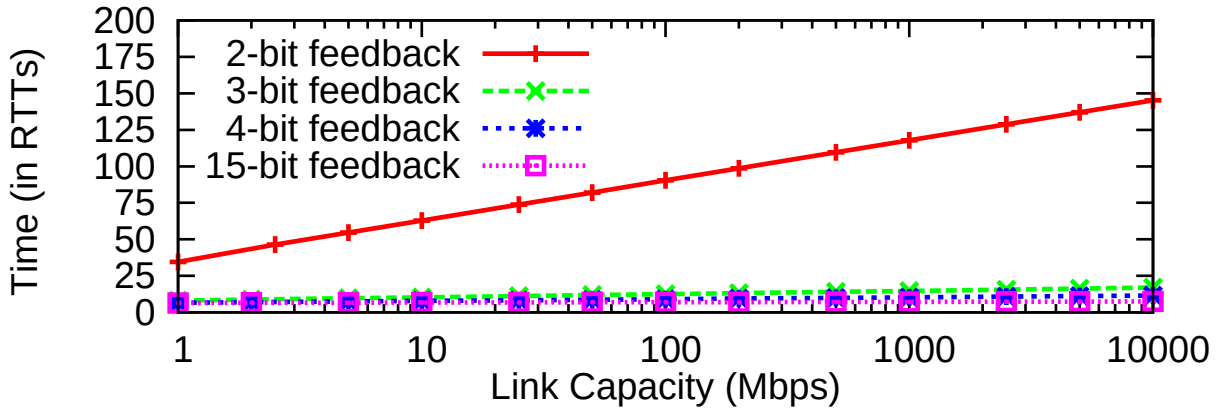


Figure 3: Comparison of the time required to achieve 80% utilization for 2-bit, 3-bit, 4-bit and 15-bit feedback schemes.

Thus, for a flow with an initial congestion window size of x_0 KB, it would take

$$r(l) = \sum_{i=1}^{l} r_i = \sum_{i=1}^{l} log_{m_i}(x_i/x_{i-1}) \tag{3.4}$$

RTTs to attain a system utilization equal to U_t, where $r(l)$ is the total time required to achieve the target utilization by a scheme that uses l congestion levels. We assume that a protocol using n bits for achieving efficient bandwidth allocations employs $l = 2^n - 3$ levels for representing the target utilization region.[6] The rest of the symbols are used for representing load factor values in $(U_t, 100)$. The analysis for determining the number of levels to represent the overload region (i.e., $\sigma \in [100\%, \infty)$) is presented in Section 3.2.2.

Consider a single flow traversing a 1 Gbps link with RTT=200 ms and an initial congestion window size of 1 KB. The above analysis implies that in order to achieve a target utilization of 80%, the 2-bit scheme would take roughly $r(1) = 118$ RTTs, the 3-bit scheme would take $r(5) =$

[6]One symbol (i.e., code $(00)_2$) is reserved for ECN-unaware source hosts to signal "not-ECN-capable-transport" to ECN-capable routers, which is needed for incremental deployment [55].

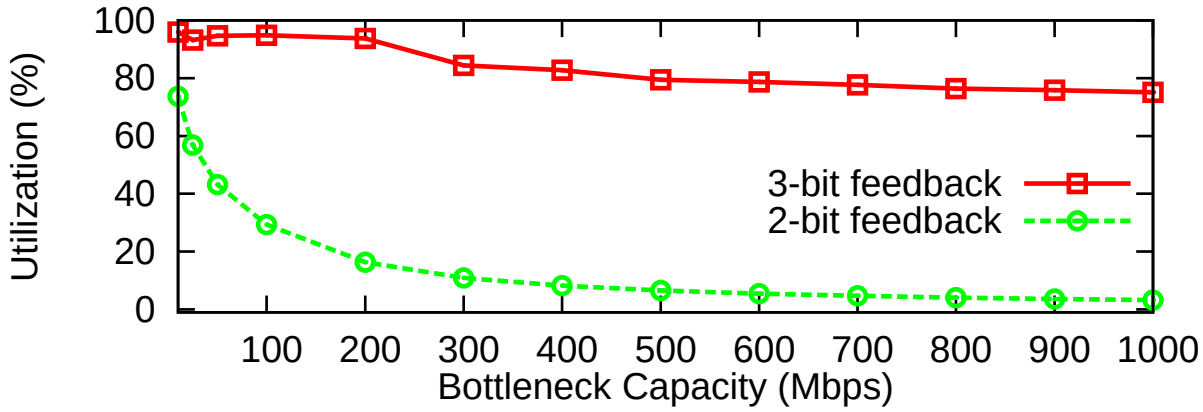


Figure 4: The figure shows the bottleneck utilization at t=10 s as a function of link capacity for the 2-bit and 3-bit feedback schemes.

15 RTTs, the 4-bit scheme would take $r(13) = 11$ RTTs and the 15-bit scheme (an approximation to the ideal scheme with infinite congestion levels) would take about $r(32765) = 8$ RTTs. Figure 3 shows the time taken by different schemes to achieve $U_t = 80\%$ as a function of the bottleneck capacity. Observe the dramatic decline in time when n is increased from 2 to 3. However, as n is increased beyond 3, the gain in performance is very little and remains largely unaffected by the bottleneck capacity. Thus for $n \geq 3$, performance improvement follows the *law of diminishing returns*. Intuitively, this happens because increasing the number of bits beyond three only helps a small portion of the target utilization region (<10%, see Figure 2). Since the time taken by a flow to attain 10% utilization is a small component of the total time required by a flow to achieve the target utilization, increasing n has little impact on performance. To validate our results, we ran ns2 simulations. Figure 4 shows the bottleneck utilization at time $t = 10$ s for protocols employing 2-bit and 3-bit feedback signals. The 3-bit protocol is able to achieve 80% utilization within the first 10 seconds across link capacities ranging from 1 Mbps to 10 Gbps, whereas, for the 2-bit protocol, utilization falls significantly as link capacity is increased.

Impact on the AFCT of flows: An important user perceivable metric is the Average Flow Completion Time (AFCT) (or response time). The 3-bit feedback scheme considerably reduces

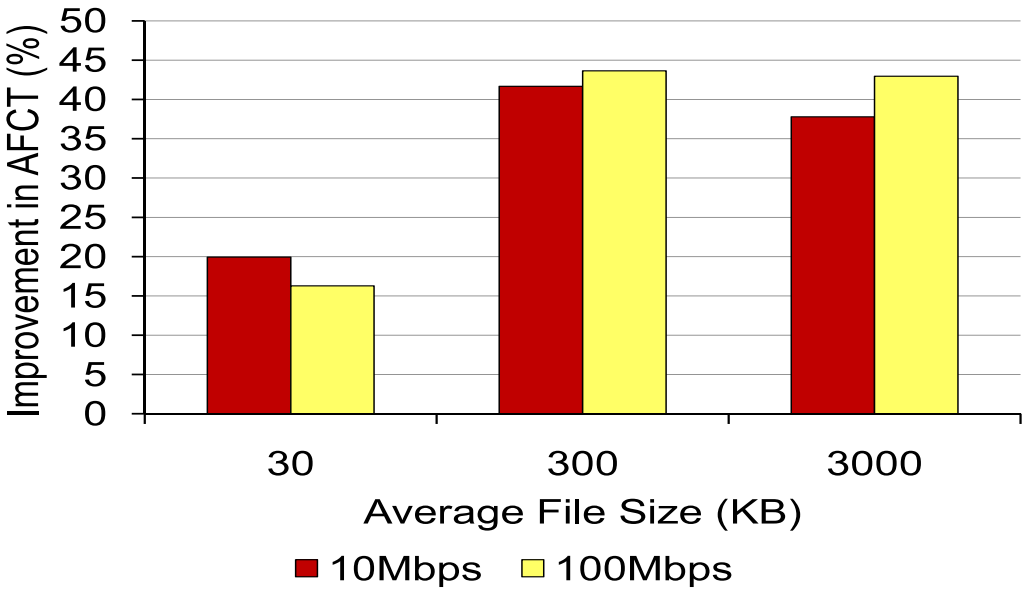


Figure 5: Improvement in AFCT that the 3-bit feedback scheme brings over the 2-bit feedback scheme as a function of the average file size on a 10 Mbps and 100 Mbps link

the AFCT due to its higher rate of convergence to efficiency. In particular, it helps short flows to finish much quicker and since most flows on the Internet are short, this improvement impacts the majority of flows [8].

Let r_2 and r_3 be the AFCT corresponding to 2-bit and 3-bit feedback schemes, respectively. The improvement in AFCT is expressed as $(1 - r_3/r_2)100\%$. Figure 5 shows the improvement in AFCT that the 3-bit scheme brings over the 2-bit scheme as a function of the average file size on a 10 Mbps and 100 Mbps link with RTT=100 ms. The file sizes obey the Pareto distribution with a shape parameter of 1.2 and the offered load was kept at 0.7. *Note that the 3-bit feedback scheme offers a reduction in AFCT of at least 16% and up to 45% over the 2-bit scheme*. Figure 6 shows the AFCT as a function of the average load at the bottleneck assuming the average file size to be 30 KB. *Observe that for average loads less than 50%, the 3-bit scheme improves the AFCT by a factor of ∼1.8. However, as the average load increases, the improvement reduces to a factor of ∼1.4.*

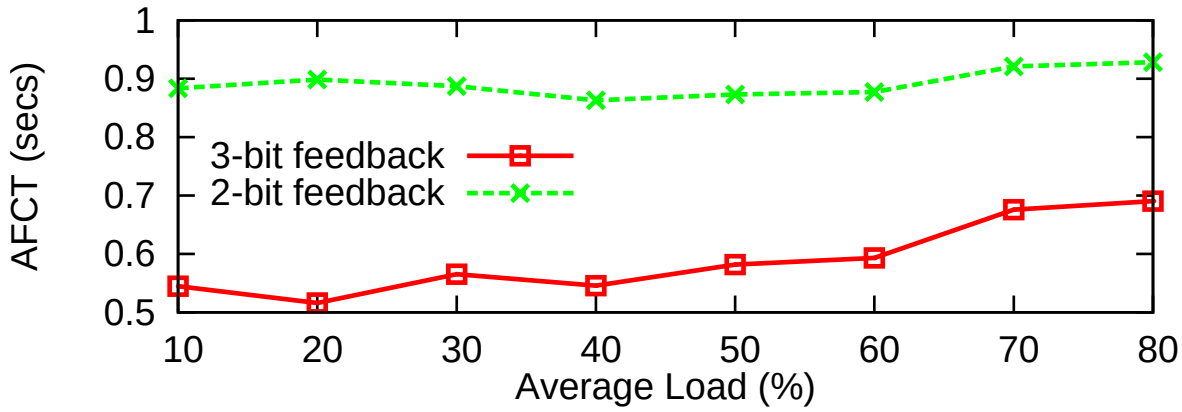


Figure 6: AFCT of flows as a function of load on a 10 Mbps link with RTT=100 ms.

3.2.2 Rate of Convergence to a Fair Share

Once high utilization is achieved, the goal of the protocol is to converge to a fair bandwidth allocation, often using control laws such as AIMD. While achieving this end, a protocol should aim to satisfy three requirements: (a) high convergence rate, (b) smooth rate variations, and (c) high responsiveness to congestion. These requirements, however, cannot be satisfied in all network scenarios. For instance, in some cases, maintaining high responsiveness to congestion may necessarily require significant variations in the rates of flows. However, one can isolate cases in which one or two of the requirements are more desirable than the rest, allowing the protocol to focus on few *complimentary* goals. These cases are as follows:

- When the system is in equilibrium (i.e., all flows have achieved their fair rates), the goal is to ensure (b) while (a) and (c) are not relevant.
- When new flows arrive, (a) and (c) are more important than (b).

A load factor based congestion control protocol may not be able to exactly discern between these cases. However, load factor values in the overload region ($\geq 100\%$) can provide for approximately identifying the above cases. The reason is that, for a fixed number of flows, the average overload remains the same. It only changes when a new flow arrives or an old flow leaves the network.

3.2.2.1 Quantifying rate of convergence to fairness and the smoothness properties of a scheme

In order to quantify rate of convergence to fairness, we measure the time taken by a newly arriving flows to achieve 70% of its fair rate. For ease of comparison, we normalize the convergence time of all schemes by the convergence time of the best scheme. We call this quantity, the *convergence ratio*. The smoothness property of a scheme is determined by the MD parameter value, β. In particular, flows that reduce their congestion windows by a factor of β in overload experience throughput variations by a factor of $1 - \beta$.

3.2.2.2 Determining the MD levels

Let a round be a single cycle between two overload events. The duration, d, of a round is determined by the increase policy and the value of β, whereas the number of rounds, p, needed for convergence to fairness is determined by β only [93].[7] Thus, a high value of β leads to slow convergence and reduces responsiveness to congestion. On the contrary, a low value of β, improves convergence and responsiveness but introduces large throughput variations.

It is important to note that when a link is highly loaded (i.e., σ is high in overload), responsiveness and convergence are more important goals than maintaining small rate variations. In order to achieve this end, we vary $\beta \in [\beta_{min}, \beta_{max}]$ with $\sigma \in [100\%, \sigma_{max}\%]$, where σ_{max} is the maximum value of σ after which sources apply β_{min}, and β_{min} and β_{max} are the minimum and maximum β values that can be used by sources. Two important factors must be considered when choosing these values. First, the minimum β value should be large enough to prevent the system from entering MI after MD because applying MI immediately after MD leads to high packet loss rate. In order to ensure this, note $\min(\sigma\beta(\sigma)) \geq 0.8$. Since for $\sigma \geq \sigma_{max}$, the smallest β is applied, therefore, β should be at least $2/3$ (for $\sigma_{max} = 1.2$). We, therefore, set β_{min} to 0.675. Second, the maximum β value should be strictly less than 1 to allow for high rate of convergence to fairness. We set β_{max} to 0.875. Note that this choice ensures that for 70% convergence only nine congestion rounds are needed [93].

We now compare the performance of the following schemes:

[7] Note that if the value of β depends on the increase policy then both of these determine p

- Scheme A uses a single β value of 0.675,
- Scheme B uses a single β value of 0.875,
- Scheme C uses two levels of MD depending on σ in overload, and
- Scheme D uses eight levels of MD as shown in Figure 7.

Varying Bottleneck Capacity: We start a single, long-lived MLCP flow at time $t = 0$. A new flow is started at $t = 50\,s$.[8] We vary bottleneck capacity and measure the convergence ratio for the second flow for each of the schemes. Figure 8 shows the convergence ratio of the four schemes along with the average load factor in overload, $\tilde{\sigma}$, at the bottleneck.[9] Observe that scheme A has the highest rate of convergence to fairness across a range of link capacities, followed by scheme D. However, scheme A also introduces the largest amount of variation in the throughput of flows; a characteristic that is highly undesirable for real-time and multimedia applications [23]. Scheme D, on the other hand, adapts β with $\tilde{\sigma}$. When C is small, scheme D applies a small β value (since $\tilde{\sigma}$ is high in overload). As the average overload decreases, scheme D increases the β value, therefore, reducing the rate variations. Scheme B takes the longest time to converge to a fair bandwidth allocation across a range of link capacities. This is due to the usage of a fixed, high β value. Scheme C improves upon the performance of scheme B. However, since it uses only two levels for representing overload, it is less aggressive than scheme D when $100\% \leq \tilde{\sigma} < 120\%$. Note that for link capacities exceeding 10 Mbps, average load factor values remain very close to 100%. This causes the sources to apply the maximum β value as allowed by each scheme; resulting in similar convergence ratios. However, this is only true for the case of two flows. Next, we show how each scheme performs when the number of flows are increased.

Varying Number of Flows: We now vary the number of long-lived flows on a bottleneck with capacity 20 Mbps. For these experiments, $N - 1$ flows are started at time $t = 0$ and flow N is started at $t = 50\,s$. We measure the convergence ratio for flow N. As the number of flows increases, average overload increases roughly linearly (see Figure 9). Observe that, while the convergence ratios achieved by schemes A and D are similar, schemes B and C take a much longer time to converge. For twenty flows, schemes A and D have convergence times that are at least

[8] Note that an inter-arrival time of 50 s ensures that the first flow is able to saturate the link before a new flow arrives

[9] The convergence time of scheme A is used as the normalizing factor because it has the highest convergence rate

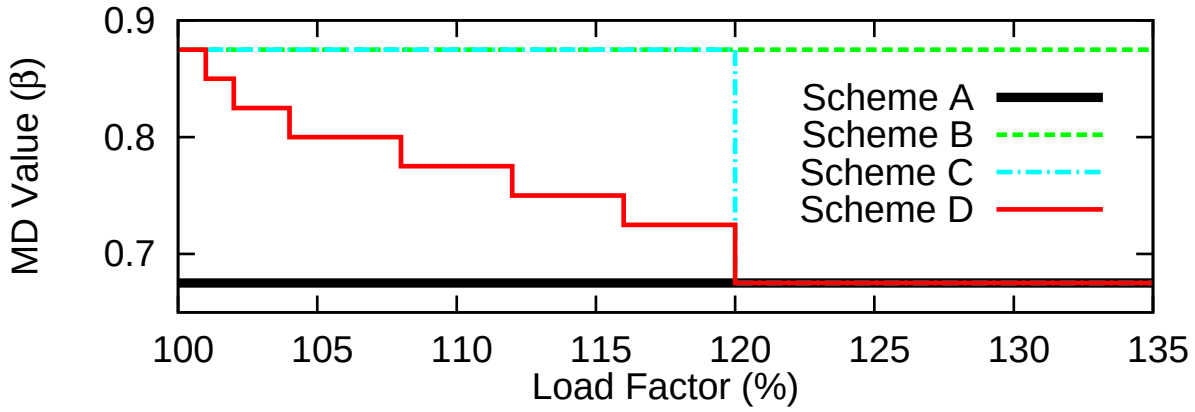


Figure 7: β as a function of load factor for different schemes

three times smaller than that of schemes B and C. The reason is that these two schemes apply a β of value 0.875, which leads to slower convergence.[10] Scheme D changes β dynamically with σ and therefore achieves the right tradeoff between convergence and rate variations. Based on these insights, we use eight levels for representing the overload region.

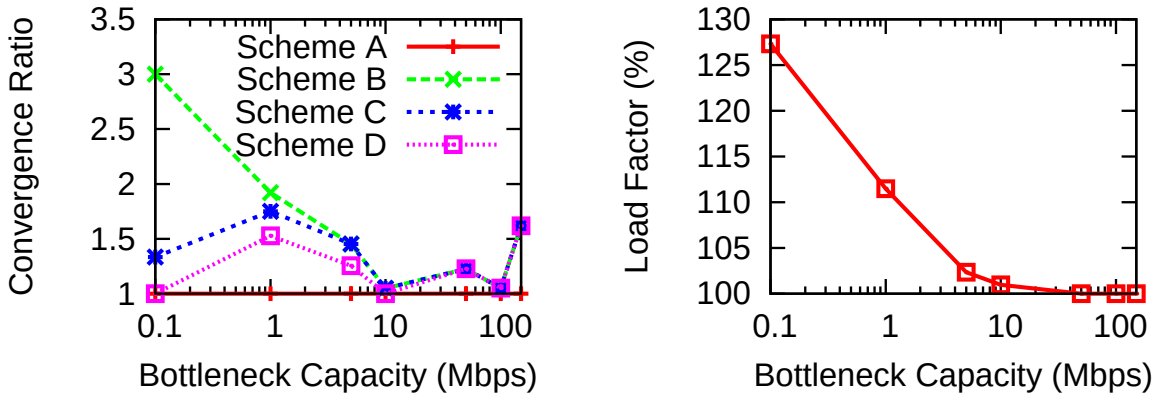


Figure 8: Convergence ratio and load factor in overload as a function of bottleneck capacity. N=2 flows, RTT=100 ms.

[10]Note that scheme C applies $\beta = 0.875$ because $\tilde{\sigma} < 120\%$ for $N \leq 20$

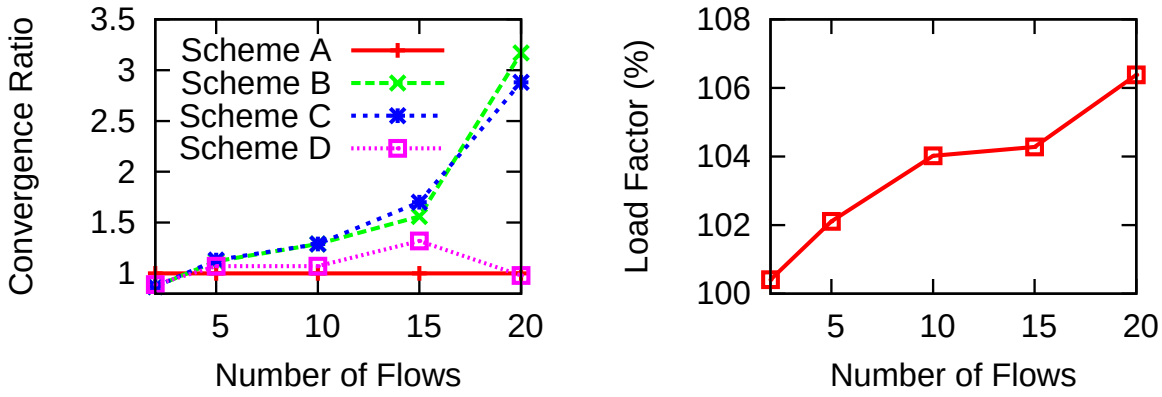


Figure 9: Convergence ratio and load factor in overload as a function of the number of flows. C=20 Mbps, RTT=100 ms.

3.2.2.3 Determining the Increase Policy The increase policy indirectly impacts (a) and (b). A large increase per RTT causes (i) MD to be applied more often and (ii) a smaller β to be applied by the end hosts, leading to fast convergence but increased oscillations. On the other hand, small increase per RTT enables existing flows to sustain their bandwidth for a longer time. However, it may lead to slow convergence. In order to achieve the benefits of these two strategies, we employ the AI-II-MD control law. When $80\% < \sigma \leq 95\%$, AI is used and for $95\% \leq \sigma < 100\%$, Inversely proportional Increase (II) is employed. AI ensures that flows quickly achieve high sending rates especially on high BDP paths, whereas II helps flows in sustaining their sending rates for a longer period of time. Since, with II, flows increase inversely proportional to the square root of their window sizes, they cause mild increments in σ when in steady state and larger when new flows arrive that have small congestion window sizes.

3.3 IMPACT OF THE LOAD MEASUREMENT INTERVAL

In this section, we discuss the impact of the load measurement interval on performance.

3.3.0.4 Estimating the load factor There are two conflicting requirements that the value of a load factor measurement interval (i.e., t_p) should aim to satisfy. First, it should be larger than the RTTs of most flows to factor out the burstiness induced by flows' responses [8, 6, 56]. Second, it should be small enough to allow for robust responses to congestion and hence avoid queue buildup [56, 92]. A single value for t_p may not be suitable for meeting both the requirements since they depend on the RTT of flows which can vary significantly across Internet links. For example, in [56], a fixed value of t_p is used, which results in significant queue buildup due to the MI gains of large RTT flows. To keep low queues, they bound the MI gains of such flows, which in turn results in considerable unfairness as shown Section 3.5.5. For small RTT ($\ll t_p$) flows, a fixed t_p results in small MI and AI gains which can considerably increase the AFCT of flows. Indeed, as the Internet incorporates more satellite links and wireless WANs, the RTT variation is going to increase. At the same time, RTT variation could be small in some cases. To meet these requirements, we dynamically adapt t_p according to the mean RTT of flows passing through the router. Each router computes the load factor σ during every t_p interval of time for each of its output links l as [94, 77, 76, 92, 56]:

$$\sigma = \frac{\lambda_l + \kappa_q \cdot q_l}{\gamma_l \cdot C_l \cdot t_p} \tag{3.5}$$

where λ_l is the amount of traffic during the period t_p, q_l is the persistent queue length during this period, κ_q controls how fast the persistent queue length drains and is set to 0.75[11]. γ_l is the target utilization, and C_l is the capacity of the link. λ_l is measured using a packet counter whereas q_l is measured using exponentially weighted moving average. The queue sample time is set at 10 ms.

3.3.0.5 Adapting t_p according to the mean RTT of flows Every packet passing through a router carries the source's estimate of its RTT. The router uses this to estimate the mean RTT of flows. To ensure stability of the RTT estimate under heterogeneous delays, it is maintained in the following way:

[11] MLCP's congestion measurement is essentially load-based even though we explicitly take into account the average queue length. Adding the queue length into the total amount of traffic only helps drain the queue faster.

(1) The data path of the MLCP router computes the average RTT over all packets seen in the control interval:

$$\overline{T}_d = \frac{\sum_i rtt_i}{n_T} \tag{3.6}$$

where $\overline{T}_d$ is the average round-trip time over interval $T = 10\,\text{ms}$, rtt_i is the RTT estimate carried in the packet header and n_T is the number of packets seen in the control interval.

(2) The control path (which performs load factor computations periodically with a period of t_p) takes $\overline{T}_d$ as input and keeps a smoother average RTT estimate, $\overline{T}_c$. The data path RTT average is used in determining the moving-average gain as follows:

$$\begin{aligned} &if(\overline{T}_d \geq \overline{T}_c) \\ &\quad \theta = T/\overline{T}_c \\ &else \\ &\quad \theta = T \cdot \overline{T}_d/(\phi \cdot \overline{T}_c^2) \end{aligned}$$

where $\phi = 50$. The control path RTT estimate is updated as follows:

$$\overline{T}_c = \overline{T}_c + \theta(\overline{T}_d - \overline{T}_c) \tag{3.7}$$

The intuition behind the above expressions is that the gain should be at most $T/\overline{T}_c$ since if average RTT is larger than T, more than one samples are received within the average RTT, so each sample should be given smaller weight. However, if $\overline{T}_d$ is smaller than $\overline{T}_c$, we want to be cautious in decreasing the control path estimate suddenly, and so the gain is made smaller by weighing it with $\overline{T}_d/\overline{T}_c$. A similar algorithm for updating the average RTT is used in [95].

The value of t_p is then chosen as follows:

$$t_p = \begin{cases} min_{\forall i \in |S|}\{s_i : s_i \in S, s_i \geq \overline{T}_c\}, & \text{if } \overline{T}_c < 1400 \\ 1400, & \text{if } \overline{T}_c \geq 1400 \end{cases}$$

where S ={80, 200, 400, 600, 800, 1000, 1200, 1400}. There are three reasons for choosing the set S. First, we do not need precise values of t_p because rigorous experimentation has shown that if the RTT of a flow is within 2.0-2.5 times t_p, there is hardly any queue buildup. Second, the mean RTT of flows must change significantly for t_p to get changed, ensuring that t_p doesn't fluctuate due to minor variations in the mean RTT. Third, these values can be communicated to the sources using only three bits. The value of t_p that is sent back to the sources is the one being used by the bottleneck router (the initial value for t_p was set at 200 ms). Using network scenarios with diverse RTTs, we show in Section 3.5.5 that setting t_p to the mean RTT of flows improves fairness significantly.

3.4 DESIGN OF THE MULTI-LEVEL FEEDBACK CONGESTION CONTROL PROTOCOL (MLCP)

In this section, we present the design of the Multi-Level feedback Congestion control Protocol (MLCP), that uses the above insights to achieve efficient and fair bandwidth allocations on High BDP paths while maintaining low persistent queue length and negligible packet loss rate. We now describe the sender, receiver and router components of MLCP.

3.4.1 MLCP Sender: Control Laws

3.4.1.1 Homogeneous RTT flows We first consider a link shared by homogeneous flows whose RTTs are equal to t_p, the load factor measurement interval. At any time t, a MLCP sender applies either MI, AI, II or MD, based on the value of the encoded load factor received from the network.

load factor region: 0-80% When the load factor at the bottleneck is below 80%, each MLCP sender applies load-factor guided MI. The MI factor applied at each transition point (i.e., 16%,

32%, 48%, 64% and 80%) are shown in Fig. 2. This translates into the following window adjustment strategy:

$$\mathsf{MI} : cwnd(t + rtt) = cwnd(t) \times (1 + \xi(\sigma)) \tag{3.8}$$

where $\xi(\sigma) = \kappa \cdot \frac{1-\sigma}{\sigma}$, σ is the load factor and $\kappa = 0.35$.

***load factor region:* >80%** When the system has achieved high utilization, senders use the AI-II-MD control law to converge to a fair share. Each sender, applies AI until σ becomes 95%, after which II is applied. When the system moves into the overload region ($\geq$100%), each sender applies MD. The following equations describe these control laws in terms of congestion window adjustments:

$$\mathsf{AI} : cwnd(t + rtt) = cwnd(t) + \alpha \tag{3.9}$$

$$\mathsf{II} : cwnd(t + rtt) = cwnd(t) + \frac{\alpha}{\sqrt{cwnd(t)}} \tag{3.10}$$

$$\mathsf{MD} : cwnd(t + \delta t) = cwnd(t) \times \beta(\sigma) \tag{3.11}$$

where $rtt = t_p$, $\delta t \to 0$, $\alpha = 1.0$ and $0 < \beta(\sigma) < 1$. To avoid over reaction to the congestion signal, MD is applied only once per t_p interval.

3.4.1.2 Parameter scaling for Heterogeneous RTT flows So far, we considered the case where the competing flows had the same RTT, equal to t_p. We now consider the case of heterogeneous RTTs. To offset the impact of heterogeneity, we normalize the RTT of each flow with the common t_p value. This emulates the behaviour of all flows having an identical RTT equal to t_p, thus making the rate increases independent of the flows' RTTs. During an interval t_p, a flow with RTT value rtt increases by a factor of $(1 + \xi_s)^{t_p/rtt}$ where ξ_s is the scaled parameter. To make the MI amount independent of a flow's RTT, $(1 + \xi_s)^{t_p/rtt} = (1 + \xi)$, which yields Eq.3.12. Similarly,

the AI gain of a flow during a time interval t_p can be obtained by solving $1 + \alpha = 1 + (t_p/rtt)\alpha_s$. However, for II, we want the increase policy to depend only on the current congestion window size, while being independent of its RTT. Therefore, we apply the same parameter scaling for II as used for AI.

$$\text{For MI : } \xi_s = (1 + \xi)^{rtt/t_p} - 1, \tag{3.12}$$

$$\text{For AI and II: } \alpha_s = \alpha \cdot \left(\frac{rtt}{t_p}\right), \tag{3.13}$$

Scaling for fair rate allocation: The above RTT-based parameter scaling only ensures that the congestion windows of flows with different RTT converge to the same value in steady state. However, fairness cannot be guaranteed, since rate ($= cwnd/rtt$) is still inversely proportional to the RTT. We need an additional scaling of the α parameter to achieve a fair share. To illustrate this, consider the AI-II-MD control mechanism applied to two competing flows where each flow $i = (1, 2)$ uses a separate α_i parameter, but a common MD parameter β. At the end of the M-th congestion epoch that includes $n > 1$ rounds of AI, $m > 1$ rounds of II and one round of MD, we have:

$$c_i(M) = \beta \cdot (c_i(M-1) + n \cdot \alpha_i + m \cdot \frac{\alpha_i}{\sqrt{c_i(M-1)}}) \tag{3.14}$$

where $c_i(M)$ is the congestion window of flow i at the end of the M-th congestion epoch. Eventually, each flow i achieves a congestion window that is proportional to α_i. Indeed, the ratio of congestion window of the two flows approaches α_1/α_2 for large values of M. In order to see this, note that $c_1(M)/c_2(M)$ equals

$$
\begin{aligned}
&= \frac{c_1(M-1) + \alpha_1(n + \frac{m}{\sqrt{c_1(M-1)}})}{c_2(M-1) + \alpha_2(n + \frac{m}{\sqrt{c_2(M-1)}})} \\
&= \frac{\beta c_1(M-2) + \alpha_1(n + \beta n + \frac{m}{k_1} + \frac{\beta m}{\sqrt{c_1(M-2)}})}{\beta c_2(M-2) + \alpha_2(n + \beta n + \frac{m}{k_2} + \frac{\beta m}{\sqrt{c_2(M-2)}})} \\
&= \frac{\beta^2 c_1(M-3) + \alpha_1(n + \beta n + \beta^2 n + \frac{m}{a_1} + \frac{\beta m}{b_1} + \frac{\beta^2 m}{c_1})}{\beta^2 c_2(M-3) + \alpha_2(n + \beta n + \beta^2 n + \frac{m}{a_2} + \frac{\beta m}{b_2} + \frac{\beta^2 m}{c_2})}
\end{aligned}
$$

where $k_i = (\beta c_i(M-2) + \alpha_i \beta n / \sqrt{c_i(M-2)} + \alpha_i \beta n)^{1/2}$, $c_i = \sqrt{c_i(M-3)}$, $b_i = \sqrt{c_i(M-2)}$ and $a_i = k_i$ with $c_i(M-2)$ expanded to the next level. For $M = k$ the expression takes the same form as the above equation, the left operand of the addition operator becomes $\beta^{k-1} c_i(M-k)$ which approaches zero as k becomes large since $\beta < 1$. The multiplicative factor of α_i's can then be eliminated since they assume the same values. Hence, the above expression approaches α_1/α_2. Therefore, to allocate the bandwidth fairly among two flows, we scale the α parameter of each flow by its own RTT.

$$
\alpha_f = \alpha_s \cdot \left(\frac{rtt}{t_p}\right) = \alpha \cdot \left(\frac{rtt}{t_p}\right)^2 \tag{3.15}
$$

Note that VCP [56] uses similar parameter scaling but it employs the AIMD control law while MLCP uses the AI-II-MD control law.

3.4.2 MLCP Router

A MLCP router performs two functions: (1) it computes the load factor over an interval t_p and (2) it estimates the average RTT of flows to adapt the load factor measurement interval.

3.4.3 MLCP Receiver

The MLCP receiver is similar to a TCP receiver except that when acknowledging a packet, it copies the header information from the data packet to its acknowledgment.

3.5 PERFORMANCE EVALUATION

Our simulations use the packet-level simulator ns2 [96], which we have extended with an MLCP module. A diverse set of network scenarios are considered including a range of bottleneck capacities, round-trip times, number of long-lived flows, offered load of short-lived, web-like flows, heterogeneous RTTs, bottleneck buffer sizes, and performance evaluation under a multiple bottleneck topology. We explain the choice and range of these parameters in the next section, followed by a discussion on the performance metrics used for evaluation, and finally, we present and discuss the evaluation results.

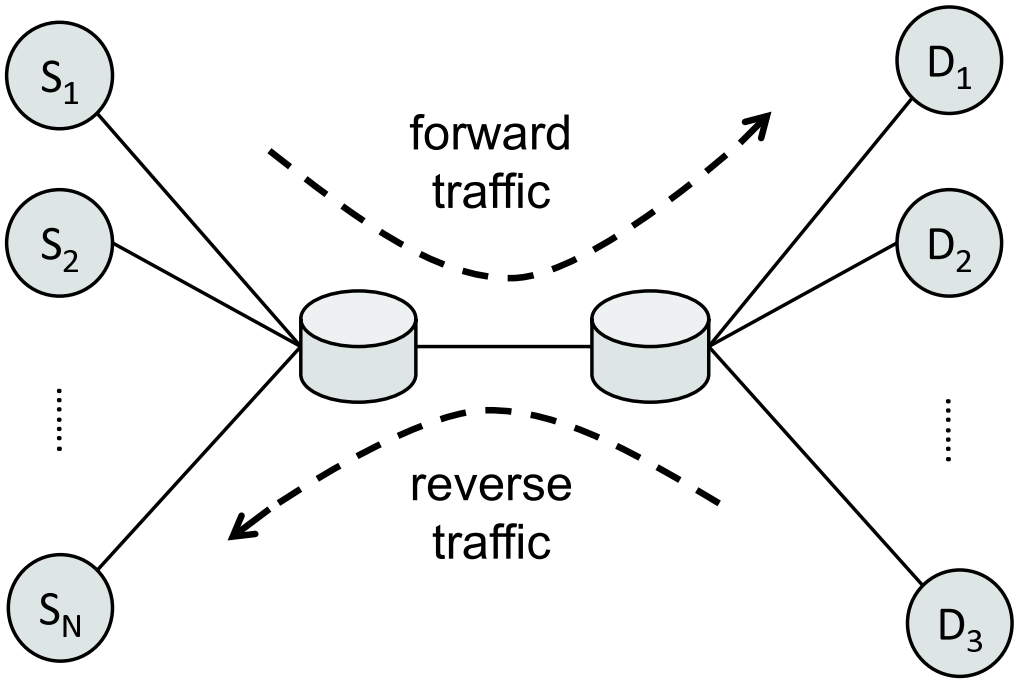


Figure 10: Dumbbell Topology

3.5.1 Network Parameters

To study the impact of various network parameters on the performance of MLCP, we first consider a basic network scenario. We then vary each network parameter in the basic scenario, one at a

Parameter	Link Capacity	Round-trip Delay	Number of Long-lived Flows
Range	100kbps-10Gbps	1ms-1s	1-1000
Parameter	Arrival Rate of Short Flows	RTT Heterogeneity	Buffer Size
Range	$1s^{-1}$-$1500s^{-1}$	(40ms,156ms)-(40ms, 3520ms)	1pkt-2000pkts

Table 4: Network parameters and the range of their values used in the evaluation

time, while keeping everything else fixed. The basic network scenario is a dumbbell topology with a single bottleneck link of capacity 200 Mbps. We set the RTT to 80 ms[12] where the forward and reverse path each has 5 FTP flows unless stated otherwise. Having 5 flows in the basic scenario allows us to evaluate the performance of MLCP under high per-flow bandwidth regimes, which are particularly problematic for congestion control protocols [6, 8].

Next, we vary the the bottleneck capacity in the basic network scenario from 100 kbps to 10 Gbps while keeping everything else fixed. This results in per-flow bandwidths ranging from 20 kbps to 2 Gbps, which covers the range of bandwidths for most existing technologies such as Dial-up modems, DSL, and Cable modems as well as higher capacity technologies that are likely to become available in the future. We then vary the round-trip propagation delay from 1 ms to 1 s while keeping everything else fixed in the basic scenario. While 1 ms is a typical round-trip delay within a LAN, Satellite links can have delays in the order of a second. This range allows us to evaluate all these scenarios. Next, we vary the number of long-lived, FTP-like flows from 1 to 1000. It is well-known that most bytes in the Internet come from long-lived flows [97]. Therefore, it is important to study the performance of new protocols across a range of long-lived flows. While most data comes from long-lived flows, most flows on the Internet are short [97]. Hence, we next add short, web-like flows to the basic scenario and vary their offered load from zero to C_l Mbps,

[12]This is roughly the delay between the East and the West coasts in the US

where C_l=155 Mbps, while keeping the rest of the parameters fixed.

We then extend the basic network scenario to consider a multiple bottleneck topology. In this topology, the capacity of each forward link is set to a value smaller than the capacity of the preceding link. 5 Flows (as in the basic network scenario) traverse all links in the forward direction whereas each of these links is also shared by a distinct set of 5 cross flows. Such a scenario stresses the protocols as they encounter competition on each forward link. We then investigate the impact of short-term dynamics on MLCP.

We then study the impact of heterogenous RTT flows by picking flow RTTs from a small ([40 ms, 156 ms]) as well as large ([40 ms, 3520 ms]) range of values. These ranges allow evaluation of RTTs that can differ by up to an order of magnitude. Finally, we evaluate the impact of bottleneck buffer size on the performance of MLCP. This is useful because technology trends indicate that it is challenging to have large buffers in all-optical networks and therefore it is important that future transport protocols perform well when the buffer size is small. Considering the basic scenario, we vary the bottleneck buffer size from 1 pkt to 2000 pkts (equal to the BDP of the path.)[13].

In our evaluation, TCP SACK is always used with RED and ECN enabled at the routers. The bottleneck buffer size is set to the BDP, or two packets per-flow, whichever is larger. The data packet size is 1000 bytes, while the ACK packet size is 40 bytes. All simulations are run for at least 100 s. The statistics neglect the first 5% of the simulation time.

3.5.2 Performance Metrics

To study the performance characteristics of MLCP, we consider a variety of performance metrics. These correspond to the properties outlined in Section 1 and includes metrics that are useful for both the network operators as well as the users. In particular, we consider the following metrics:

- *Link Utilization*: It is the fraction of time for which the link is utilized. Given a certain link capacity, higher utilization values mean higher throughput.
- *Average Queue Length*: It is the average queue length over a certain interval. We normalize this with the buffer size to obtain a value between 0% and 100%.

[13] Router buffers are often set to the bandwidth-delay product of the path to enable TCP flows to achieve 100% link utilization [98]

- *Packet Drop Rate*: It is the ratio of dropped packets to the total number of packets sent.
- *Jain's Fairness Index*: It is a metric that quantifies the fairness achieved by flows sharing a bottleneck link. It is equal to $(\sum_{i=1}^{N} x_i)^2/(N\sum_{i=1}^{N} {x_i}^2)$, where x_i is the throughput of flow i and $i \in \{1, .., N\}$. It a value between 0 and 1. When all flows achieve the same throughput on a bottleneck link, the index equals 1. If only *K* of *N* flows receive bandwidth (and others none), index is *K/N*.

3.5.3 Single Bottleneck Topology

We first evaluate the performance of MLCP for the case of a single bottleneck link shared by multiple MLCP flows over the dumbell topology shown in Figure 10. We vary each network parameter in the basic scenario, one at a time, while keeping everything else fixed. In the basic scenario, the bottleneck link rate is set to 200 Mbps, whereas all other links have a capacity of 2000 Mbps. The round-trip propagation delay is set to 80 ms where the forward and reverse path each has 5 FTP flows.

3.5.3.1 Impact of Bottleneck Capacity MLCP achieves high utilization across a wide range of bottleneck capacities as shown in Figure 11. VCP, on the other hand, becomes inefficient at high bottleneck capacities. The utilization gap between MLCP and VCP starts widening when the bottleneck capacity is increased beyond 10 Mbps. This difference becomes more than 60% on a 10 Gbps link. VCP's performance degrades because it uses a fixed MI factor of value 1.0625, which is too conservative for high link capacities. On the contrary, MLCP adapts its MI factor, increasing far more aggressively in low utilization regions, allowing it to remain efficient on high capacity links. Utilization with TCP SACK remains considerably lower than that of MLCP and VCP. This happens because TCP uses a conservative increase policy of one packet/RTT and an aggressive decrease policy of halving the window on every congestion indication, leading to inefficiency on high BDP paths.

The average queue length for MLCP remains close to zero as we scale the link capacities. However, for very low capacities (e.g.,100 Kbps), MLCP results in an average queue length of about 20% despite keeping zero loss rate. This happens because the value of α is high for such

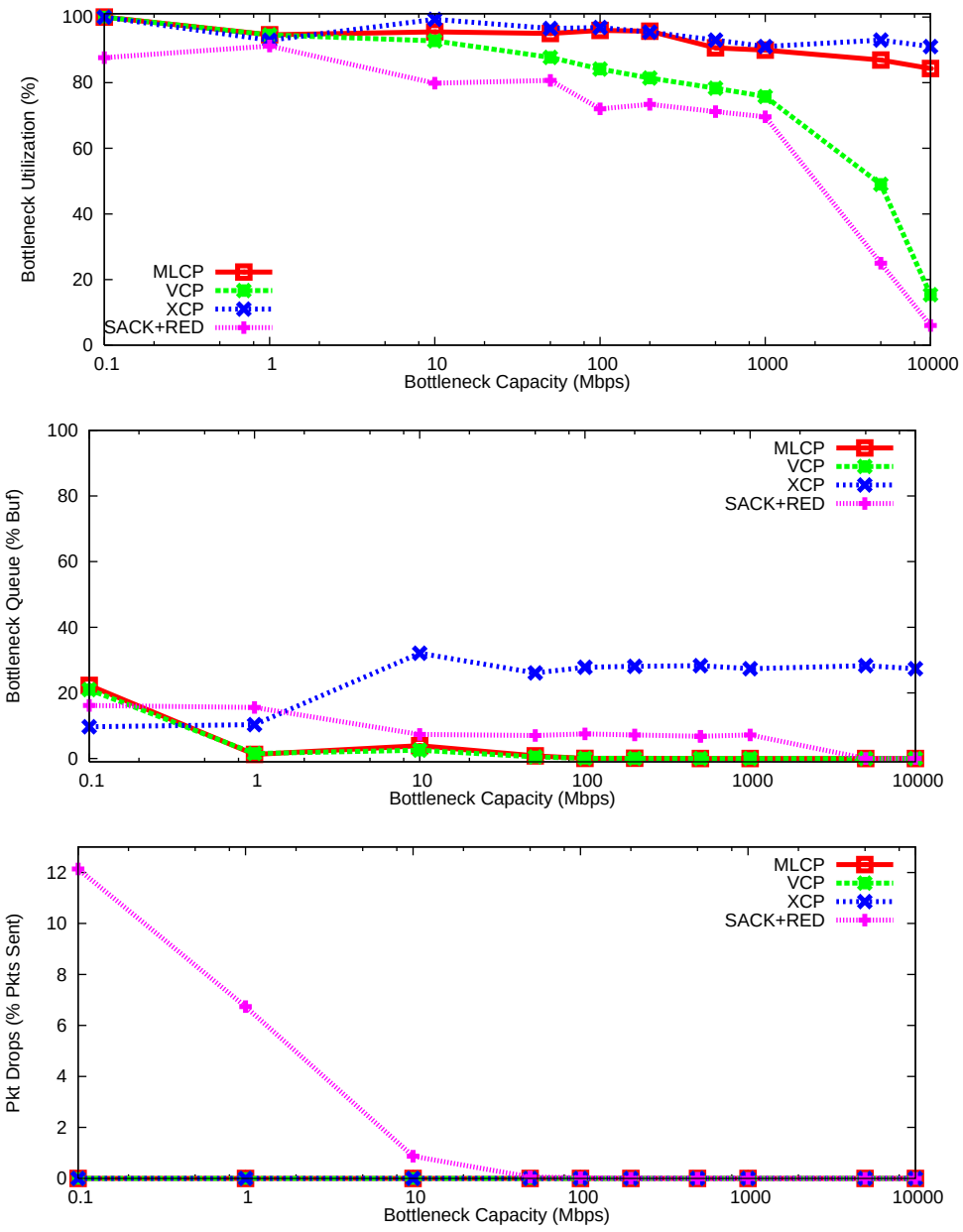


Figure 11: One bottleneck with capacity varying from 100 Kbps to 10 Gbps (Note the logarithmic scale on the x-axis).

capacities which leads to queue buildup. Note that while MLCP achieves roughly the same utilization as XCP, it is able to maintain a lower average bottleneck queue for link capacities $\geq$2 Mbps. Packet loss rate with VCP and XCP also remains close to zero whereas SACK results in loss rates

that are as high as 12% for low capacities.

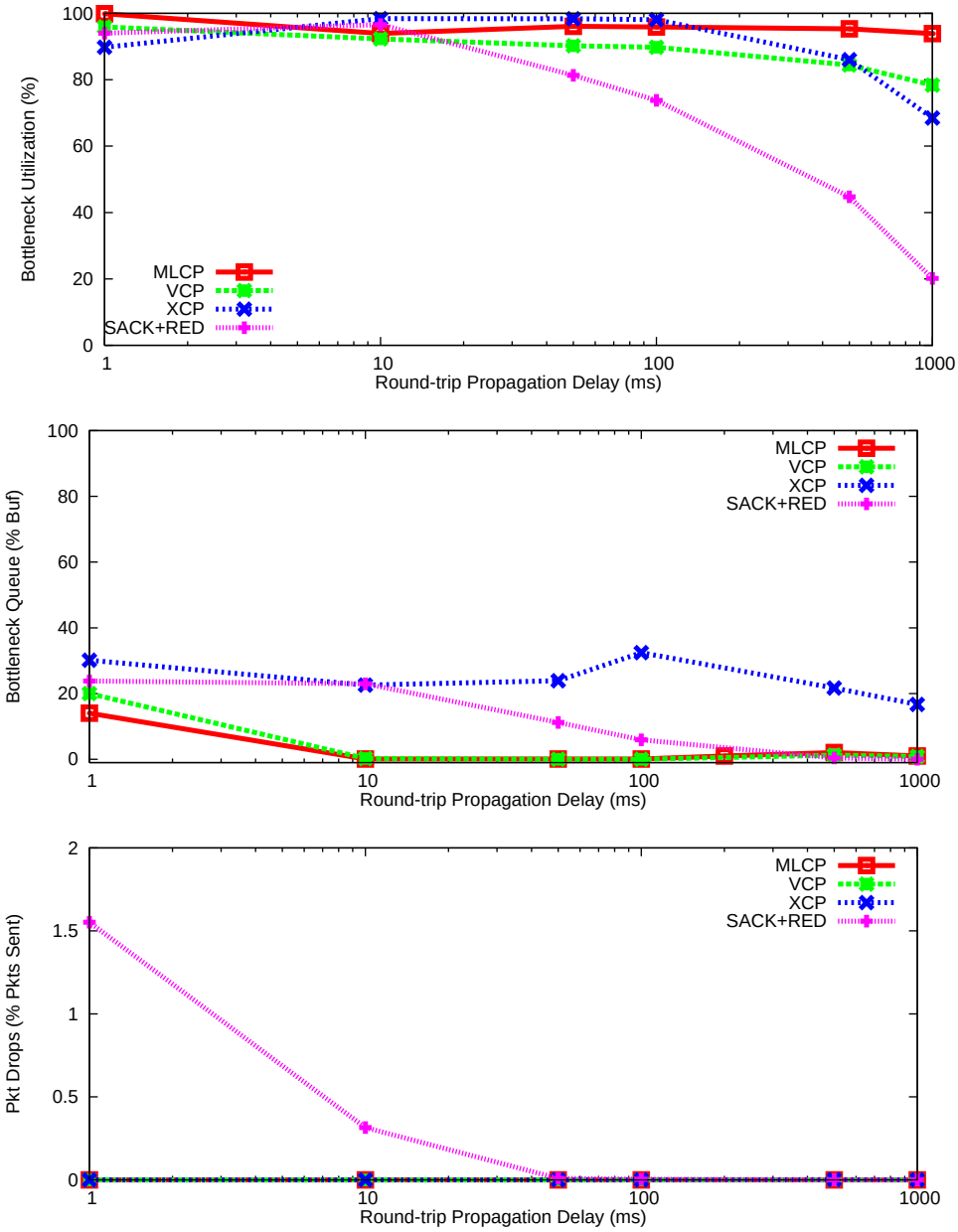


Figure 12: One bottleneck with round-trip propagation delay ranging from 1 ms to 1 s (Note the logarithmic scale on the x-axis).

3.5.3.2 Impact of Feedback Delay We fix the bottleneck capacity to 200 Mbps and vary the round-trip propagation delay from 1 ms to 1 s. As shown in Figure 12, MLCP scales better than

VCP, XCP, SACK+RED, and SACK+RIO. For delays larger than 100 ms, the utilization gap between MLCP and VCP increases from roughly 5% to more than 40%. With SACK+RED, utilization drops most rapidly as delays are increased. The difference between MLCP and SACK+RED increases from 20% for 100 ms to more than 60% for 1 s. Note that the average queue length remains less than 15% for MLCP across the entire RTT range. These results indicate that MLCP could be effectively used in long-delay satellite networks.

3.5.3.3 Impact of Number of Long-lived Flows It is well-known that most bytes in the Internet come from long-lived flows [97]. Therefore, it is important to study the performance of new protocols across a range of long-lived flows. We now vary the number of long-lived flows (in both directions) and study its impact.

Figure 13 shows that as we increase the number of long-lived flows (in either direction), MLCP is able to maintain high utilization ($\geq$90%), with negligible average queue length and near-zero packet drop rate. For small flow aggregates [1-50], TCP SACK's utilization remains lower than that of MLCP, VCP and XCP (due to larger available per-flow bandwidth), whereas the difference between them grows to as large as 20%. SACK results in higher average queue length than MLCP and VCP. Loss rate for SACK, however, increases to only as high as 6%. This relatively low loss rate for SACK is a consequence of using RED with ECN enabled at the routers. When the number of flows is less than five, MLCP achieves a higher average utilization than XCP. However, as the number of flows is increased, XCP and MLCP achieve similar average utilization. Note that MLCP has a much lower average queue size compared to XCP even though they have similar loss rates.

3.5.3.4 Impact of Short-lived, Web-like Traffic While most data comes from long-lived flows, most flows on the Internet are short [97]. Hence, we now add short, web-like flows to the basic scenario and vary their offered load, while keeping the rest of the parameters fixed. These flows arrive according to a Poisson process, with an average arrival rate varying from 1/s to 1000/s. Their transfer size obeys the Pareto distribution with an average of 30 packets. This setting is consistent with the real-world web traffic model [97]. Figure 14 illustrates the performance of MLCP in comparison to VCP, XCP and TCP SACK. When the arrival rate is less than 1000/s, MLCP achieves higher utilization than VCP, XCP and TCP SACK. However, note that XCP and VCP achieve more

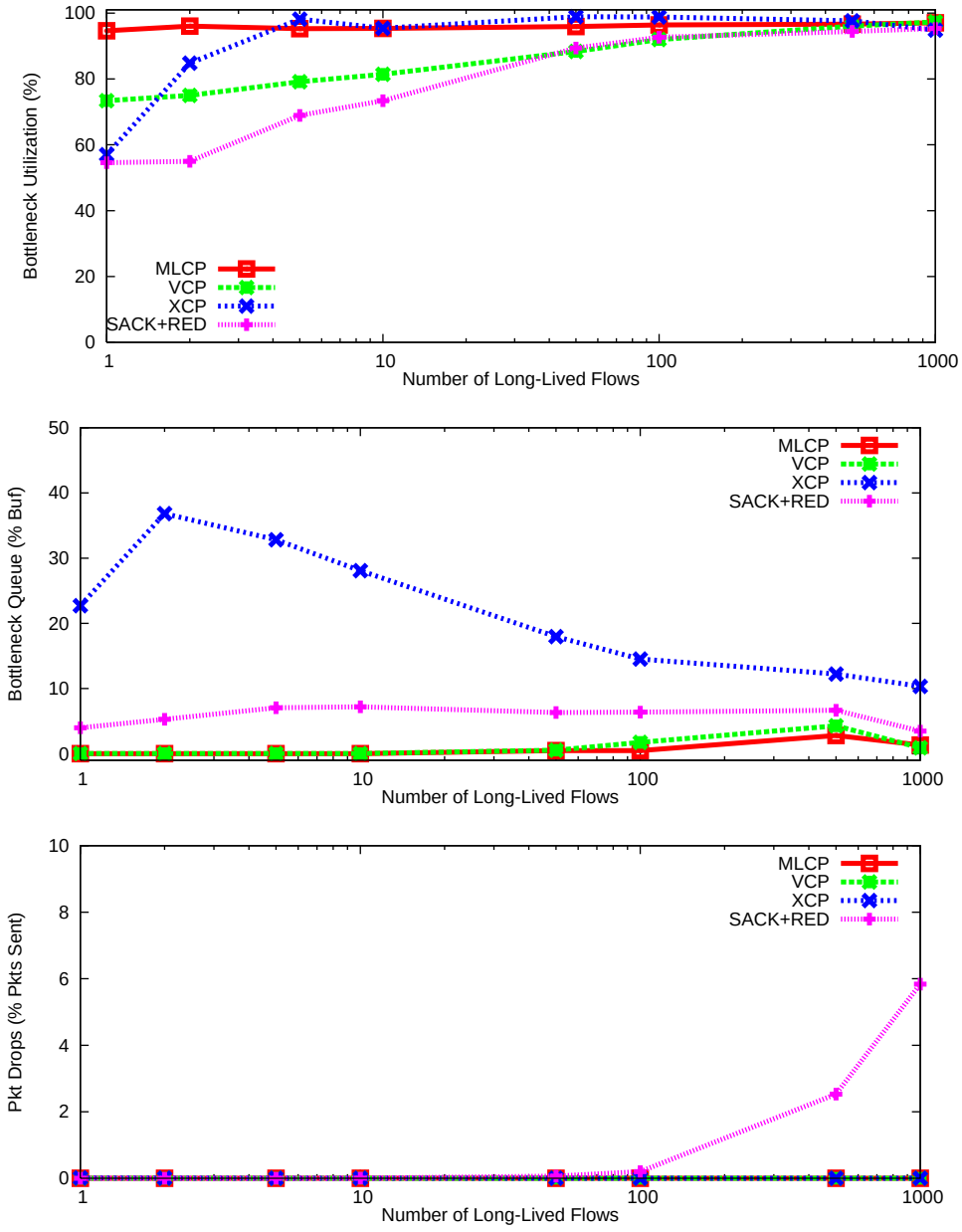


Figure 13: One bottleneck with the number of long-lived, FTP-like flows increasing from 1 to 1000 (Note the logarithmic scale on the x-axis).

than 80% in all cases. When the arrival rate is increased beyond 1000/s, loss rate for VCP and XCP increases almost linearly to 10% and 7%, respectively. The average queue length for VCP and XCP rises to about 90% and 80% of the buffer size, respectively. This illustrates VCP's low responsive-

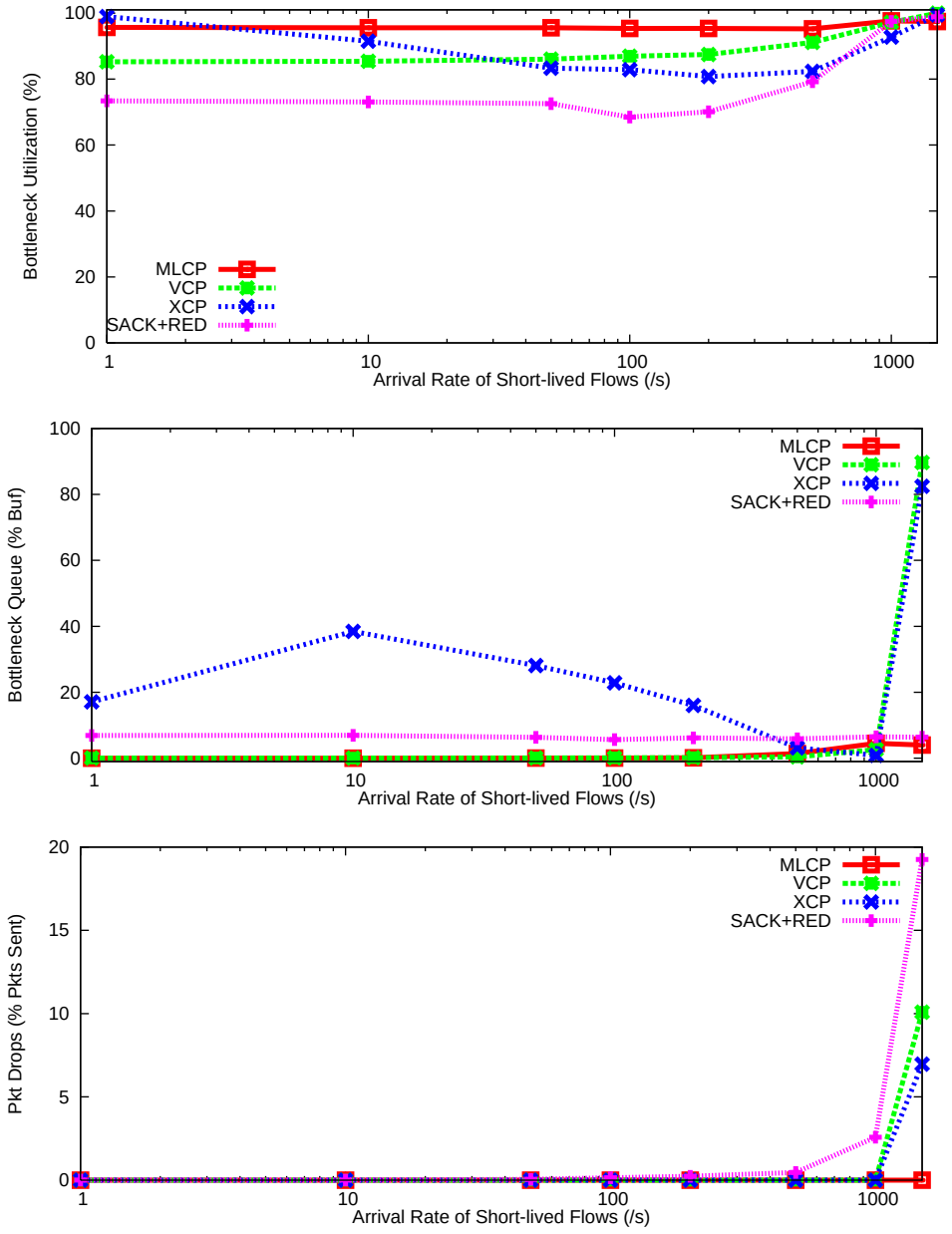


Figure 14: One bottleneck with short-lived, web-like flows arriving/departing at a rate from 1/s to 1500/s

ness to high congestion; a consequence of using a single, high value of $\beta = 0.875$. MLCP, on the hand, is able to maintain almost 100% utilization, with negligible average queue length and near zero packet drop rate even under heavy congestion. Using multiple levels of MD allows MLCP to

be more aggressive in its decrease policy than VCP, resulting in high responsiveness to congestion. Moreover, the AI parameter setting in VCP is too large when the link is heavily congested. MLCP, on the hand, applies II after the load factor exceeds 95%, which tends to lower the rate at which flows increase their rates. TCP SACK results in low link utilization when the arrival rate is smaller than 500/s. However, as the arrival rate increases, the traffic becomes more bursty due to many flows being in slow-start which causes packet losses to increase.

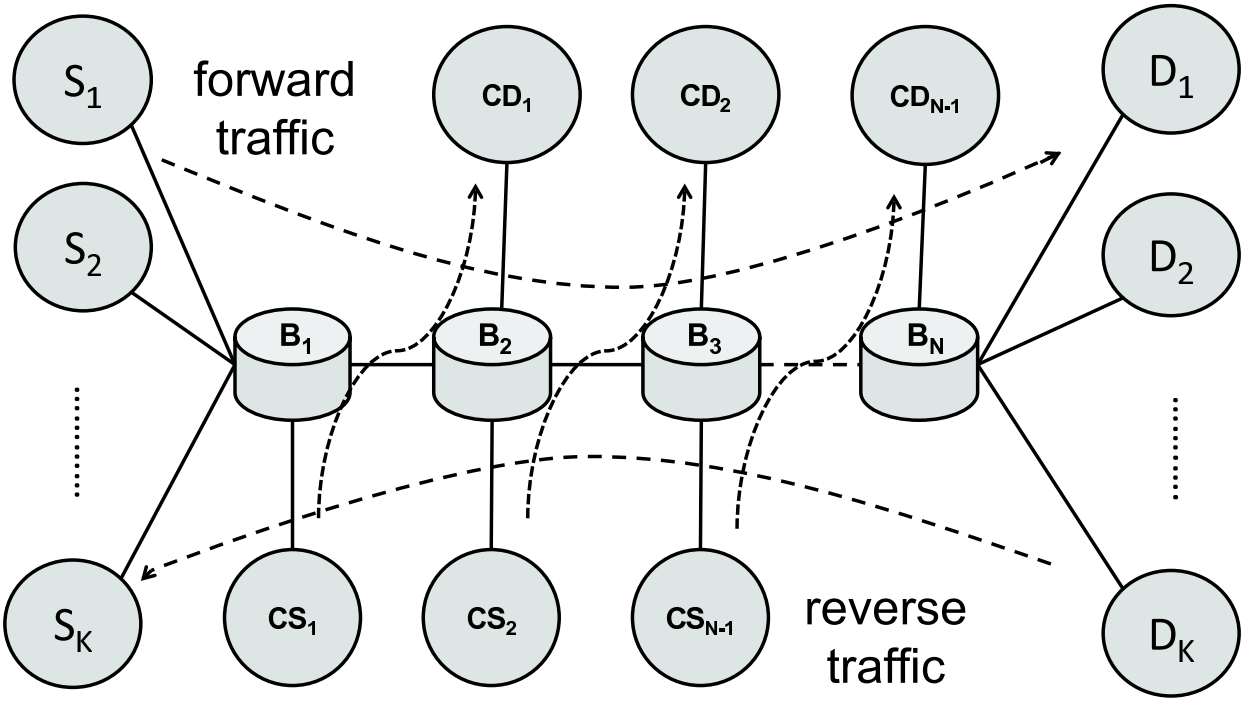


Figure 15: Parking-lot topology

3.5.4 Multiple Bottleneck Topology

Next, we study the performance of MLCP with a more complex topology of multiple bottlenecks. For this purpose, we use a parking-lot topology with 10 bottlenecks as shown in Figure 15. The $B_1 - B_2$ link has capacity 200 Mbps, and each of the following links has capacity which is 10 Mbps smaller than the previous one. This results in a capacity of 110 Mbps for the $B_{10} - B_{11}$ link. The propagation delay of each link is set to 20 ms. There are 30 long-lived FTP flows traversing all the

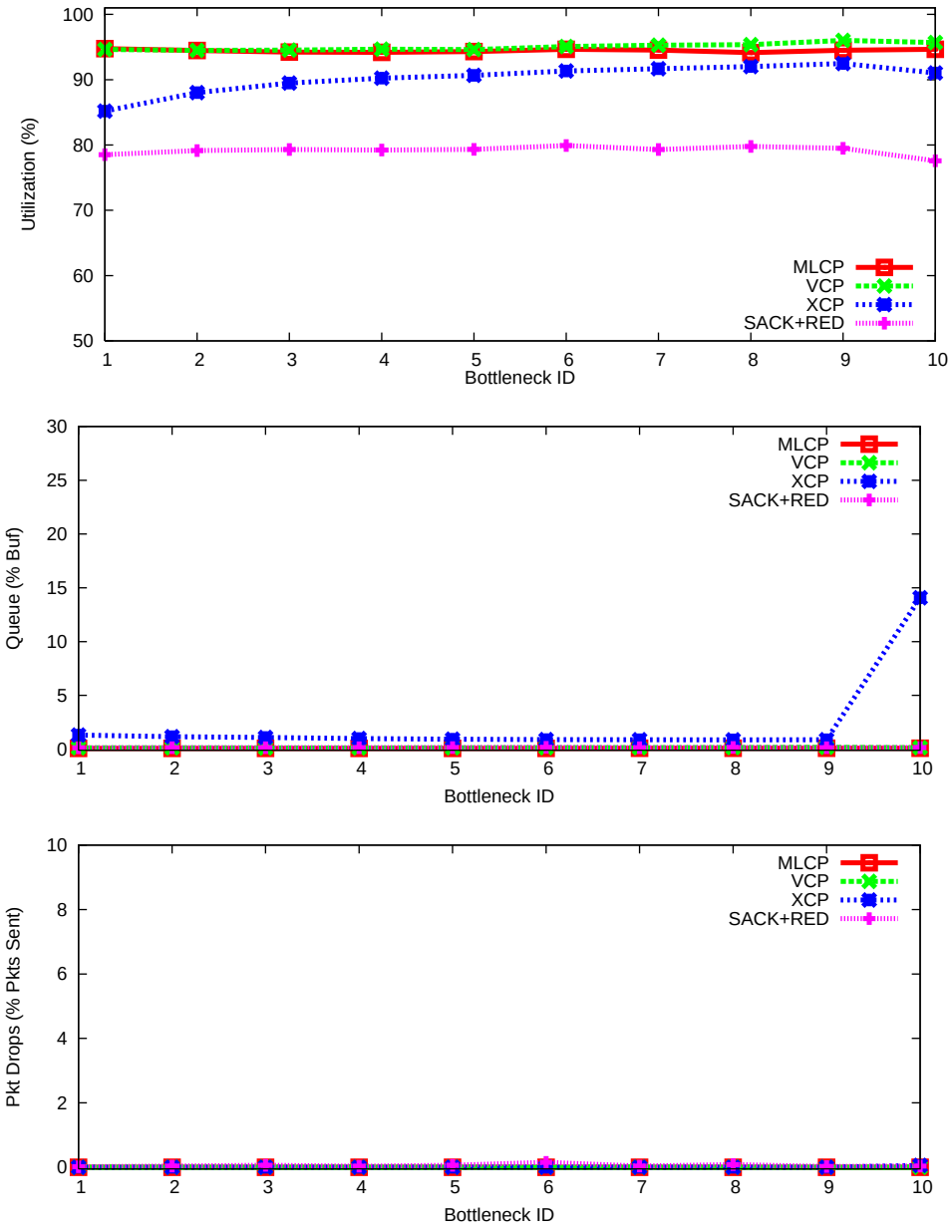


Figure 16: Multiple congested bottlenecks

links in the forward direction, and 30 FTP flows in the reverse direction. In addition, each link has 5 cross FTP flows traversing the forward direction. The round-trip propagation delay that traverses all links in the forward direction is 80 ms (the same as in the basic scenario) whereas for the cross

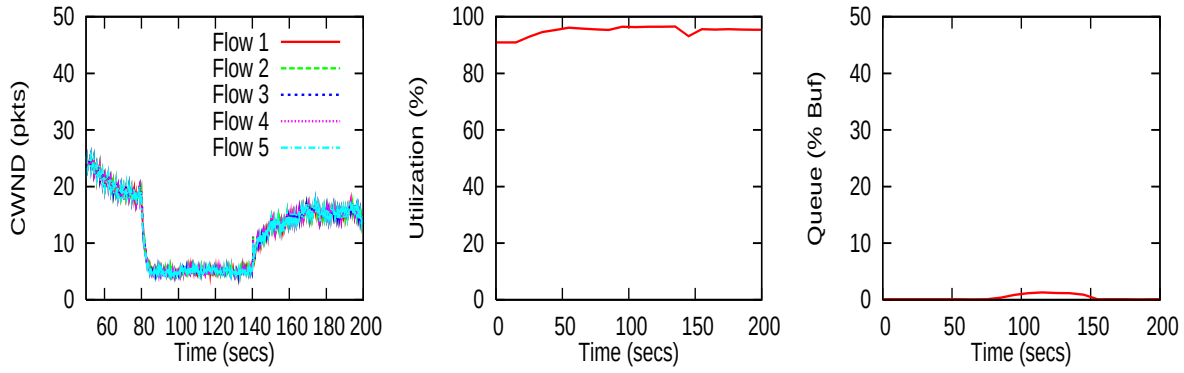


Figure 17: MLCP is robust against and responsive to sudden, traffic demand changes.

flows it is set to 20 ms[14]. These settings depict a scenario where a set of long RTT flows compete with a set of short RTT flows at each link and each set of flows have different bottlenecks.

Figure 16 shows that MLCP achieves higher utilization when compared with SACK and XCP on all the 10 bottleneck links whereas the performance is similar to that of VCP. In particular, MLCP achieves about 25% higher utilization than SACK on all bottlenecks. When compared with XCP, MLCP achieves utilization that is as high as 10%. Observe that XCP achieves about 85% utilization on the first link whereas on the following links, its utilization increases as link capacity decreases. This happens because XCP enabled-routers independently compute per-flow bandwidth, which means that if a flow is bottlenecked at a downstream link, an upstream router would still attempt to allocate bandwidth to that flow to ensure local fairness. This leads to link under-utilization [99]. The average queue length remains less than 2% of the buffer size for all protocols except for XCP. In case of XCP, the queue length to about 15% on the most constrained link. Note that all protocols achieve near-zero packet loss rate on all the links.

3.5.4.1 Dynamics All the previous simulations focus on the steady-state behaviour of MLCP. Now, we investigate its short-term dynamics using the multiple bottleneck topology considered in the previous section .

[14]The difference in the RTT of forward flows and cross flows allows us to evaluate the impact of RTT heterogeneity on the performance of MLCP.

Sudden Demand Changes: To study the behaviour of MLCP when the demand at the links changes suddenly, 100 new forward FTP flows are made active at $t = 80\,s$ and they leave at $t = 140\,s$[15]. Figure 17 shows that MLCP can quickly adapt to sudden fluctuations in the traffic demand (The figure draws the congestion window dynamics for the five forward MLCP flows). When new flows enter the system, the flows adjust their rates to the new fair share while maintaining the link at high utilization. At $t = 140\,s$, when 100 flows depart creating a sudden drop in the utilization, the system quickly discovers this and ramps up to almost 100% utilization within a couple of seconds. Note that both forward and cross flows utilize this available bandwidth. Observe that during the adjustment period the bottleneck queue remains low. The result shows that MLCP is very responsive to sudden variations in the available bandwidth.

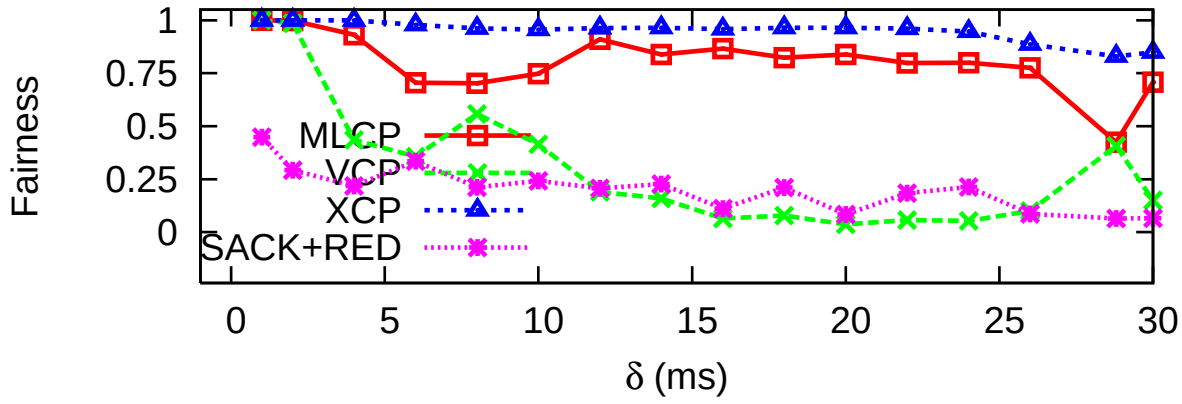


Figure 18: Jain's fairness index $\{(\sum_{i=1}^{N} x_i)^2/N \cdot \sum_{i=1}^{N} x_i^2$ for flow rates x_i, i∈[1,N]$\}$ under scenarios of one bottleneck link shared by 30 flows, whose RTT are in the ranges varying from [40 ms, 156 ms] to [40 ms, 3520 ms]

3.5.5 Fairness

We now compare the fairness properties of MLCP, VCP and TCP SACK. We consider 30 FTP flows (in both directions) in the basic scenario of a single bottleneck link with capacity 200 Mbps[16].

[15]The 100 flows traverse all the links in the forward direction

[16]30 flows are chosen so that their mix contains a large range of flow RTTs

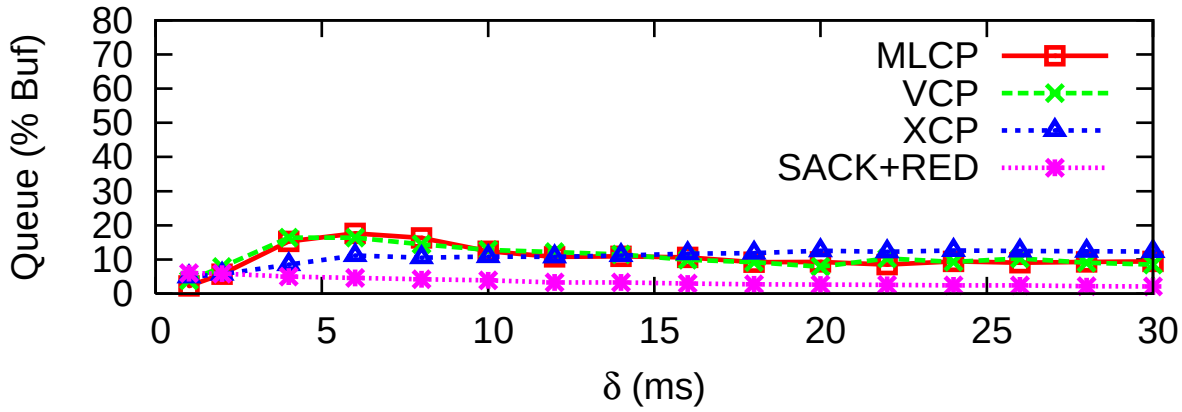


Figure 19: Bottleneck queue as a function of the RTT variation

Each forward flow j's RTT is chosen according to $rtt_j = 40 + j * 4 * \delta$ ms for $j = 0, .., 29$, and where δ is the one-way propagation delay for a non-bottleneck link. We perform simulations with δ varying from 1 ms to 30 ms. When δ is 1 ms, RTTs are in the range [40 ms,156 ms]. When $\delta = 30$, the RTTs are in the range [40 ms,3520 ms]. Figure 18 shows the comparison of the fairness achieved by MLCP, VCP, XCP and TCP SACK. While XCP achieves the highest level of fairness across a large range of RTT variations, MLCP, on the hand, achieves good fairness ($\geq$0.75, $\forall\delta$) while maintaining $<$20% average queue length (see Figure 19). With VCP and TCP SACK, fairness decreases considerably as the network incorporates more diverse RTT flows. In the case of VCP, this occurs due to the bounding of the MI and AI gains. With TCP, flows achieves RTT-proportional sending rates, therefore, large RTT flows achieve much smaller throughput than small RTT flows. This reduces the overall fairness for TCP SACK.

3.5.6 Impact of Buffer Size

Recent advances in technology suggest that all-optical networks are likely to become commonplace in the future. However, optical packet buffers can only hold a few dozen packets in an integrated opto-electronic chip. Larger all-optical buffers remain infeasible, except with unwieldy spools of optical fiber (that can only implement delay lines, not true FCFS packet buffers) [100]. In order to

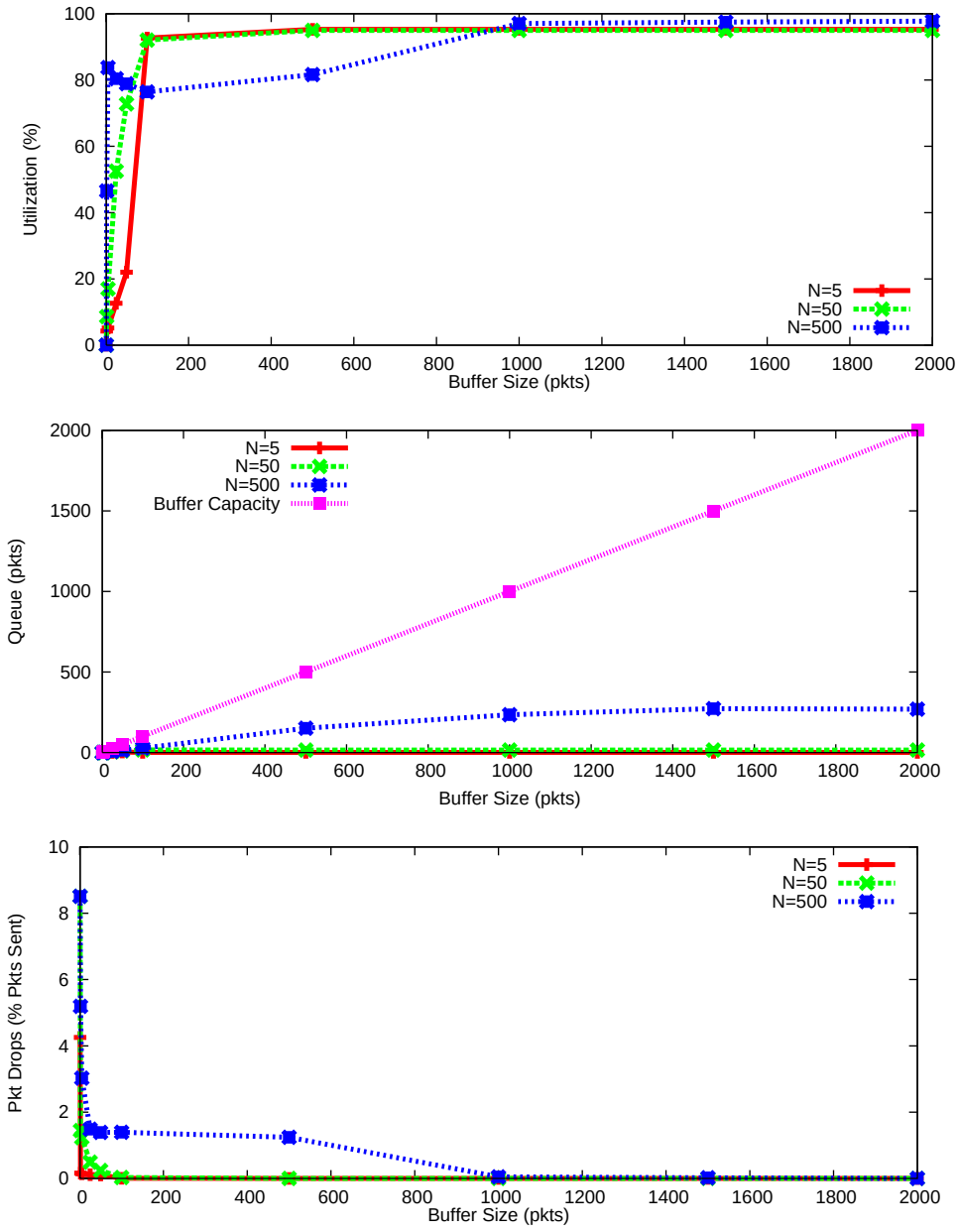


Figure 20: One bottleneck with C=200 Mbps, RTT=80 ms and the number of long-lived flows varying from 5 to 500.

make TCP amenable to small buffers, researchers have suggested using paced TCP [100], proposed changes in the AIMD parameters [101] and proposed new protocols to make it friendly with small buffers [102]. Dhamdhere et al. [103] showed that with small buffers (sized according to the

square-root rule [98]) TCP observes loss rates as high as 15% on congested Internet links. The key problem with TCP is the coupling of packet loss with congestion indication that creates self-induced losses. MLCP decouples congestion indication from packet loss and thus only requires packet buffers for absorbing short-term packet bursts.

In this section, we analyze the performance of MLCP with small buffers. We consider the basic scenario with a 200 Mbps bottleneck link and a round-trip propagation delay of 80 ms. This gives a BDP of 2000 packets where each packet has a size of 1000 bytes. We vary the buffer size from 1 pkt to 2000 pkts and the number of long-lived flows, N, from 5 to 500 yielding a range of per-flow bandwidths in [400 Kbps, 40 Mbps]. Figure 20 shows that MLCP needs a buffer size of less than 100 packets ($\approx$5% of the BDP of the path) to achieve more than 80% utilization and achieve a loss rate of less than 2% across values of N that differ by an order of magnitude. For $N = 5$, when the buffer size is $\leq$50 packets, MLCP achieves less than 50% utilization. This happens because MLCP's aggressive MI policy results in bursty packet losses which decreases utilization. As N is increased, the per-flow BDP of the path decreases which reduces the buffer size requirement for achieving high utilization. Hence for $N = 500$, MLCP needs a buffer size of $\approx$5 packets to achieve 80% utilization. As we increase the buffer size, utilization improves, the average queue length stabilizes to a value much less than the BDP of the path and the loss rate decreases sharply. Note that the average queue size remains less than 250 packets ($<$12.5% of the BDP of the path) even in the most congested scenario of 500 flows. These results indicate that MLCP can achieve high performance even with as small as 100 packet buffers.

3.6 STABILITY ANALYSIS

MLCP employs an aggressive load factor guided MI control law, which tracks the available bandwidth exponentially fast. This naturally raises stability concerns[17]. In this section, we observe that the fluid approximation model of the traffic presented in [56] can be used to show that such a control law does not result in instability when used in conjunction with a load factor dependent MD.

[17]Note that the results presented in Section 4.6 demonstrate the stability of MLCP across a wide range of network scenarios using simulation.

The fluid model considers a single link shared by multiple flows and is described by the following differential equation:

$$\dot{w}_i(t) = \frac{1}{RTT} \cdot [w_i(t) \cdot \xi(\sigma(t)) + \alpha] \tag{3.16}$$

where $w_i(t)$ is the congestion window of flow i at time t and $\xi(\sigma(t)) = (1 - \sigma(t))/\sigma(t)$ and α are the MI and AI parameters, respectively.

The above differential equation models (1) the load-factor guided MI control law (characterized by $\xi(\sigma(t))$), (2) the AI control law (characterized by α), and (3) the MD control law whose parameter value depends on the load factor (characterized by $\xi(\sigma(t))$ for $\sigma(t) \geq 1$) and thus models the impact of multiple back-off factors. All of these features are used by MLCP. The stability conditions of the above model are given by the following theorem[18]:

Theorem 3.6.1. Under the model given by Equation 3.16, where a single bottleneck is shared by a set of synchronous flows with the same RTT, if $\kappa \leq \frac{1}{2}$, then the delayed differential equation described in [56] is globally asymptotically stable with a unique equilibrium $w^* = \gamma C \cdot RTT + N\frac{\alpha}{\kappa}$, and all the flows have the same steady-state rate $r_i^* = \frac{\gamma C}{N} + \frac{\alpha}{\kappa \cdot RTT}$

The above result holds for any link capacity, feedback delay, and number of flows. Moreover, the global stability result does not depend on the network parameters. It demonstrates that the use of the load factor guided MI, AI and multiple back-off factors does not cause instability as long as $\kappa \leq \frac{1}{2}$ for the case of a single bottleneck link. Intuitively, as the load factor at the bottleneck approaches 1, $\xi(\sigma(t))$ approaches zero, causing sources to become less aggressive as the available bandwidth decreases. When $\sigma(t)$ exceeds one, sources backoff in proportion to the amount of overload. This helps in keeping low persistent queue length. Note that this is unlike TCP's unguided exponential increase during slow-start, which becomes unstable as capacity or delay increases [104, 31].

The above model makes some simplifications in order to make the analysis tractable and differs from MLCP in the following ways: First, it uses MI and AI together at any given time. MLCP, on the other hand, uses either MI, AI, or II at a given time. In MLCP, AI results in a faster

[18]For the proof of the model, we refer the reader to [56]

window growth than II because $\alpha \geq \frac{\alpha}{w_i(t)}$, $\forall w_i(t) \geq 1$, therefore, the MLCP window growth is less aggressive than the growth given by the above differential equation. Second, it uses the exact load factor, whereas MLCP uses a quantized value of the load factor to choose the control laws and their parameters values.

3.7 RELATED WORK

Chiu et al. [45] studied the efficiency and fairness properties of the AIMD, AIAD, MIAD, and MIMD control laws in the presence of 1-bit load feedback. On the other hand, we study the marginal utility of increased bits for feedback while considering the efficiency and fairness properties of the MI-AI-II-MD control law. Xia et al. [56] proposed a 2-bit feedback scheme called VCP. They use the load feedback to choose between different control laws. However, we provide a general treatment of the interplay between performance and feedback. In that sense, our work can be seen as a generalization of their work. With regard to the protocol, VCP differs from MLCP in four ways: (a) MLCP uses 4-bits for feedback instead of two. This allows MLCP to obtain near-optimal performance in terms of rate of convergence to efficiency and fairness. (b) VCP employs load factor estimates to choose between different control laws but uses fixed parameters for them. On other hand, MLCP uses load factor estimates to choose the parameters of different control laws as well. (c) VCP uses a fixed t_p, which presents a trade-off between fairness and low queues, VCP chose the latter. MLCP, on the other hand, adapts t_p, which allows it to remain fair in the presence of diverse RTT flows while maintaining low queues and (d) VCP uses AIMD in steady-state, whereas MLCP employs AI-II-MD. This has two benefits. First, II enables smooth rate variations while improving fairness. Second, it considerably increases robustness to congestion.

3.8 SOFTWARE

The ns2 implementation for MLCP is publicly available for download at the following URL: http://www.cs.pitt.edu/~ihsan/mlcp-0.1.tar.gz.

3.9 SUMMARY

In this chapter, we analyzed the trade-off between increasing the amount of feedback information and the resulting performance improvements for load factor based congestion control protocols. We showed that while 2-bit scheme is far from optimal, using 3 bits is sufficient for achieving near-optimal performance in terms of rate of convergence to efficiency. We also showed that introducing multiple levels of MD allows a load factor based congestion protocols to achieve high rate of convergence to fairness, smooth rate variations and increased robustness to congestion. Using these fundamental insights we designed a low-complexity protocol that achieves efficient and fair bandwidth allocations, minimizes packet loss and maintains low average queue length in high BDP networks. A fluid model of the protocol showed that the protocol remains globally stable for the case of single bottleneck link shared by identical RTT flows.

4.0 DESIGN OF AN EFFICIENT FRAMEWORK FOR CONGESTION CONTROL

Many transport protocols that use explicit feedback from the network require more bits for feedback than are available in the IP header such as XCP [40] (128 bits), RCP [8] (96 bits) and MLCP [57] (4 bits). In case of MLCP, we showed in the previous chapter that at least 4-bit quantization of load factor is needed to achieve optimal convergence speed in terms of fairness and efficiency. Changing the IP header requires a non-trivial and a time consuming standardization process. Therefore, there is a need to explore techniques for obtaining high resolution congestion estimates using the existing ECN bits. Further, it is important to investigate whether a congestion control protocol based on such schemes can meet the desirable goals of a transport protocol.

In this chapter, we present a framework for congestion control, called Binary Marking Congestion Control (BMCC) for high bandwidth-delay product networks. The basic components of BMCC are $i)$ a packet marking scheme for obtaining high resolution congestion estimates using the existing bits available in the IP header for Explicit Congestion Notification (ECN) and $ii)$ a set of load-dependent control laws that use these congestion estimates to achieve efficient and fair bandwidth allocations on high bandwidth-delay product networks, while maintaining a low persistent queue length and negligible packet loss rate. We present analytical models that predict and provide insights into the convergence properties of the protocol. Using extensive packet-level simulations, we assess the efficacy of BMCC and perform comparisons with several proposed schemes. BMCC outperforms VCP, MLCP, XCP, SACK+RED/ECN and in some cases RCP, in terms of average flow completion times for typical Internet flow sizes.

With BMCC, each router periodically computes the load factor (ratio of input traffic and queue length to capacity) on its output links. To achieve efficient and fair bandwidth allocation with high convergence speeds, a high resolution estimate of the computed load factor is needed. BMCC uses

a packet marking scheme called Adaptive Deterministic Packet Marking (ADPM) [105] to obtain congestion estimates with up to 16-bit resolution using the existing two ECN bits. ADPM conveys signals with a lower Mean Square Error (MSE) than competitors such as REM [76] and RAM [106]. It monitors side information, such as the IPid field, and uses that to interpret the ECN bit of each packet differently, as proposed by Thommes and Coates [107]. Each arriving packet at the receiver carries a bound on the value of the load factor at the bottleneck. The receiver's estimate of the load factor is updated whenever a new packet provides a tighter bound. The load factor estimate is echoed back to the sources via acknowledgement packets using TCP options. Based on this value, sources apply load-dependent Multiplicative Increase (MI), Additive Increase (AI) and Multiplicative Decrease (MD). Unlike in VCP, the parameters of these control laws depend on the high resolution estimate of the load factor. When the load factor is small, sources increase their rates rapidly to fill the bottleneck capacity. Otherwise, sources apply AIMD to achieve fairness. The rates of adjustment are chosen to ensure RTT fairness.

To illustrate the convergence and fairness properties of BMCC, consider a simple network scenario with one bottleneck link of capacity 100 Mbps. Figure 21 shows the throughput of three BMCC flows that arrive with an inter-arrival time of 100 s and have round-trip propagation delays of 150 ms, 200 ms and 250 ms, respectively. Observe that the three flows rapidly achieve rates that are within 70% of their fair share before converging to a fair (≈50 Mbps each) and efficient (≈100 Mbps) bandwidth allocation. Figure 21 also shows the corresponding load factor at the bottleneck and the flows' estimates. ADPM allows flows to quickly track changes in load factor at the bottleneck.

BMCC's use of load factor as a signal of congestion enables it to decouple loss recovery from congestion control. This facilitates distinguishing error losses from congestion-related losses, which is important in wireless environments. Furthermore, the scale-free nature of load factor allows it to be encoded using few bits.

Using extensive ns2 simulations, we show that BMCC achieves efficient and fair bandwidth allocation while minimizing packet loss, bottleneck queue and AFCT. We present analytical models which provide insights into the performance of BMCC. These insights are likely to lead to better designs for next-generation congestion control protocols. We also evaluate mechanisms for reduc-

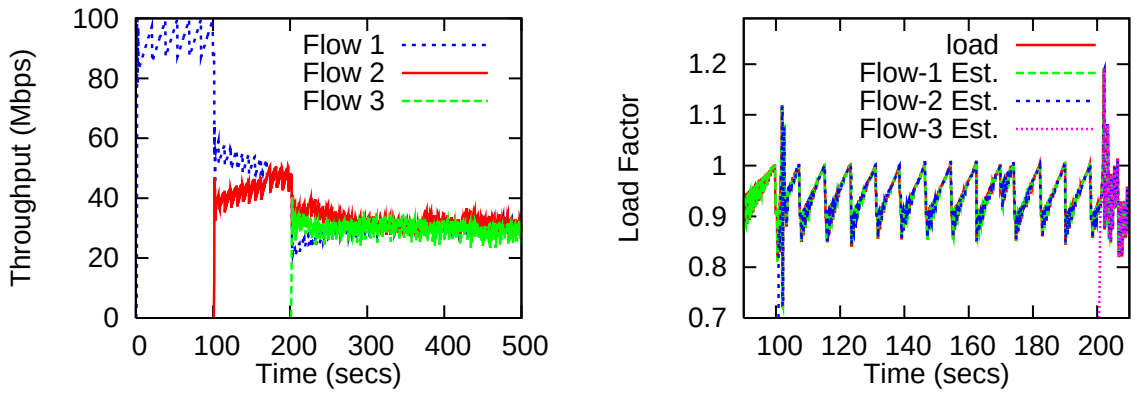


Figure 21: Comparison of the load factor at the bottleneck and the flows' estimates of it

ing the overhead of TCP options in conveying the load factor estimates from the receiver back to the sender. The resulting algorithm, employed by BMCC, introduces little overhead and is robust to lost ACKs.

The rest of the chapter is organized as follows. Section 4.1 describes the control laws and algorithms used by BMCC. Some design issues related to BMCC are addressed in Section 4.2. This is followed in Section 4.3 by a model for understanding the convergence properties of BMCC. Section 4.4 analyzes the impact of ADPM on convergence and Section 4.5 studies ways to reduce the overhead of using TCP options. Section 4.6, compares the performance of BMCC with that of other schemes by simulation.

4.1 DESIGN OF THE BINARY MARKING CONGESTION CONTROL (BMCC) PROTOCOL

BMCC uses ADPM to estimate the load factor f on the most congested link, and then uses the estimate, $\hat{f}$, to adjust the send window, w. Let us first consider how f is computed at the routers, then how the estimate $\hat{f}$ is determined, and finally how $\hat{f}$ is used by the senders. The values of the unspecified parameters will be discussed at the end of this section.

4.1.1 BMCC Router

During every time interval t_p, each router computes the load factor on each of its output links as

$$f = \frac{\lambda + \kappa_1 q_{av}}{\gamma_l C_l t_p} \tag{4.1}$$

where λ is the amount of traffic received during t_p, C_l is the capacity of the link and $\gamma_l \leq 1$ is the target utilization. Also, q_{av} is the exponentially weighted moving average queue length, using a time-constant of $8t_p$, and $\kappa_1 = 0.75$ controls how fast to drain the queue [56, 76, 77, 59].

4.1.2 BMCC Receiver and ADPM

The router conveys its load factor to the sender by applying ADPM [105] to the ECN bits. Recall that the ECN bits on an unmarked packet are initially $(10)_2$, and routers set these bits to $(11)_2$ to indicate congestion. If $f \geq u$ or the packet already contains a mark $(11)_2$, then BMCC marks the packet with $(11)_2$, where u is the maximum value of f that ADPM can signal. Otherwise, ADPM computes a deterministic hash h of the packet contents, such as the 16-bit IPid field. This hash is compared to f, and the packet is marked with $(01)_2$ if $f > h$, or left unchanged otherwise. At the receiver, the ECN bits will reflect the state of the most congested router on the path.

The receiver maintains the current estimate, $\hat{f}$ of the load factor at the bottleneck on the forward path. When a packet is received, this estimate is updated as:

$$\hat{f} \leftarrow \begin{cases} u & \text{if } b_{ecn} = (11)_2 \\ h & \text{if } (b_{ecn} = (10)_2 \text{ and } h < \hat{f}) \\ & \text{or } (b_{ecn} = (01)_2 \text{ and } h > \hat{f}) \\ \hat{f} & \text{otherwise} \end{cases}$$

where b_{ecn} refers to the two ECN bits in the IP header of the received packet. The estimate $\hat{f}$ is sent to the sender using TCP options, as described in Section 4.5. Note that the receiver's estimate will lag behind the true value [108], except that values over u are signaled immediately to indicate severe overload.

The resolution depends on the fraction of packets that hash to a particular range. For BMCC, values of f below a threshold η_0 are rounded up to η_0, and the hash is such that 1/4 of packets hash to values in (η_0, η) for some design parameter η, 1/4 of packets hash to $(\eta, 1)$ and 1/2 hash to $(1, u)$.

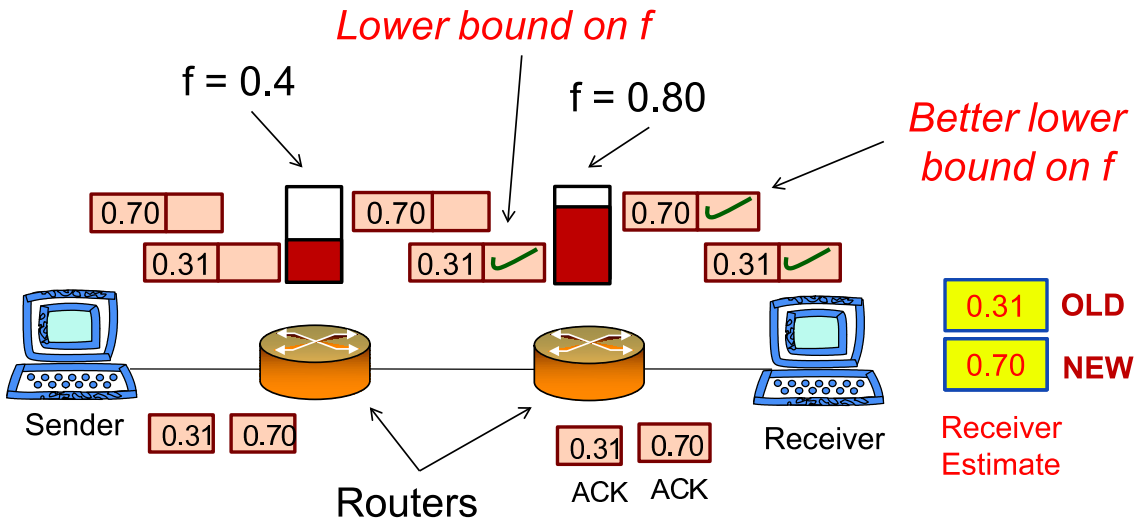


Figure 22: ADPM Illustration

Figure 22 illustrates the working of ADPM. Suppose the path between the sender and the receiver contains two routers with a load of 0.4 and 0.8, respectively. The first packet generated by the sender contains a value in the IPID field that hashes to 0.31. Since this is a lower bound on the load, the first router sets the ECN bits of the packet. When this packet is received, the receiver updates its estimate to 0.31. This value is then communicated to the sender through ACKs. The next packet hashes to 0.7. In this case, the first router leaves the ECN bits unchanged whereas the second router marks the packet because 0.7 is less than 0.8. When this packet is received, the receiver updates its estimate to 0.7 because it is a tighter lower bound on the most congested router. As more packets get received, this estimate becomes more accurate.

4.1.3 BMCC Sender

As argued in [56], the ideal window update strategy given a load factor of f is to increase the total load by a factor of $1 - f$, in a way which balances rates. However, as neither the bottleneck capacity nor number of competing flows are known, BMCC uses different control laws, based on whether the most loaded link is lightly-, heavily- or over-loaded, corresponding to $f \in [0, \eta)$, $[\eta, 1)$ or $[1, \infty)$ for some η. As it will be shown, this also allows both fairness and rapid acquisition of idle capacity.

4.1.3.1 Low Load $(0 \leq \hat{f} < \eta)$ To achieve high utilization rapidly, sources apply MI with factors proportional to $1 - \hat{f}$. In particular,

$$w(t + T) = w(t)(1 + \xi(\hat{f})), \tag{4.2}$$

where T is the RTT of the flow, $\xi(\hat{f}) = \kappa_2(1-\hat{f})/\hat{f}$ and κ_2 is a step size. Since BMCC implements the algorithm VCP sought to, the analysis of [56] shows that stability is achieved for $\kappa_2 < 1/2$; BMCC uses $\kappa_2 = 0.35$.

BMCC aims to give equal rate to flows with different RTTs. Since flows with large RTTs update less often, the rule

$$w(t + T) = w(t)(1 + \xi(\hat{f}))^{T/t_p} \tag{4.3}$$

is used so that windows grow at a rate independent of T.

4.1.3.2 High Load $(\eta \leq \hat{f} < 1)$ When the system has achieved high utilization, the algorithm must seek fairness. This is achieved using AIMD. In high load, sources apply AI:

$$w(t + T) = w(t) + \alpha, \tag{4.4}$$

with $\alpha = (T/t_p)^2$ chosen to cause the equilibrium window to be proportional to the flow's RTT, giving RTT fairness [56].

4.1.3.3 Overload $(1 \leq \hat{f} < \infty)$

When the load factor is greater than 1, the sources use MD:

$$w(t+T) = w(t)\beta(\hat{f}), \tag{4.5}$$

where

$$\beta(\hat{f}) = \beta_{\max} - \frac{\Delta\beta(\hat{f}-1)}{(u-1)} \tag{4.6}$$

varies linearly in $[\beta_{\min}, \beta_{\max}] \subset (0,1)$, u is the maximum value of f that ADPM can signal, and $\Delta\beta = \beta_{\max} - \beta_{\min}$.

4.1.4 Parameter values

4.1.4.1 Measurement interval, t_p : The period t_p should be greater than the RTT of most flows to smooth out sub-RTT burstiness, but should also be small enough to ensure responsiveness to congestion. Internet measurements [14] report that vast majority of flows have RTTs less than 200 ms. Hence, BMCC uses $t_p = 200$ ms.

4.1.4.2 Mode threshold, η : A high value of η avoids under-utilization, but limits AIMD's scope to induce fairness, and risks overshooting the link capacity and causing excess packet loss. To balance these, BMCC uses $\eta = 0.75$, and $\eta_0 = 0.15$. As an exception, new flows remain in MI until $\hat{f}$ first reaches 1.

4.1.4.3 Backoff parameter, β : The MD parameter varies from $\beta_{\min} = 0.65$ when $f = u = 1.2$ to $\beta_{\max} = 0.875$ when $f = 1$, for the following reasons. A high value of $\beta_{\max}$ (such as 0.99) leads to slow convergence and reduces responsiveness to congestion. In contrast, a low value of $\beta_{\max}$ (such as 0.5) reduces the average throughput ($\sim C(1+\beta)/2$) and introduces large variations in the throughput of flows, which is highly undesirable for real-time and multimedia

applications [23]. To balance these, BMCC uses $\beta_{\max} = 0.875$ (see Section 4.3). However, to ensure high responsiveness and fast convergence under high load, $\beta(f)$ is decreased linearly with f until $\beta(f) = \beta_{\min}$. The choice of $\beta_{\min}$ determines the range of values that $\beta(f)$ can assume. This range should be large enough to reap the benefits of small β values but small enough to prevent flows from entering MI after overload detection, which can lead to high packet loss rate. To select the value to ensure this, note that $\min(f\beta(f)) \geq \eta$. Since for $f \geq 1.2$, the lowest β is applied, $\beta(f)$ should be at least $\eta/1.2 = 0.625$. BMCC uses $\beta_{\min} = 0.65$.

4.2 DESIGN ISSUES

The foregoing design raises some questions, which are now addressed.

4.2.1 What is the congestion level assumed by new flows?

ADPM needs an initial estimate of the congestion level. New flows initially estimate $\hat{f} = \eta_0$, and thus increase their windows by a factor of $\xi(\eta_0) \approx 3$ per t_p. This helps short flows to finish quickly, and does not induce excessive burstiness, since it is only slightly faster than existing slow start.

4.2.2 Can new flows cause overload before ADPM has been able to signal congestion?

When $f \in [1, u)$, sources enter MD *probabilistically* using ADPM. In the presence of a large number of flows, congestion will be avoided if most reduce their windows, even if some miss the congestion signal. However, if $f > u = 1.2$, each flow decreases its window deterministically (using the standard ECN "Congestion Experienced" codepoint) which prevents persistent congestion.

4.2.3 Sources may apply different β values at the same time; does this lead to unfairness?

Sources with ADPM use β that varies in $\left[\beta(f), \beta(\hat{f})\right]$ depending on the estimated load factor. This may lead to short-term unfairness (on the scale of a few RTTs) but causes no long-term harm.

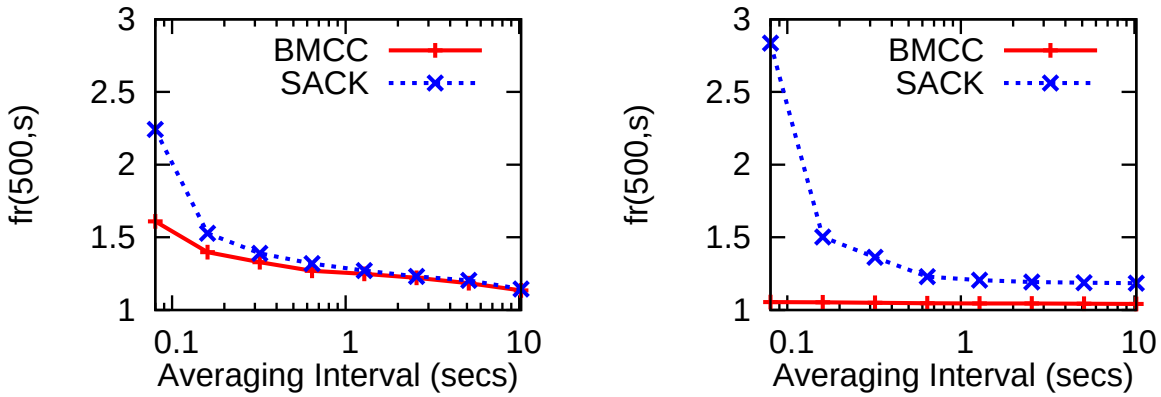


Figure 23: Fairness rate as a function of the averaging interval (T=80 ms) on a 1 Mbps and a 45 Mbps link.

To quantify this, we compare the level of unfairness caused by TCP SACK and BMCC for a range of time scales.

Consider an averaging interval s. For two SACK flows with unsynchronized losses, let $X_i(t,s)$ be the average rate of each flow i over the interval $(t, t+s)$, and let the "fairness rate" be

$$fr(\tau, s) = \frac{1}{\tau}\int_0^{\tau} \frac{\max_i(X_i(t,s))}{\min_j(X_j(t,s))}\,dt,$$

where $i, j \in \{1,2\}$ and τ is the total observation period. Figure 23 shows $fr(500,s)$ against the averaging interval s for two link capacities and $T = 80$ ms. Observe that the fr curve for BMCC remains below that of SACK implying that it is always fairer than SACK on short time scales. The value is higher on the 1 Mbps link; this is because the average β value is higher in this case and hence the variation also is.

4.2.4 Why use a higher MI threshold when flows start?

A long-lived BMCC flow uses AIMD in steady-state. In the presence of such a flow, a newly arriving flow would normally apply AIMD too, which causes slow convergence. In order to improve

the AFCT of short flows in this scenario, a larger initial η is used. This allows new BMCC flows to apply the more aggressive, load-factor guided MI until the end of the first congestion epoch.

4.3 MODELS FOR CHARACTERIZING THE PERFORMANCE OF BMCC

The properties of BMCC will now be studied by considering the rate achieved by a newly starting flow in two important cases: a link already carrying N long-lived flows, and an idle link.

4.3.1 Convergence to Fairness on a Loaded Link

This section considers the rate of convergence to fairness when a new flow starts competing with N long-lived BMCC flows. This scenario reflects the fact that most data in the Internet comes from "elephant" flows, and so a highly multiplexed bottleneck is likely to have long-lived flows [97].

Consider a bottleneck link of capacity C_l shared by N flows with equal RTTs $T = t_p$. Define a round as a single AIMD cycle, the duration of which is d_e RTTs. Note that BMCC causes these rounds to be synchronized between flows, because congestion is signaled to all flows, rather than random packet drops for a small number of flows. Let $\Delta w_{ij}(e) = w_i(e) - w_j(e)$ for flows i and j in the eth round. Since $\beta_{\min}$ is chosen so that flows do not re-enter MI mode after an MD, this difference is affected only by MD events. Thus

$$\begin{aligned}\mathbb{E}[\log(\Delta w_{ij}(e))] &= \mathbb{E}[\log(\beta(f))] + \mathbb{E}[\log(\Delta w_{ij}(e-1))] \\ &= e\mathbb{E}[\log(\beta(f))] + \log(\Delta w_{ij}(0)).\end{aligned} \tag{4.7}$$

For the new flow i to reach parity with a flow j with $w_j(0) = B \equiv C_l T/N$, in the sense of $\Delta w_{ij}(e)/B \leq \delta$, it would take

$$m = \frac{\log(1/\delta)}{\mathbb{E}[\log(1/\beta(f))]} \tag{4.8}$$

AIMD rounds, which is constant with respect to BDP [93].

The duration of an AIMD round can be calculated by noting that, when the link utilization exceeds 100%, the aggregate congestion window size, $w_a = \sum_{i=1}^{N} w_i(t)$ is at least $k = C_l T$. Therefore, it would take at least $d_e = k(1 - \mathbb{E}[\beta(f)])/N\alpha$ RTTs for w_a to become greater than k after applying MD.

Note that m is negatively correlated with the durations of the m epochs, and so the actual mean convergence time is slightly less than $\mathbb{E}[m]\mathbb{E}[d_e]$. Thus, the total number of RTTs needed to converge to a fair bandwidth allocation is bounded above, and approximated, by

$$r = B\frac{\log(1/\delta)(1 - \mathbb{E}[\beta(f)])}{\alpha\mathbb{E}[\log(1/\beta(f))]}. \tag{4.9}$$

Thus, the convergence time until absolute fairness varies with BDP as $O(B)$.

In order to derive the expected value of $\beta(f)$, we will use the following lemma, established in Section 4.4.

Lemma 4.3.1. *In steady-state, flows with homogenous round-trip times detect overload with high probability ($\geq$0.8) in one t_p with ADPM.*

To determine the expected value of $\beta(f)$ in overload, we first derive the maximum value of f when it exceeds 100%.

In steady-state, each BMCC flow achieves the same rate independent of its RTT. Since f is measured over t_p, this implies that in overload $\lambda \geq t_p N(C_l/N)$. Further, with AI, the aggregate rate increases by N pkts per t_p. Thus, when overload is first detected, up to t_p after onset, λ is uniformly distributed on $(C_l t_p, C_l t_p + N)$, and the average queue length, q_{av}, will be uniformly distributed in $(0, N)$. From (4.1), the first congested load factor is thus distributed as

$$f \sim \frac{C_l t_p + N(1+\kappa_1)U(0,1)}{\gamma_l C_l t_p}$$
$$\sim 1 + \frac{2N}{C_l t_p} U(0,1), \tag{4.10}$$

where $U(0,1)$ is a uniformly distributed random variable on $(0,1)$, $\gamma_l = 1$ and $\kappa_1 \leq 1$. Since $\beta(f)$ is linear by (4.6), $\mathbb{E}[\beta(f)] = \beta(\mathbb{E}[f])$, which by (4.10) gives

$$\mathbb{E}[\beta(f)] = \begin{cases} \beta_{\max} - \frac{N\Delta\beta}{k(u-1)} & f \leq u \\ \beta_{\min} & f > u \end{cases}$$

This establishes the mean values

- Duration of an epoch: $\mathbb{E}[d_e] = \frac{k}{N\alpha}(1-\beta_{\max}) + \frac{\Delta\beta}{\alpha(u-1)}$
- Number of epochs: $\mathbb{E}[m] = \dfrac{\log(1/\delta)}{\mathbb{E}[\log(1/\beta(f))]}$

Figures 24 and 25 show $\mathbb{E}[m]$ and $\mathbb{E}[d_e]$ respectively, as β is varied in $[0.5, 1)$ (with $\beta = \beta_{\min} = \beta_{\max}$), and as a function of $N \in [2, 120]$ for different BDPs, $k \in \{1000, 5000, 10000\}$ pkts. Observe that while $\mathbb{E}[d_e]$ decreases linearly with β, $\mathbb{E}[m]$ increases exponentially with it, leading to slower convergence as shown in Figure 26. Further, if $\beta(f)$ is made a linear function of f then $\mathbb{E}[m]$, $\mathbb{E}[d_e]$ and r decrease with N. The reason is that as N increases, the average load factor value at the bottleneck increases causing the sources to apply a smaller $\beta(f)$ value. This results in improved convergence. Also note that convergence becomes slower as the BDP of the path increases.

The above analysis is related to the model presented in [93]. However, they consider the back-off parameter, $\beta(f)$, to be a binary random variable, whereas in our case it is continuous. The analysis can be extended to flows with heterogeneous RTTs but at the price of a more involved development (see [93]). The qualitative insights relating to the influence of the AIMD parameters, however, remain unchanged.

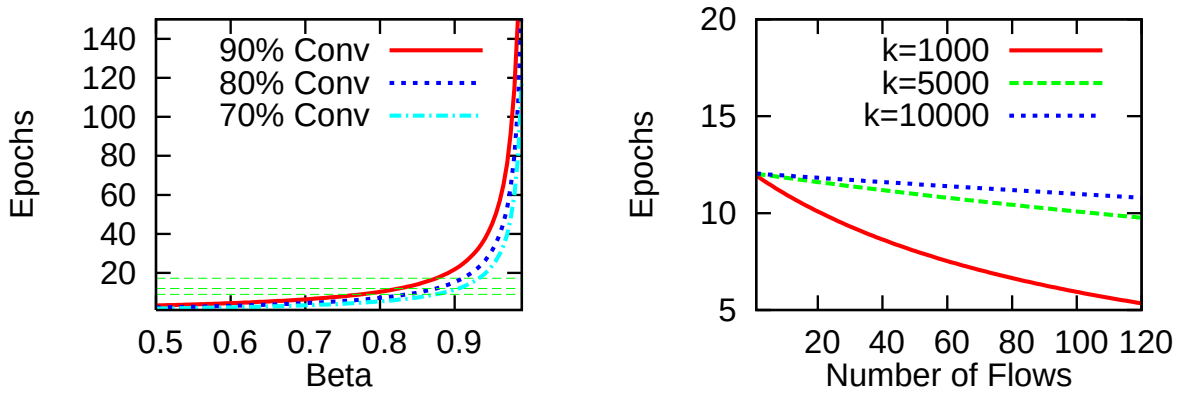


Figure 24: Number of epochs needed for 70%, 80% and 90% convergence as a function of β (The green lines show the epochs for $\beta_{\max} = 0.875$) and 80% convergence as a function of the number of flows for different BDPs, $k \in \{1000, 5000, 10000\}$ pkts (where $\beta_{\max} = 0.875$ and $\beta_{\min} = 0.65$)

4.3.2 Flow starting on an idle link

A well known problem of TCP SACK is that a flow sending on an idle path with large BDP takes too long to start up [8], and then causes many packet losses when the window finally reaches the BDP. BMCC addresses this issue by increasing its rate faster until incipient congestion is explicitly signaled, and then slowing down the rate of increase, in three phases. These will now be examined for the fluid limit of large BDPs with packet pacing and without delayed acknowledgements, and in the simple case that $T = t_p$. Other cases are similar, except that small BDPs require the behavior of ADPM to be modelled.

The first phase, with $f = w/(C_l T) < \eta_0$, increases the window by a factor of $\kappa_2(1-\eta_0)/\eta_0 \approx 3$ each RTT, giving

$$w(t) = 3^{t/T} \qquad t \le t_1 \equiv T \log_3(\eta_0 C_l T) \tag{4.11}$$

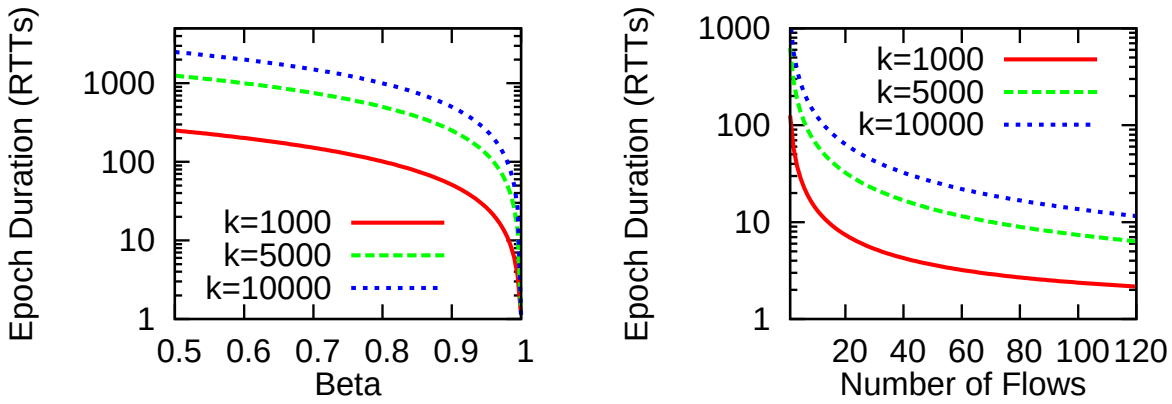


Figure 25: Duration of an epoch as a function of β, with $N = 2$ and the number of flows for different BDPs, k (where $\beta_{\max} = 0.875$ and $\beta_{\min} = 0.65$)

If the BDP is sufficiently large, $\eta_0 C_l T$ is large enough for ADPM to estimate the load factor accurately, and to enter the second phase. If the BDP is small, this aggressive phase will last longer giving BMCC a faster start-up. The second phase, with $f = w/(C_l T) \in (\eta_0, \eta)$, increases the window by $w \cdot \kappa_2(1-f)/f = w \cdot \kappa_2(C_l T - w)/w$ each RTT, giving

$$\frac{dw}{dt} = \frac{\kappa_2}{T}(C_l T - w).$$

Solving this, and applying the initial condition

$$w(t) = C_l T \left(1 - (1-\eta_0)e^{-\kappa_2(t-t_1)/T}\right) \qquad t_1 < t \le t_2 \qquad (4.12)$$

where t_2 is defined by

$$t_2 - t_1 = \frac{T}{\kappa_2} \log\left(\frac{1-\eta_0}{1-\eta}\right).$$

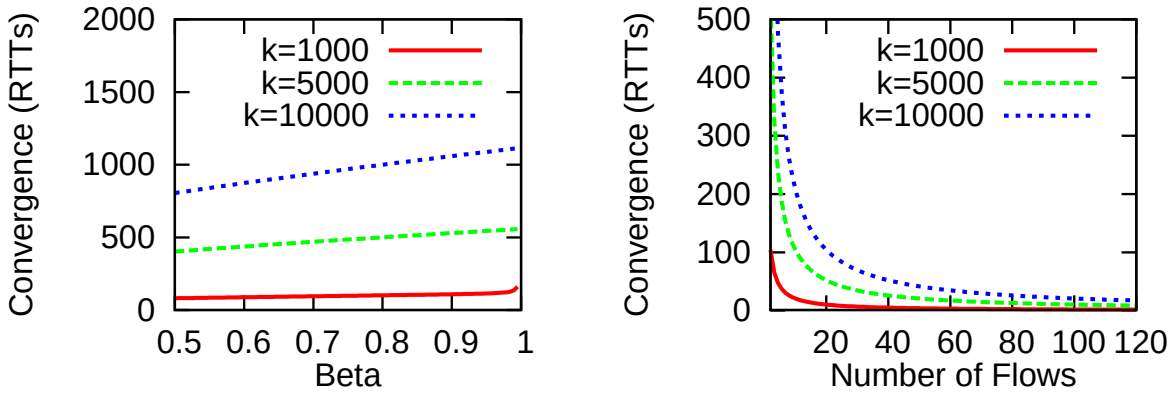


Figure 26: Convergence time as a function of β with $N = 2$ and number of flows for different BDPs, k (where $\beta_{\max} = 0.875$ and $\beta_{\min} = 0.65$)

Note that, unlike regular multiplicative increase, this results in concave *negative* exponential growth. The derivative is continuous at the point of inflection, t_1.

In the third phase, w grows at $\alpha = 1$ packet per RTT, giving $w(t) = \eta C_l T + \alpha(t - t_2)/T$. This decreases the rate of increase of the window unless $\alpha > \kappa_2 C_l T(1-\eta)$, corresponding to a window of $\alpha/(\kappa_2(1-\eta)) \approx 12$ packets.

The total amount of data transmitted until time t is $d = \int_0^t w(\tau)/T \, d\tau$, given by

$$d(t) = \begin{cases} 3^{t/T}/\log(3) & t \le t_1 \\ d(t_1) + C_l(t - t_1) + H(t)T/\kappa_2 & t_1 < t \le t_2 \\ d(t_2) + \eta C_l(t - t_2) + A(t) & t > t_2 \end{cases}$$

where $H(t) = (e^{-\kappa_2(t-t_1)/T} - 1)(1 - \eta_0)$ and $A(t) = \alpha t(t - 2t_2)/(2T)$.

In contrast, SACK has a window of $w_R(t) = 2^{t/T}$ for all $t < T\log_2(C_l T)$, and sends data $d_R(t) = 2^{t/T}/\log(2)$. Similarly, VCP sets $G = 1.0625$ and has

$$w_V(t) = \begin{cases} G^{t/T} & t \le t_V \\ G^{t_V/T} + \alpha(t - t_V)/T & t > t_V \end{cases}$$

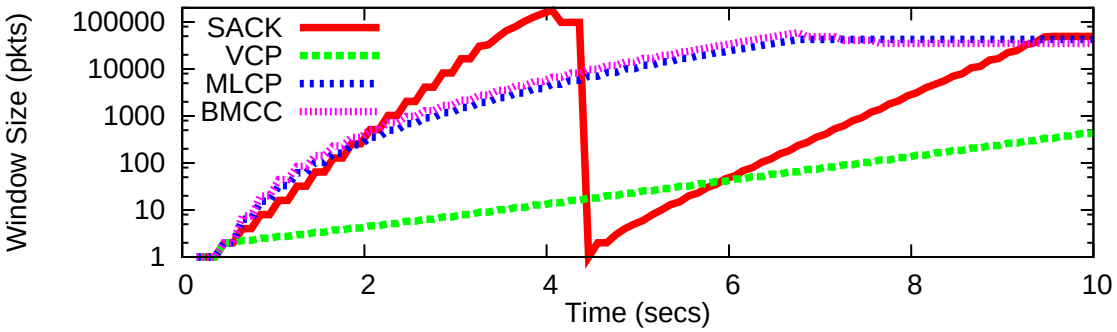


Figure 27: Comparison of the growth rate of the congestion window sizes for SACK, VCP, MLCP and BMCC on a 2 Gbps link with $T = 200$ ms

and

$$d_V(t) = \begin{cases} G^{t/T}/\log(G) & t \leq t_V \\ d_V(t_V) + 0.8C_l(t - t_V) + A_V(t) & t > t_V \end{cases}$$

where $t_V = T \log_G(0.8C_lT)$ and $A_V(t) = \alpha t(t - 2t_V)/(2T)$.

Figure 27 compares the growth of the congestion window of a single SACK, VCP, MLCP and BMCC flow starting on an idle link. The bottleneck is a 2 Gbps link with RTT=200 ms yielding a BDP of 50000 pkts. The bottleneck buffer size was set to the BDP of the path as specified by the BDP rule. Observe that in the first 2 s, BMCC attains a larger window size than SACK. As the available bandwidth decreases, BMCC adjusts its growth rate and therefore, has a steadier increase. SACK, on the other hand, continues to grow its window size by a factor of two every round-trip time (a straight line on the exponential plot in Figure 27). When it completely fills the router buffers, it has a window size of ∼100000 pkts. Since SACK does not know that it has saturated the path, it continues to increase its window, which results in a loss of ∼90000 pkts giving rise to timeouts. BMCC does not experience a single packet loss. MLCP achieves near-optimal rate of

convergence to efficiency for load factor based schemes [57] but uses 4 bits for feedback. Observe that BMCC very closely approximates the performance of MLCP using only the existing bits. In contrast, VCP is too conservative. In the first 10 s, it is able to attain a congestion window size of only 475 packets (∼1% of the BDP of the path) while a BMCC flow attains a window size of 50000 pkts in less than 6.5 s.

4.4 QUANTIFYING THE IMPACT OF ADPM

In the previous section, we used Lemma 4.3.1 to assume that sources detect overload within t_p of when f exceeds 100% using ADPM. In this section, we justify that assumption.

Because f increases by $F = 2N/C_l t_p$ per t_p, and is sampled once per t_p, we can model the first congested load factor as

$$f \sim U(1, 1 + F).$$

Given f, the probability that a packet of flow i detects overload using ADPM is equal to the probability that the packet's hash value is $h \in [1, f]$. Since half of the hash values are uniformly distributed in $[1, u]$, this is $p_f = 0.5(f - 1)/(u - 1)$. The probability that overload remains undetected after d packets is then $(1 - p_f)^d$. Thus, the overall probability of overload being undetected after d packets is

$$P(R > d) = \mathbb{E}_f[(1 - p_f)^d],$$

where R is the number of packets from flow i until overload is detected. Let $D = 0.5F/(u - 1)$. Then $p_f \sim U(0, D)$, and

$$\begin{aligned} P(R > d) &= \frac{1}{D} \int_0^D (1 - x)^d \, dx \\ &= \frac{1 - (1 - D)^{d+1}}{D(d + 1)}. \end{aligned} \tag{4.13}$$

Using the above expressions, we now prove Lemma 1.

Proof of Lemma 1. Consider a single bottleneck link with N long-lived BMCC flows in steady-state, each with $T = t_p$. In the first t_p interval after overload, each flow sends $d = (C_l t_p/N + 1) = (k/N + 1)$ packets. Substituting the values of d and D in (4.13), we get

$$P(R > d) = \frac{1 - (1 - N/(k(u-1)))^{k/N+2}}{(1 + 2N/k)/(u-1))}$$
$$\leq \frac{1}{1/(u-1)} = 0.2.$$

□

4.4.1 Experimental Validation

In order to validate the above model, we run extensive ns2 simulations and compare the results against the predicted values. In the first set of experiments, we vary the average per-flow BDP of the path from 25 pkts to 1000 pkts. We maintain ten long-lived BMCC flows with heterogenous round-trip times that vary in [25 ms, 295 ms] (each with a fixed difference of 30 ms). We call the values computed using the above expressions "Model" and those through ns2 simulations "Experimental".

Figures 28 and 29 show values of $P(R > d)$ and $\mathbb{E}[R]$ as a function of the average per-flow BDP of the path. Observe that $P(R > d)$ remains below 0.2 across a range of per-flow BDPs and $\mathbb{E}[R]$ increases almost linearly with the per-flow BDP as predicted by the model. Simulations results yield a smaller value for $P(R > d)$ because F assumes a higher average value than predicted by the model. This may be because flows with $T > t_p$ tend to be bursty, leading to a larger queue buildup and thus a higher value for F. Note that for small per-flow BDPs (<50 pkts), the value of $P(R > d)$ is close to 0.2. The reason is that for such small BDPs, f becomes greater than 1.2 many times in which case ADPM is not used for overload detection. In these experiments we only consider overload detected through ADPM. If overload is allowed to recover through ADPM for f>1.2, $P(R > d)$ would become much lower than 0.2.

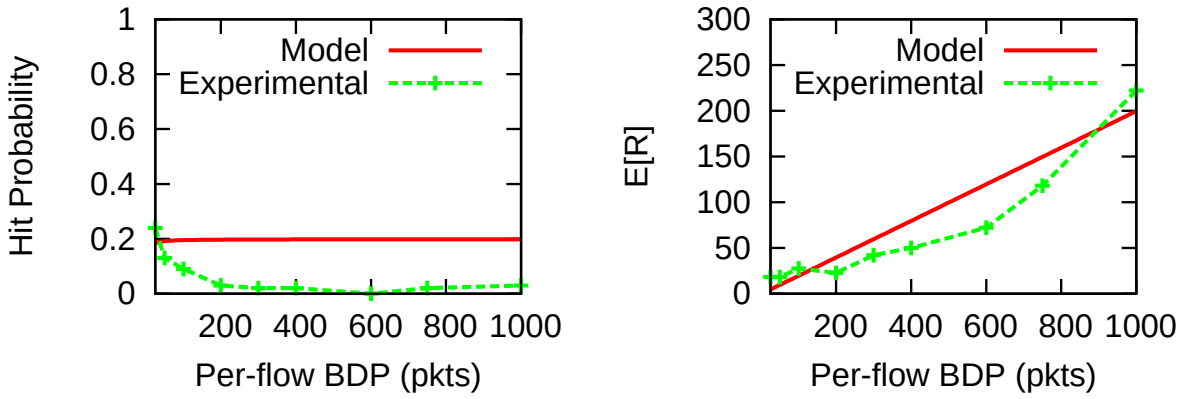


Figure 28: Probability of overload detection in one t_p with ADPM and the average number of packets needed to detect overload as a function of the average per-flow BDP of the path. $N = 10$ and $T \in [25\,\text{ms}, 295\,\text{ms}]$.

In the next set of experiments, we vary the number of long-lived flows while keeping the BDP of the path fixed at 1000 pkts. The generated flows have round-trip times that vary in $[25\,\text{ms}, (N-1)30+25\,\text{ms}]$. Figures 28 and 29 show that $P(R > d)$ remains below 0.2 for a wide range of per-flow BDPs, following closely the trend predicted by the model.

4.5 REDUCING THE OVERHEAD OF USING TCP OPTIONS

The BMCC receiver communicates the estimated load factor, $\hat{f}$, to the sender using TCP options. Unlike on the forward path, TCP options are acceptable for this because they need not be processed by the routers. However, they are a significant overhead. Rather than simply piggybacking $\hat{f}$ on every ACK, it is only necessary to send $\hat{f}$ if it changes. This approach (which we call "non-redundant") increases the sensitivity to lost ACKs.

The alternative used by BMCC is to send the estimate immediately after it changes, and then to send redundant copies with decreasing frequency. Each receiver maintains a counter i, and sends $\hat{f}$ every ith ACK. The counter is reset to 1 each time the load factor changes and incremented by 1

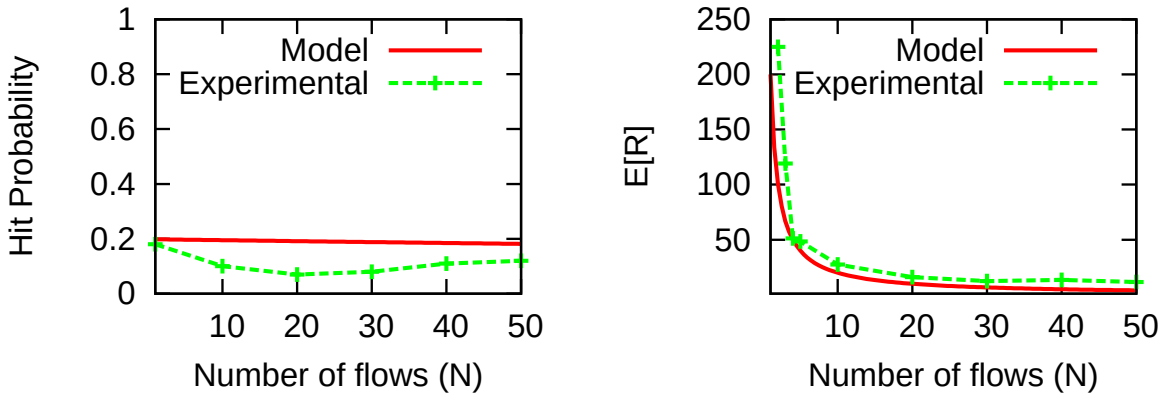


Figure 29: Probability of overload detection in one t_p with ADPM and the average number of packets needed to detect overload as a function of the number of flows ($k = 1000$ pkts). $T \in [25\,\text{ms}, (N-1)30\,\text{ms}]$.

each time $\hat{f}$ is sent. This scheme is robust against losing a small number of ACKs, but if $\hat{f}$ changes on average once per n packets, the overhead is only $\log(n)/n$ times that of naïvely echoing on every ACK.

We conducted ns2 simulations to evaluate the reduction in overhead for a dumbbell topology with a bottleneck link of capacity 100 Mbps, carrying ten long-lived flows in each direction with T=80 ms. Table 5 shows statistics that correspond to the average of the ten flows in the forward path. Of 552942 ACKs generated by the receivers, the load factor estimate changed for 6914 (1.3%), which carried $\hat{f}$ under both schemes, whereas 12.9% carried $\hat{f}$ under BMCC's robust scheme.

4.6 PERFORMANCE EVALUATION

In this section, we evaluate and compare the performance of BMCC with other protocols using the packet-level simulator ns2 [96], which we have extended with a BMCC module. A diverse set of network scenarios are considered including varying link capacities in the range

	ACKs sent	ACKs with $\hat{f}$	Reduction(%)
non-redundant	552942	6914	98.7
BMCC	552942	71476	87.1

Table 5: Overhead of signalling from receiver to sender.

[100 kbps, 2 Gbps], round-trip times in the range [1 ms, 2 s], number of long-lived, FTP-like flows in the range [1, 1000], and short-lived, web-like flows with offered load and average transfer sizes in the range [0.1C_l Mbps, C_l Mbps] and {30KB, 300KB, 3000KB}, respectively. All simulations use a dumbbell topology with a single bottleneck link. The basic setting is a 155 Mbps link (equal to the capacity of an OC3 link) with 80 ms RTT where the forward and reverse path each has 5 FTP flows unless stated otherwise[1]. TCP SACK is always used with RED and ECN enabled at the routers. The bottleneck buffer size is set to the BDP, or two packets per-flow, whichever is larger. The data packet size is 1000 bytes, while the ACK packet size is 40 bytes. All simulations are run for at least 100 s. The statistics neglect the first 5% of the simulation time.

4.6.1 Varying Bottleneck Capacity

We vary the bottleneck capacity from 100 kbps to 2 Gbps while keeping everything else fixed. Figure 30 shows that BMCC is able to maintain high utilization ($\geq$90%) while maintaining low persistent queue length ($<$20% BDP) and negligible packet loss rate across a range of link capacities. While average utilization for SACK reduces to 60$\sim$70% for link capacities $>$10 Mbps, VCP, MLCP, XCP and RCP all achieve $\geq$85% utilization across a range of link capacities. RCP and SACK result in high loss rates for small capacities whereas all other schemes experience negligible loss rates. Average queue length with XCP (5$\sim$40% BDP) is much higher than other schemes for link capacities higher than 1 Mbps.

[1] 80 ms is the RTT between the East and the West coasts. We use 5 flows so as to evaluate the performance under per-flow bandwidth regimes

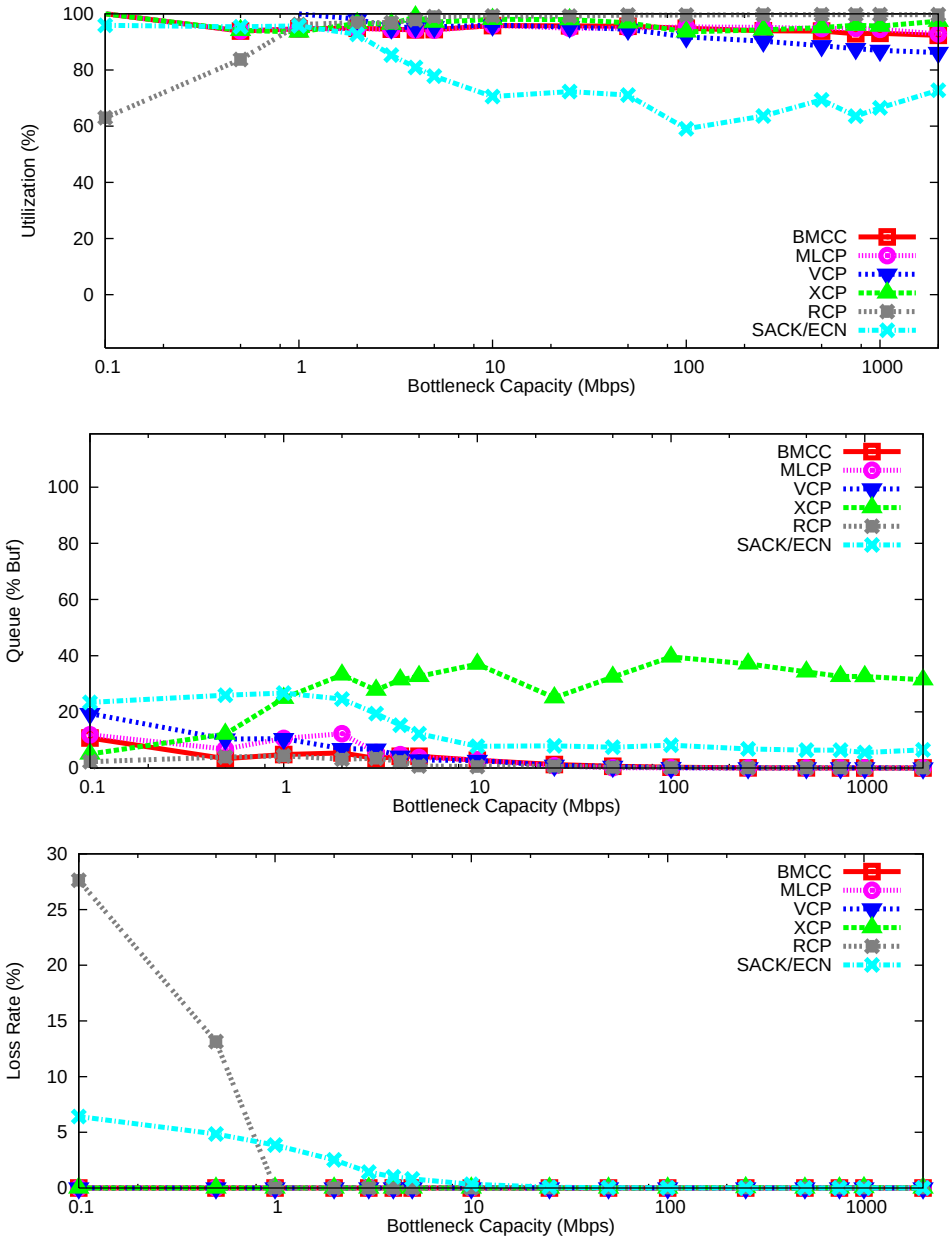


Figure 30: Impact of varying the bottleneck capacity from 100 kbps to 2 Gbps.

4.6.2 Varying Feedback Delay

We now vary the round-trip time from 1 ms to 2 s while keeping everything else fixed. Figure 31 shows that BMCC, MLCP and VCP are able to maintain high utilization ($\geq$80%) while maintain-

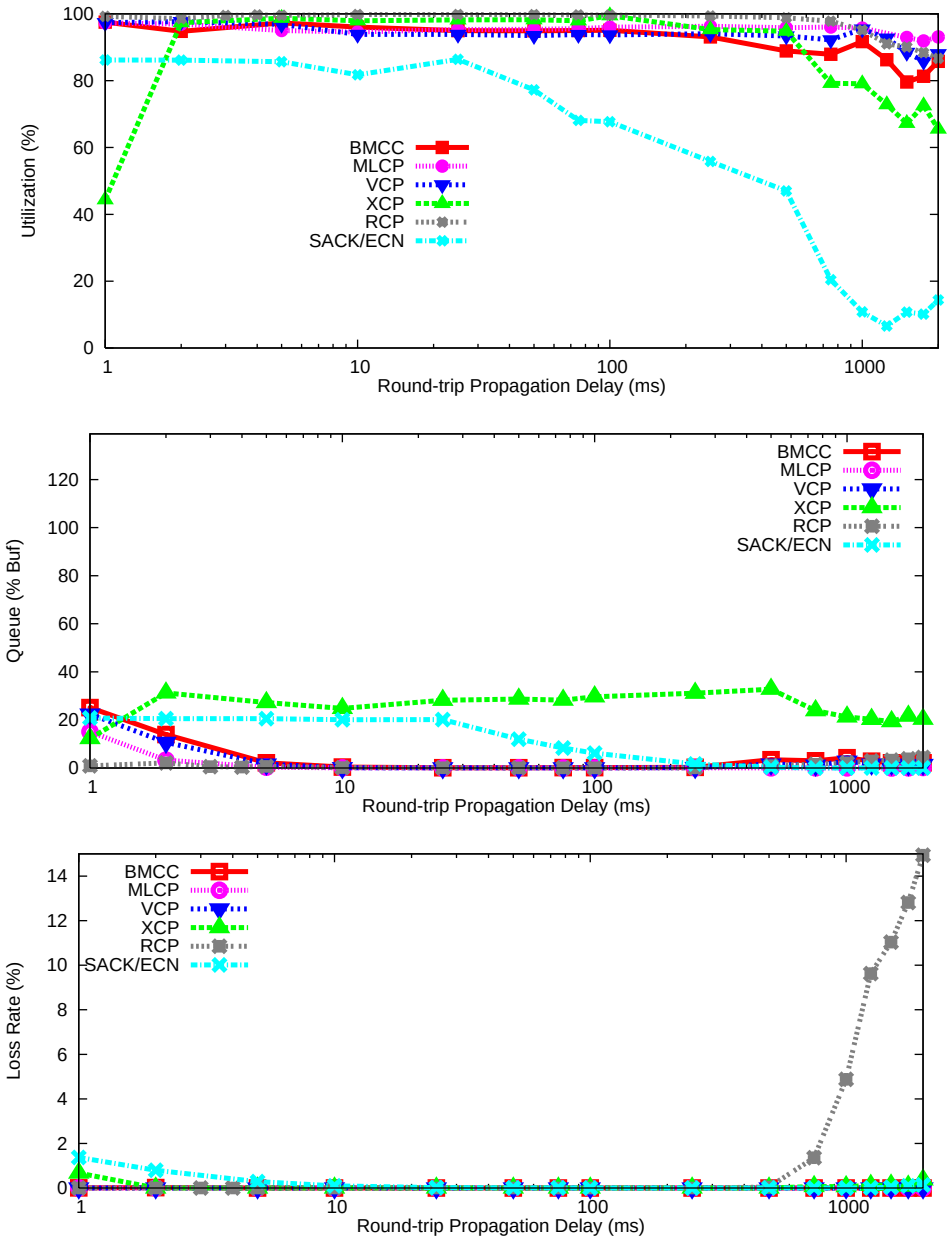


Figure 31: Impact of varying the round-trip propagation delay from 1 ms to 2 s.

ing low persistent queue length ($\leq$25% BDP) and negligible packet loss rate across a range of RTTs. Observe that with SACK, average utilization reduces to $<$20% for large RTTs. XCP results in a higher average queue length (20$\sim$35% BDP) than other schemes. For low RTTs (e.g., $<$2 ms) the average queue length for BMCC, MLCP and VCP rises to about 5$\sim$25% BDP. This happens

because the AI parameter value used in these schemes is large for such small BDP paths. Note that for large RTTs, RCP results in a loss rate of up to 15%; a consequence of its aggressive rate allocation scheme.

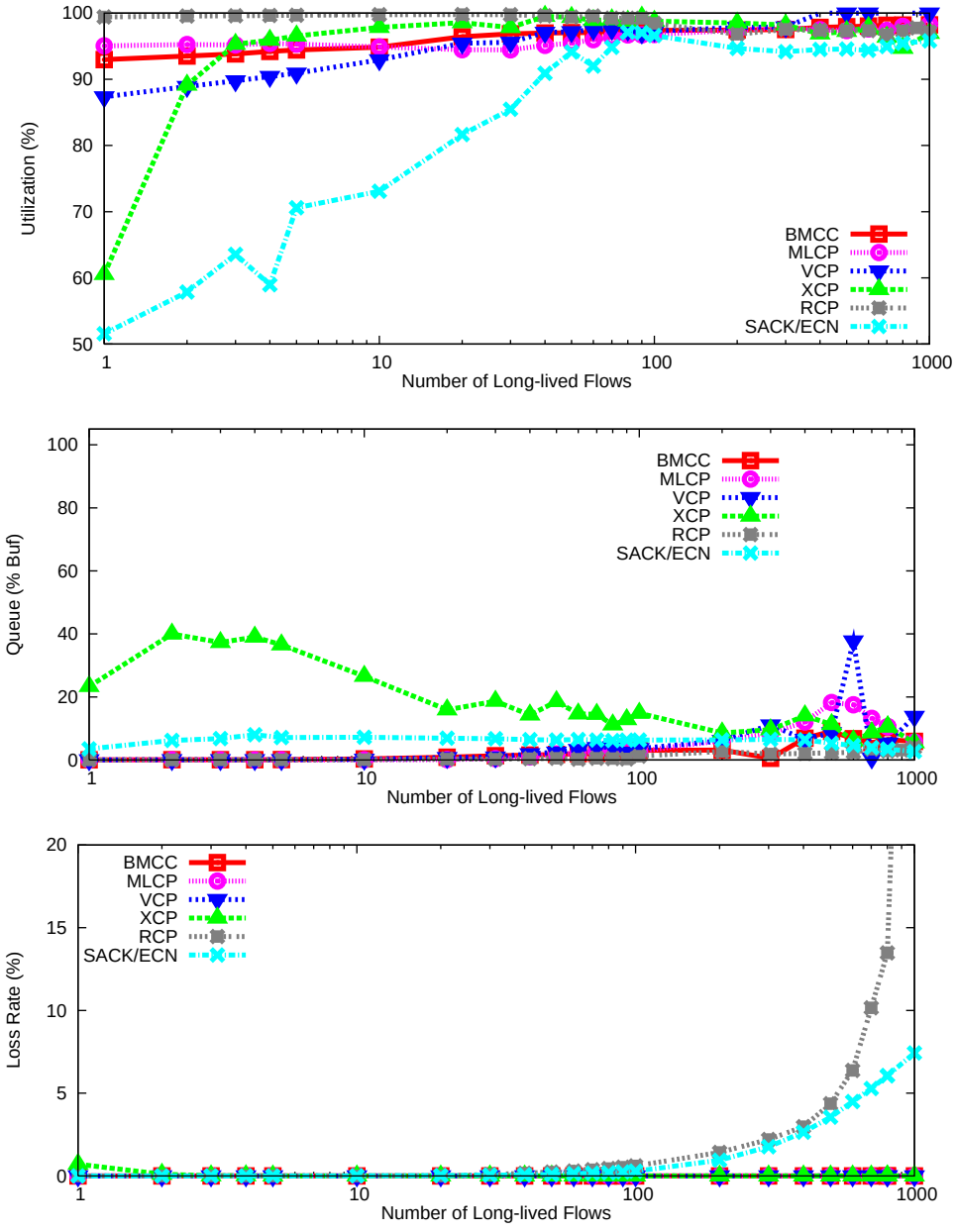


Figure 32: Impact of varying the number of long-lived, FTP-like flows from 1 to 1000.

4.6.3 Varying Number of Long-lived Flows

The number of long-lived flows is now varied from 1 to 1000 while keeping everything else fixed. Figure 32 shows that BMCC, MLCP, VCP, XCP and RCP are able to maintain high utilization ($\geq$90%). While average queue length for BMCC, MLCP and RCP remains less than $<$10% BDP, it is higher for XCP ($\sim$10%$-$40%), VCP (in some cases $\sim$20%$-$40%) and SACK ($\sim$10%). A higher average queue length for VCP in some cases is due to the usage of a higher MD factor than MLCP and BMCC. While loss rate for SACK rises to $\sim$8% for 1000 flows, it becomes more than 20% with RCP.

4.6.4 Pareto-Distributed Traffic

To study the performance of BMCC in the presence of variability and burstiness in flow arrivals, we generate web-like flows whose transfer sizes obey the Pareto distribution (shape=1.4) and arrive according to a Poisson process [8].

i) Varying Average File Size: We vary the average file size from 30 KB to 3000 KB for bottleneck capacities of 10 Mbps and 100 Mbps and measure the AFCT of flows[2]. Figure 33 shows the AFCT (normalized by the smallest AFCT) as a function of the average file size. Observe that on a 10 Mbps link, BMCC outperforms all schemes across a range of file sizes. VCP and SACK stretch the AFCT of flows by a factor of up to $\sim$3.5 and $\sim$2 over BMCC, respectively. On a 100 Mbps link, RCP performs best when the average file size is 30 KB and 300 KB. XCP, however, outperforms other schemes when the average file size is 3 MB. BMCC is the second best performing scheme in all cases. For an average transfer size of 30 KB, VCP, XCP and MLCP stretch the AFCT of flows by factors of up to $\sim$2.7, $\sim$2.4 and $\sim$2.3.

VCP results in the highest AFCT because it uses a conservative MI factor of 1.0625, a consequence of quantizing the load factor information into only three levels. BMCC, in contrast, obtains load factor estimates of up to 16-bit resolution, allowing larger MI factors in low-load. BMCC also

[2]The average file size on several Internet links has been reported to be in the order of few 10s of KBs [109]. However, with the increased popularity of websites such as YouTube, the average transfer size is likely to increase. Therefore, we evaluate the performance of BMCC across transfer sizes that differ by a factor 10, specifically, {30KB, 300KB, 3000KB}.

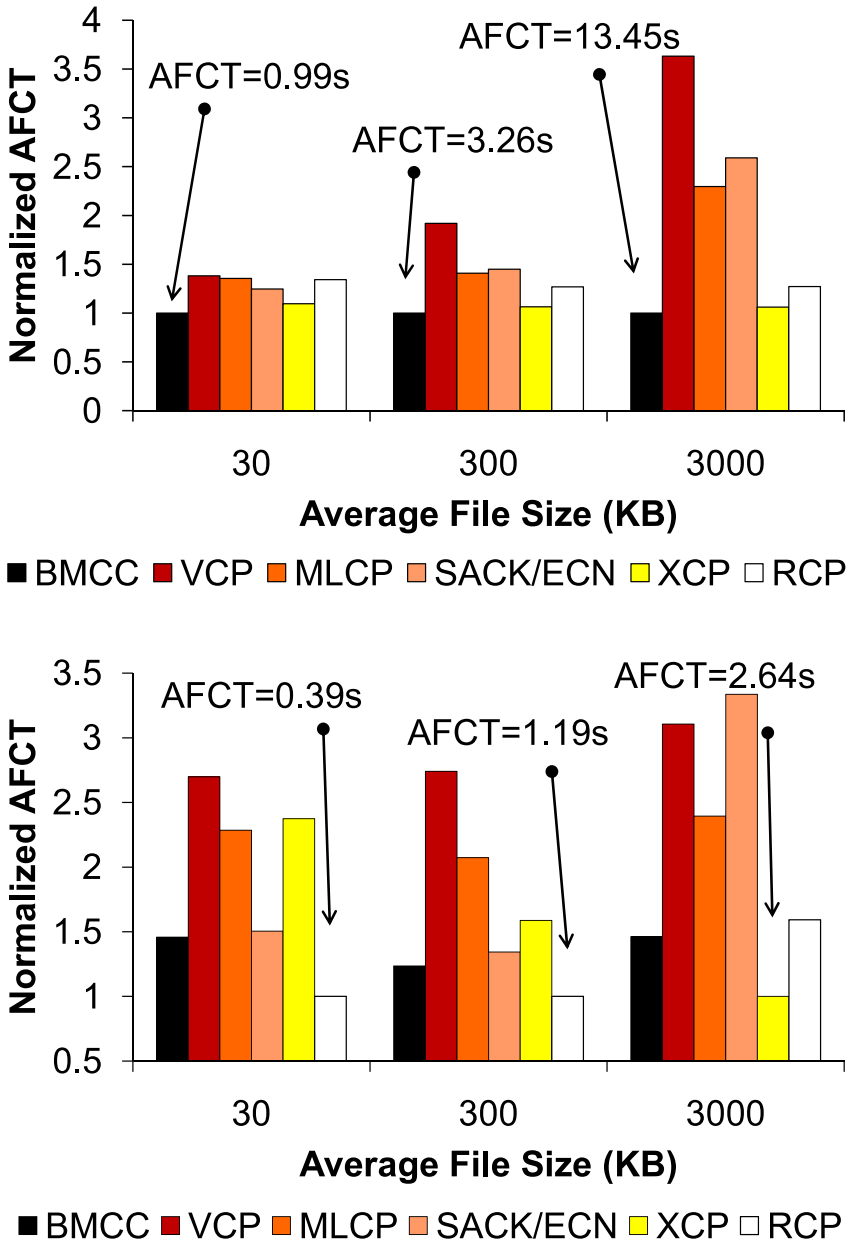


Figure 33: Normalized AFCT as a function of the average file size for bottleneck capacities of 10 Mbps and 100 Mbps. The arrows indicate the scheme with the best AFCT.

allows new flows to use MI for longer than VCP or MLCP, which reduces the AFCT. With SACK's slow-start algorithm, flows use a fixed MI factor of two. In high load, this is too aggressive, induc-

ing a high loss rates which increases the AFCT. XCP increases the AFCT flows because new flows apply AIMD. RCP gives higher rates to new flows which helps short flows to finish quickly.

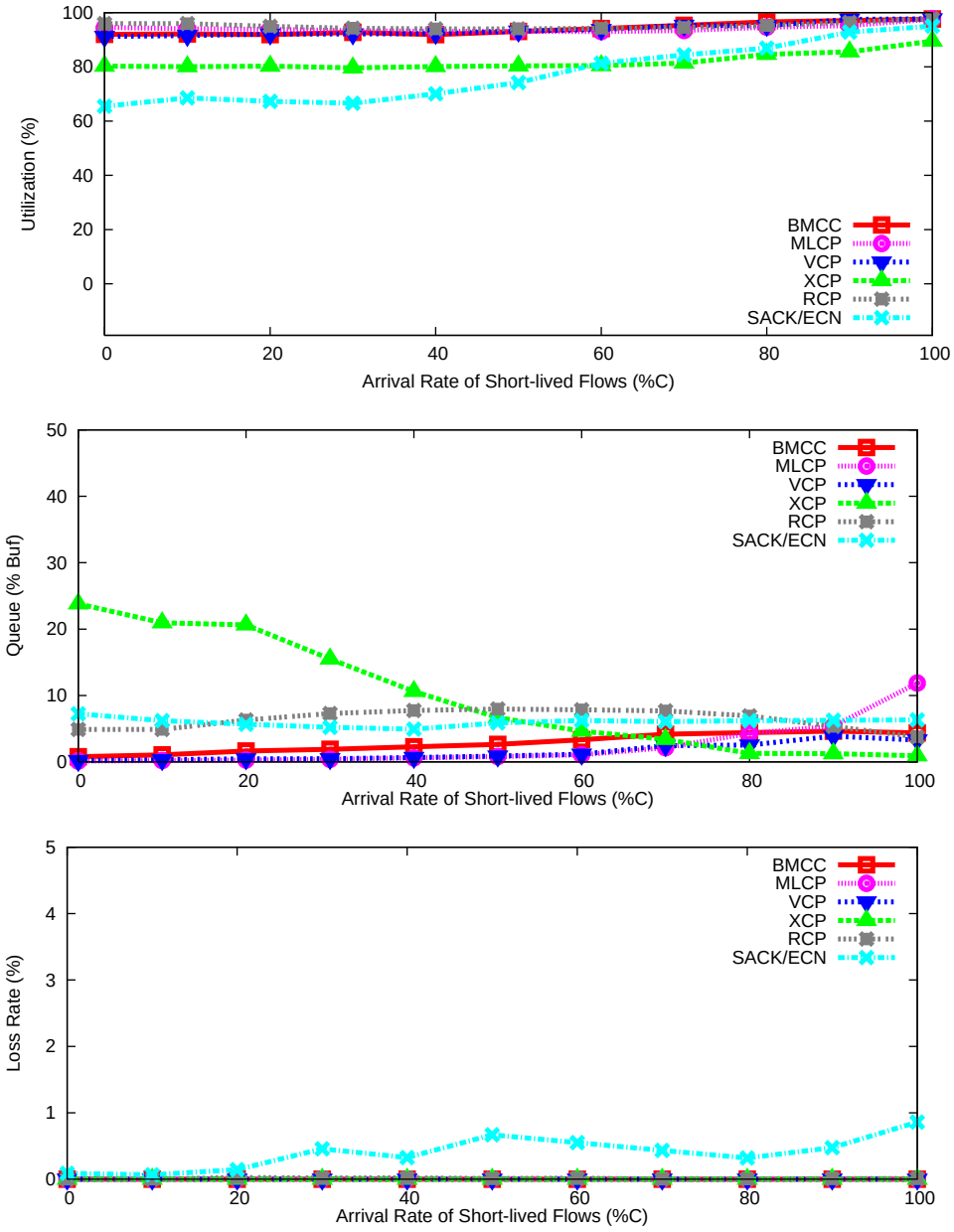


Figure 34: Impact of varying the offered load of short-lived, web-like traffic from $0.1C_l$ Mbps to C_l Mbps, where $C_l = 155$ Mbps.

ii) Varying Traffic load: Assuming an average file size of 30 KB for web-like flows, we now vary their offered load from 0% to 100% of the bottleneck capacity while maintaining 5

long-lived flows in either direction. Figure 34 shows that BMCC, MLCP, VCP and RCP are able to maintain high utilization ($\geq$90%) while maintaining low persistent queue length ($\leq$5% BDP) and negligible packet loss rate. Average utilization with XCP is less $\sim$10% when compared with BMCC and for SACK the difference is as large as 25% under low loads. Average queue length for XCP is higher (10$\sim$25% BDP) for loads $<$50%. The relatively small loss rate for SACK $\sim$0.5%-1% is due to the presence of RED/ECN at the routers.

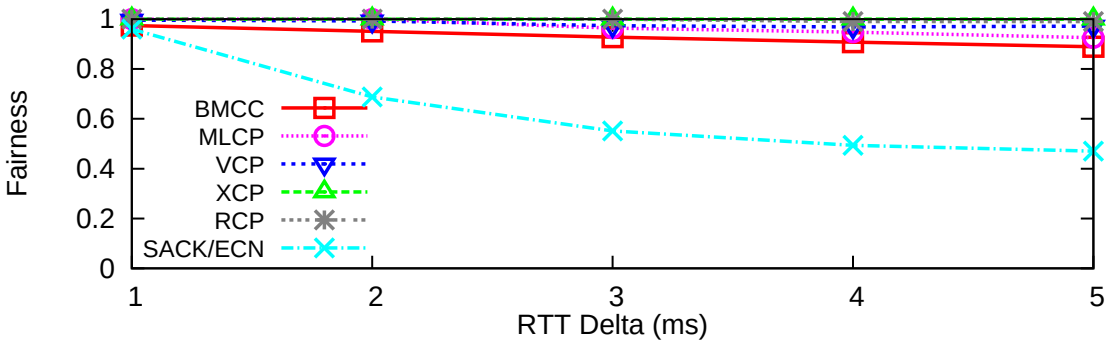


Figure 35: Jain's fairness index $[(\sum_{i=1}^{N} x_i)^2/(N\sum_{i=1}^{N} {x_i}^2)$, where x_i is the throughput of flow i and $i \in \{1, .., N\}]$ as a function of δ.

4.6.5 Fairness

We now compare the fairness properties of BMCC with other schemes. We consider a single bottleneck link of capacity 60 Mbps with 20 long-lived flows in either direction. Each forward flow j's RTT is chosen according to $T_j = 40 + 4j\delta$ ms for $j = 0,\ldots,19$, where δ is the one-way propagation delay for a non-bottleneck link. We perform simulations with δ varying from 1 ms to 5 ms. When δ is 1 ms, RTTs are in the range [40 ms, 116 ms]. When δ is 5 ms, the RTTs are in the range [40 ms, 420 ms]. Observe that BMCC, MLCP, VCP, XCP and RCP achieve high level of fairness ($\geq$0.9) across a large range of RTT variations. SACK, however, becomes very unfair in the presence of RTT heterogeneity.

4.7 RELATED WORK

In this section, we discuss prior works that are most directly related to that of BMCC.

4.7.1 Packet Marking Schemes

The first schemes to convey a continuous value using a single bit per packet [76, 110, 111] randomly mark packets with a probability dependent on the value to be sent. These schemes require at least $n-1$ packets to signal n different values, like unary coding, and implicitly sum the values at routers on the path.

The "side information" contained in the IP header was first used for setting the ECN bits in [106], which considers the time-to-live (TTL) field. This idea was extended by Thommes and Coates [107], who provided an efficient, deterministic marking algorithm, using the 16-bit IP packet identifier ("IPid") to allow the value to be encoded base-two. Following that, [112] proposed a similar scheme for estimating the maximum value, appropriate for max-min flow control. These deterministic marking schemes provide estimators that potentially have a much lower MSE than the random marking schemes [107, 112].

All these schemes must specify *a priori* how to trade resolution for agility. Random marking schemes must choose the interval over which to average random marks, while deterministic schemes use a fixed quantizer, which means that the MSE is poor until sufficient packets have been processed [105].

ADPM implicitly adapts its effective quantization resolution based on the dynamics of the value. Analysis and numerical results in [105, 108] show that static values can be estimated precisely, whilst rapidly changing values can be tracked quickly. These results also show that ADPM provides a MSE that is several orders of magnitude smaller than the estimators based on random marking of packets [76, 110, 111] or deterministic marking with static quantization [107, 112].

4.7.2 Recent Protocols and/or Frameworks

We now discuss some more recent related works, which were done either after or in parallel to our work. We discuss each of these works in turn.

Nedeljko et. al. [41] proposed a framework called UNO that uses the IP identification field to convey the load factor and RTT information using the existing ECN bits. For conveying the load factor, each router examines the 3 least significant IPID bits of the incoming packets and sets the ECN bit if they match any of the 3-bit representation of the load levels. Since UNO requires exact matches, this means senders need to wait for potentially large number of packets to obtain the maximum load factor and adjust their congestion window sizes. With our framework, routers provide bounds on the load factor, which means that the Mean Square Error (MSE) is low even if small number of packets are received [105]. This allows sources to adjust their congestion window sizes right a way. Moreover, UNO's 3-bit quantization does not allow for load representation in the overload region and therefore, does not lend itself to multiple backoff factors. This could be dealt with by considering more IPID bits, however, this would require sources to wait for much larger number of packets before they can obtain the maximum load factor and react to it. This may be a serious concern, especially for short flows. To convey the RTT information, UNO requires the usage of both the ECN bits. This makes it incompatible with the existing interpretation of the ECN bits.

Li et. al [42] proposed the Multi Packet Congestion-control Protocol (MPCP) which uses only the ECN bits to convey the load factor information. They achieve this by concatenating the set of ECN bits in a packet chain. This requires MPCP to do segmentation and reassembly at the routers as well as at the end-hosts, which increases router complexity. Moreover, packet reordering and packet loss makes it challenging for MPCP to performance well under diverse settings. BMCC, on the other hand, does not require segmentation or reassembly either at the routers or at the end-hosts.

Most recently, Alizadeh et. al. proposed DCTCP for use in data center networks [9]. DCTCP leverages the ECN bits to mark packets during times of congestion. This marking is based on the average queue length observed at the routers. Using the the fraction of marked packets, DCTCP

receivers infer the *degree* of congestion at the bottleneck. DCTCP sources adaptively backoff with a factor that depends on the degree of congestion. This allows the protocol to maintain low queues and avoid problems such as TCP incast in data center networks [11]. Other than this, DCTCP uses the same algorithms and parameters as TCP. Since it doesn't change TCP's increase policy, DCTCP doesn't address TCP's slow convergence and fairness issues in large BDP networks. BMCC, on the other hand, not only employs adaptive backoff factors but also uses different control laws and a richer congestion signal which allows it to performance well across a diverse range of network scenarios.

4.8 SOFTWARE

The ns2 code for BMCC is publicly available for download at the following URL: http://www.cs.pitt.edu/~ihsan/bmcc-0.1.tar.gz.

4.9 SUMMARY

This chapter presented the design, analysis and simulation evaluation of BMCC; a congestion control protocol that uses a packet marking scheme and the existing ECN bits to obtain congestion estimates of up to 16-bit resolution. BMCC achieves high utilization, low persistent queue length and negligible packet loss rate on high BDP paths and enables flows with different round-trip times to achieve max-min fairness. We presented analytical models that predict and provide insights into the convergence properties of the protocol. We also evaluated mechanisms for reducing the overhead of TCP options in conveying the load factor estimates from the receiver back to the sender. The resulting algorithm, employed by BMCC, introduces little overhead and is robust to lost ACKs. Using extensive packet-level simulations, we assessed the efficacy of BMCC and performed comparisons with several proposed schemes. BMCC outperforms VCP, MLCP, XCP, SACK+RED/ECN and in some cases RCP, in terms of average flow completion times for typical Internet flow sizes.

5.0 INCREMENTAL DEPLOYMENT

In this chapter, we study how BMCC can be incrementally deployed on the Internet. In particular, we investigate the performance of BMCC when deployed in conjunction with TCP SACK and Drop-Tail or RED/ECN routers. We show that TCP flows can starve BMCC flows when sharing a BMCC-enabled bottleneck whereas the converse holds true when they share a non-BMCC bottleneck. To address the former case, we present simple router algorithms that prevent starvation and allow fairer bandwidth sharing between BMCC and TCP. These algorithms have applicability beyond that of BMCC. For the latter case, we propose mechanisms for BMCC flows to detect the bottleneck type (which we show is feasible) and shift to TCP mode. We believe such switching has a useful role to play in the migration towards more efficient congestion control.

The rest of the chapter is organized as follows. In Section 5.2, we discuss why BMCC is amenable to incremental deployment on the Internet. We then analyze the performance of BMCC under different deployment scenarios in Section 5.3. In Section 5.4, we propose and evaluate solutions to overcome incremental deployment issues. Finally, we offer concluding remarks in Section 5.5.

5.1 CONSIDERATIONS FOR INCREMENTAL DEPLOYMENT

Protocols that use explicit feedback from the network typically require changes in the end-hosts as well as routers [6, 8, 74]. These changes, however, cannot be done overnight. Thus, it is important that a new protocol is able to run over existing infrastructure, and share routers with existing protocols. Moreover, it is also important that a protocol performs well when congestion occurs at a device, such as a firewall, which does not provide any feedback.

Since many new protocols aim to avoid queueing, they are likely to be starved by Reno-like flows which continue to increase their rate until the buffer overflows. Conversely, if a new protocol decreases its rate less (or increases more) than Reno, then it is likely to starve Reno-like flows if a congested resource does not provide feedback.

It is often proposed that the former problem be solved by having separate queues for packets which do and don't support the new protocol [75]. However, that raises complex management issues and unnecessarily increases the cost. Many proposals for high bandwidth-delay products include a "compatibility mode" which revert to Reno-like behavior when the bandwidth-delay product is low. Similarly, it is possible for a new protocol to seek to detect the presence of congestion at points which do not provide explicit feedback, and revert to a loss-based mode. Problems may arise if the detection process fails, but such mode switching may have a useful role in the migration towards more efficient congestion control.

5.2 WHY BMCC?

BMCC is particularly suited to deployment in the Internet because its signalling is compatible with the existing Internet Protocol (IP) packet header.

Many protocols, such as XCP [6], RCP [75], MaxNet [74] and MLCP [57] require that routers send a (quantized) real number indicating the amount of congestion. This requires additional fields, either in an IP option, a TCP option [74] or modified header [6], or a "shim layer" [75]. None of these can be universally deployed because many routers are configured to drop packets containing IP options, and IP payloads may be encrypted.

An alternative, used by ECN [55] and VCP [56] is to squeeze two extra bits into the IP header. Because they signal only binary [55] or ternary [56] congestion indication, these schemes provide minimal benefit, although [56] allows low mean queue size at the expense of slow convergence [10].

BMCC uses these bits to send a continuous congestion signal by coding the value over multiple

packets, using ADPM [105]. This is an enhancement of random marking [110, 76, 106] and of Thommes and Coates's scheme [107] to achieve a wider dynamic range of signals using few packets. This allows BMCC to be IP-compatible while achieving fast convergence. BMCC achieves efficient and fair bandwidth allocations on high bandwidth-delay product paths while maintaining low queues, negligible packet loss rate and small average flow completion times [58, 10].

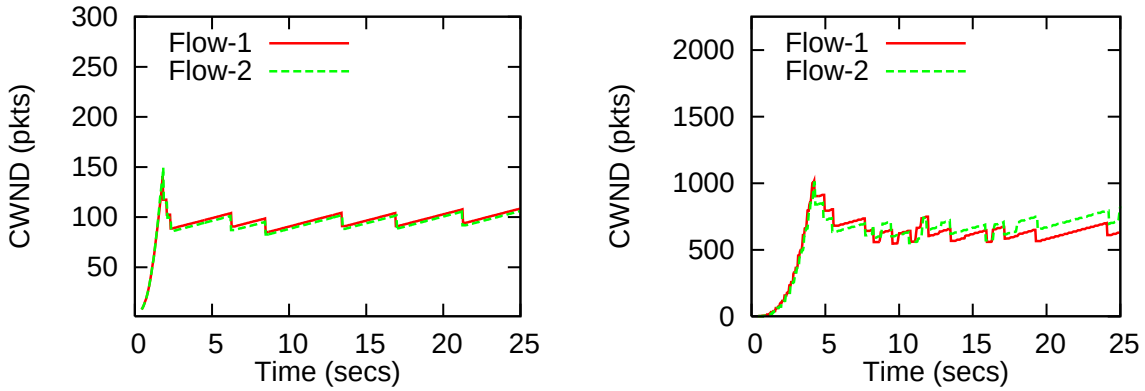
5.3 PERFORMANCE EVALUATION UNDER DIFFERENT PARTIAL DEPLOYMENT SCENARIOS

In this section, we study the performance of BMCC under different partial deployment scenarios and protocol mixes using ns-2 simulations. First, we study the performance of BMCC under Drop-Tail and RED (with ECN support) routers. We then add SACK flows to the existing BMCC traffic and study their interaction under different bottlenecks. Second, we study the performance of a mix of protocols under a BMCC-enabled bottleneck.

In all simulations, the non-bottleneck routers are assumed to be Drop-Tail. Unless explicitly stated otherwise, the bottleneck capacity, C, and the round-trip propagation delay, T, are set to 45 Mbps and 40 ms, respectively. The capacity of the non-bottleneck links is set to $10{\cdot}C$ Mbps. The buffer size of all routers is set to the bandwidth-delay product. We always maintain bidirectional cross traffic, with an offered load equal to 10% of the bottleneck capacity. The cross traffic arrives according to a Poisson process and has sizes that follow the Pareto distribution with an average file size of 30 kB and a shape parameter of 1.2 [97]. All flows use a data packet size of 1 kB.

5.3.1 Performance over non-BMCC routers

In this section, we evaluate the performance of BMCC over non-BMCC bottlenecks such as Drop-Tail and RED routers. To aid comparison between different bottleneck types, we first present results of BMCC's performance under BMCC-enabled bottleneck routers. Figure 36 shows the variations in the congestion window size of two BMCC flows that traverse a BMCC-enabled bottleneck, for $T = 40$ ms and $T = 300$ ms. Observe that BMCC introduces little variations in the



(a) T = 40 ms (b) T = 300 ms

Figure 36: Congestion window size of two BMCC flows passing through a BMCC router for T=40 ms and T=300 ms, respectively.

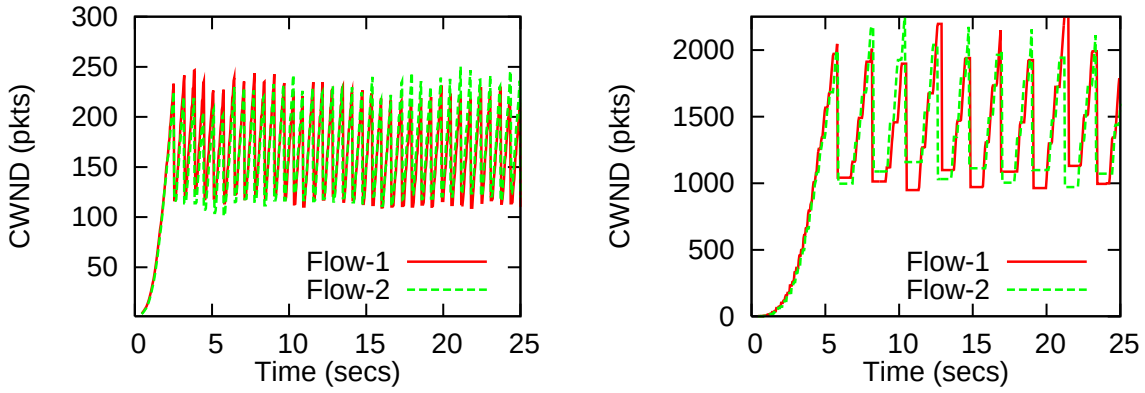
congestion window sizes. This happens because BMCC sources backoff by a small factor when there is low statistical multiplexing, which is typically indicated by lower load factor values in overload. This helps BMCC in sustaining high throughput without incurring large variations in it.

5.3.1.1 BMCC over Drop-Tail Figure 37 shows the variations in the congestion window size of two BMCC flows, passing through a Drop-Tail bottleneck, for $T = 40$ ms and $T = 300$ ms. In this case, BMCC sources increase their windows by a factor of 3 per t_p[1] until a packet gets dropped at the bottleneck, at which time sources backoff by a factor of two. This cycle gets repeated because there is no change in the receivers' estimates[2]. In other words, BMCC employs MIMD with a MI factor of $3^{T/t_p}$ per round-trip time and a MD factor of 0.5. Prior work has shown that MIMD does not converge to a fair bandwidth allocation in the presence of synchronous feedback [45].

5.3.1.2 BMCC over RED+ECN Figure 38 shows the variations in the congestion window size of two BMCC flows traversing a RED/ECN bottleneck router. In this case, BMCC flows increase

[1]This happens because new flows assume the initial value of f to be 15%)

[2]Note that with drop-tail routers, all received packets will be unmarked. Hence, there won't be any change in $\hat{f}$ because $h > \hat{f}, \forall h$.

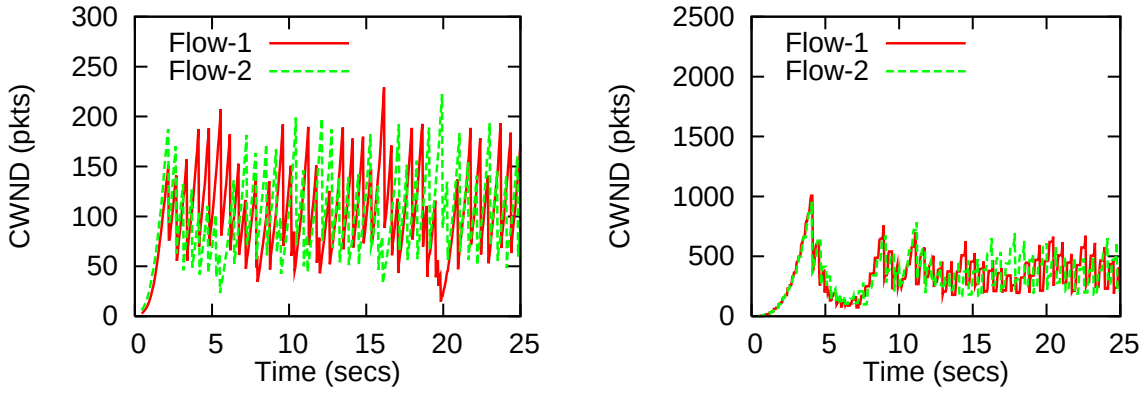


(a) T = 40 ms (b) T = 300 ms

Figure 37: Congestion window size of two BMCC flows passing through a Drop-Tail router for T=40 ms and T=300 ms, respectively

their congestion windows by a factor of $3^{T/t_p}$ per T until overload. In overload, BMCC sources apply MD with a factor of 0.65 (if there is no packet loss) because RED routers mark the ECN bits with the $(11)_2$ symbol that is interpreted by BMCC as severe overload. After which, receiver's estimate decreases until it approaches 15%. Observe that congestion window sizes assumed by BMCC flows are lower than in the case of Drop-Tail routers. This is because RED signals congestion earlier than via packet drops. However, note that higher congestion window sizes in case of Drop-Tail routers do not necessarily imply higher throughput. In fact, increasing the congestion window beyond a certain threshold (depending on the aggressiveness of the source control laws) only increases the RTT of flows without changing the flow throughput. This hurts the performance of other flows due to increased queueing delays.

5.3.1.3 Mix of Protocols over Drop-Tail and RED We now mix three BMCC flows with three SACK flows and analyze their performance under different bottlenecks. Figure 40 and 39shows the congestion window sizes under Drop-Tail and RED routers, respectively. With Drop-Tail routers, BMCC flows grab a much large share of bandwidth than SACK since SACK uses AIMD whereas BMCC uses a much more aggressive MIMD. Note that SACK flows get completely starved when



(a) T = 40 ms (b) T = 300 ms

Figure 38: Congestion window size of two BMCC flows passing through a RED router with ECN support for T=40 ms and T=300 ms, respectively

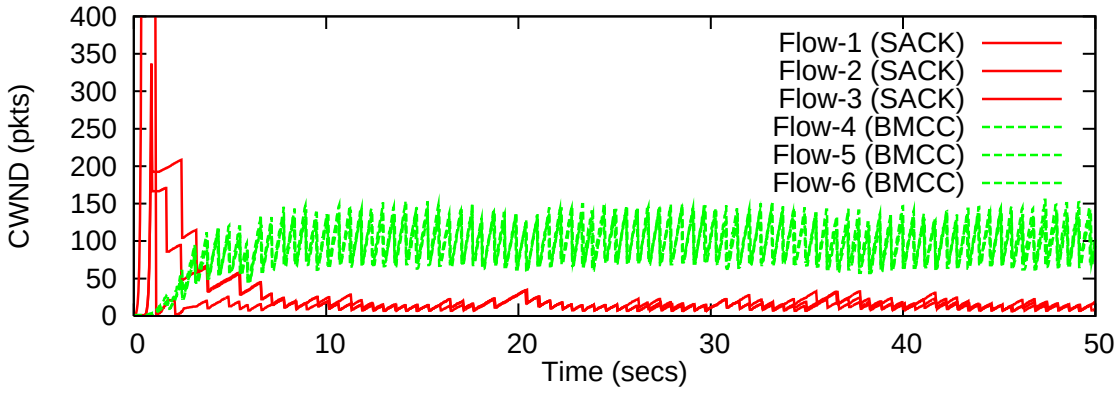
$T = 300$ ms.

In case of RED (with ECN support) routers, performance is similar to the Drop-Tail case. SACK flows achieve small congestion window sizes when $T = 40$ ms and get completely starved when $T = 300$ ms.

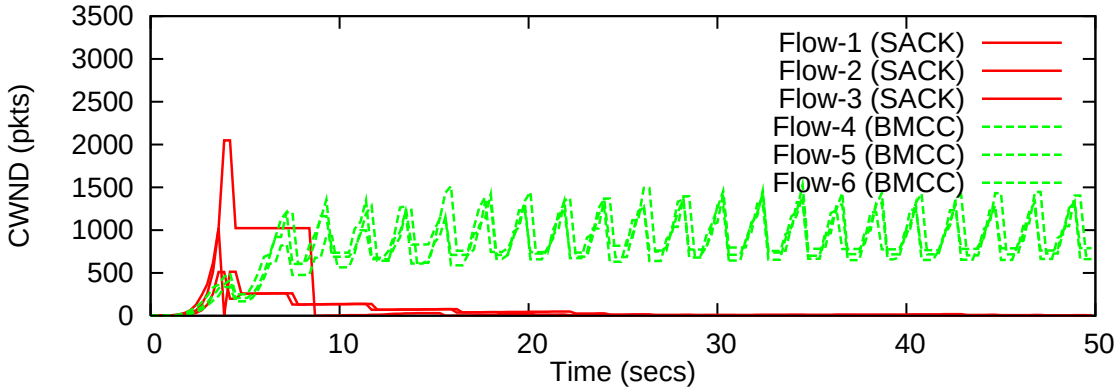
5.3.1.4 Discussion: The above scenarios represent cases where the network topology does not have any (non-bottleneck) BMCC routers. The behavior in the presence of non-bottleneck BMCC routers would depend on the load factor on these routers. For instance, if the average load factor f on these routers remains below 75%, BMCC sources would still apply MIMD with MI factors $1 + \xi(f)$. However, if f is between 75% and 100%, then BMCC sources would apply AIMD. In this case, SACK flows would be able to compete more fairly with BMCC traffic.

5.3.2 Performance over BMCC routers

We now consider a mix of protocols while assuming a BMCC-enabled bottleneck router.



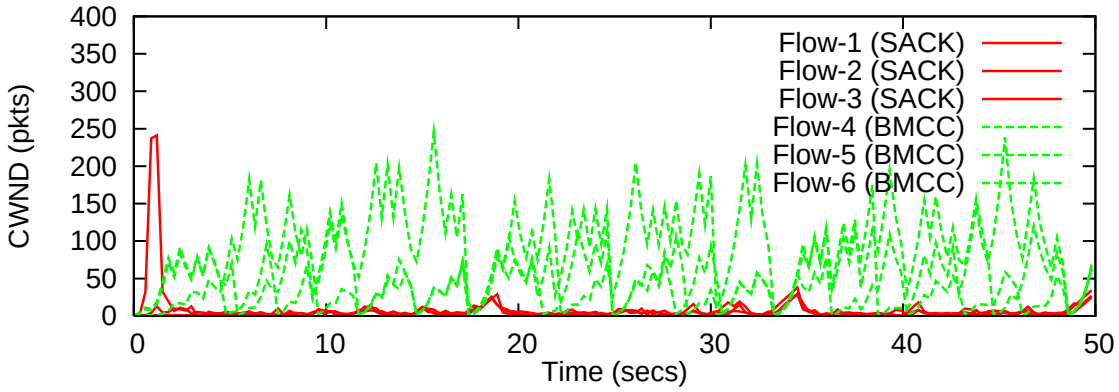
(a) T = 40 ms, Bottleneck = Drop-Tail



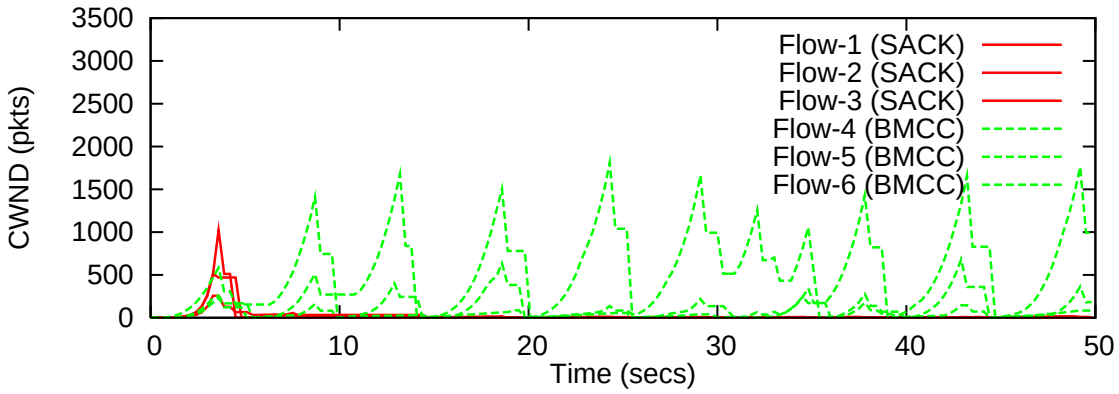
(b) T = 300 ms, Bottleneck = Drop-Tail

Figure 39: Congestion window size of 3 BMCC and 3 SACK flows sharing a Drop-Tail bottleneck for T=40 ms and T=300 ms, respectively.

5.3.2.1 BMCC and SACK We generate 3 SACK and 3 BMCC flows while retaining all the settings as in the previous experiments. All BMCC flows are started at the beginning of the experiment, whereas SACK flows arrive between $10\,s$ and $12\,s$ Observe that the SACK flows completely starve BMCC flows (see Figure 41). This happens because while BMCC aims to avoid queueing, SACK deliberately fills router buffers until there is a packet loss. Therefore, SACK flows maintain high large average queue length at the bottleneck. Since BMCC uses the average queue length to compute the load factor, this gives to a sustained increase in the load factor, causing BMCC flows to back-off very frequently.



(a) T = 40 ms, Bottleneck = RED+ECN



(b) T = 300 ms, Bottleneck = RED+ECN

Figure 40: Congestion window size of 3 BMCC and 3 SACK flows sharing a RED/ECN bottleneck for T=40 ms and T=300 ms, respectively.

5.3.2.2 BMCC and SACK+ECN We now generate 3 ECN-compatible SACK flows and 3 BMCC flows while retaining all the settings as in the previous experiments. Figure 42 shows that in this case also, the BMCC flows get completely starved by SACK flows. In this case, when the load factor exceeds 100%, BMCC flows back-off probabilistically using ADPM. Since ADPM allows quick detection of overload, this causes BMCC flows to back-off. However, SACK flows continue to grab more bandwidth by increasing their rate until load factor exceeds 120% (after which SACK flows receive the $(11)_2$ mark and back-off). When $\hat{f} \in [100, 120]$, BMCC flows back-off repeatedly, leading to very low congestion window sizes.

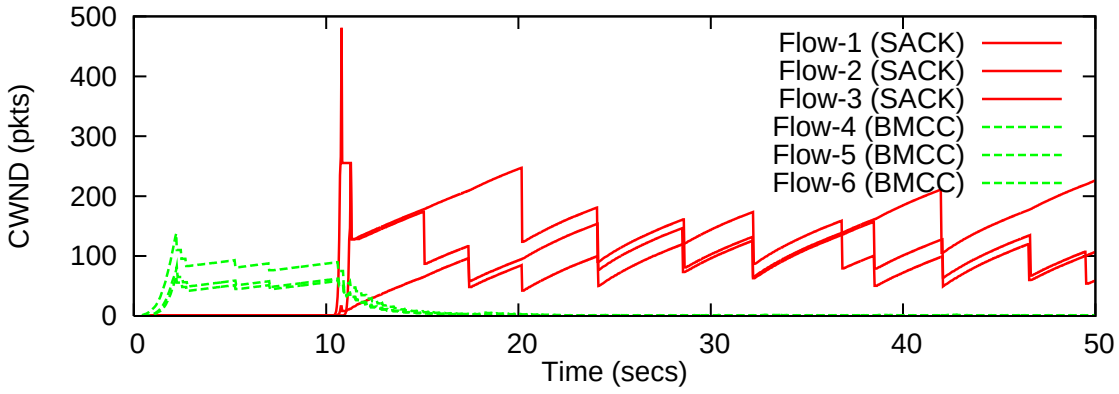


Figure 41: 3 SACK and 3 BMCC flows sharing a BMCC-enabled bottleneck link

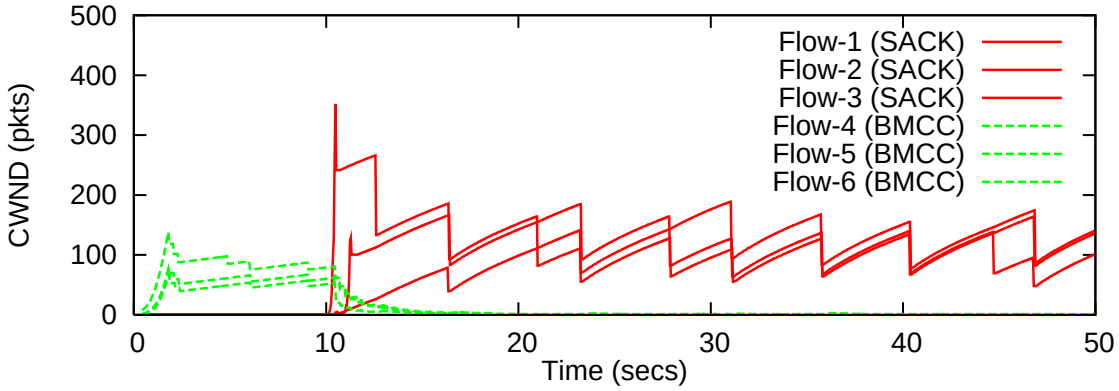


Figure 42: 3 SACK+ECN and 3 BMCC flows sharing a BMCC-enabled bottleneck link

5.3.3 Summary

The results in this section show that when BMCC and SACK flows share a non-BMCC bottleneck (in this case, a Drop-Tail router or a RED/ECN router), BMCC flows can starve TCP traffic. This happens because in the absence of feedback, BMCC sources apply MIMD whereas TCP continues to use the AIMD control law. This can cause TCP flows to achieve very low throughput.

On the other hand, when BMCC and SACK flows share a BMCC-enabled bottleneck, SACK

flows can starve BMCC traffic. This is due to the fact that BMCC flows aim to avoid queueing whereas SACK flows maintain high standing queues. This causes BMCC flows to backoff very frequently, resulting in very low throughput.

5.4 ALGORITHMS FOR IMPROVING BANDWIDTH SHARING BETWEEN TCP AND BMCC

In the previous section, we observed that SACK flows (with and without ECN support) starve BMCC flows. This is due to the fact that SACK flows maintained high persistent queue length, leading to sustained value of load factor above 100%. This caused BMCC flows to back-off multiple times, leading to starvation. In this section, we discuss a number of solutions and their shortcomings. We then propose one solution and show that it is effective in handling the issues raised in the previous section.

5.4.1 Deployment over BMCC bottlenecks

Deterministically marking/dropping packets in overload: One possible solution to the above problems is to disable ADPM in overload and mark ECN-capable transports with the $(11)_2$ symbol and drop packets from non-ECN-capable transports. However, this change will force BMCC flows to use a single back-off factor. This will remove the benefit that BMCC has of dynamically adapting the back-off factors depending on the load at the bottleneck, which has a significant impact on network utilization and convergence rates. Moreover, this solution is likely to starve or result in very low throughput for SACK flows with $T \ll 200\,ms$. The reason is that BMCC routers keep load factor estimates for $t_p = 200\,ms$. A small RTT flow (e.g., with $T = 40\,ms$) will reduce its congestion window multiple times in one t_p, leading to very small windows. Note that reducing t_p is not likely to help because that would cause large variations in traffic loads due to the burstiness induced by sources with $T \gg t_p$.

Probabilistically marking packets with the $(11)_2$ symbol: We can prevent starvation of BMCC flows when sharing a link with SACK+ECN flows by probabilistically marking (or drop-

ping) packets with the $(11)_2$ symbol when $f \in [100\%, 120\%]$. Packets can be marked (or dropped) if a packet was marked (with $(01)_2$) using ADPM. However, this would cause most BMCC flows to back-off by a factor of 0.65, thus reducing the benefit of small decreases in low overload. Also, ADPM yields a marking/dropping probability close to 0.5 which is too large and is likely lead to multiple window reductions as in the above case. To remedy this situation, one could mark/drop packets with low probability when overload is small and increase this probability as f approaches 120%.

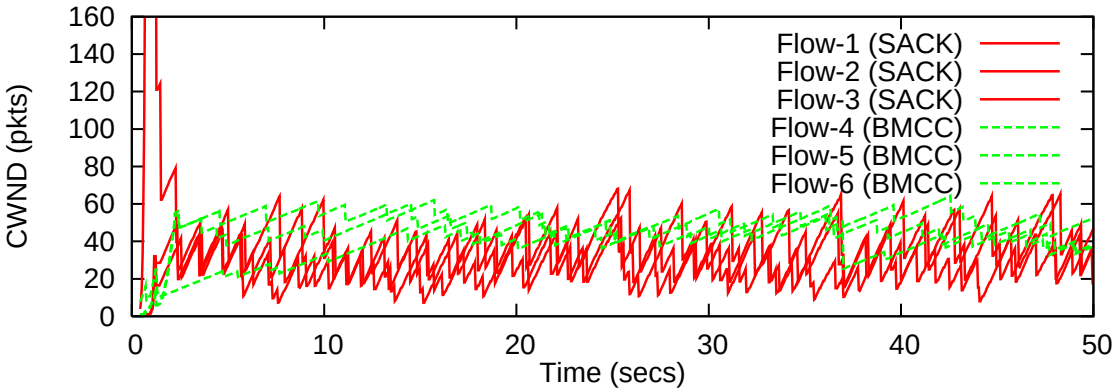


Figure 43: 3 SACK+RED/ECN and 3 BMCC flows sharing a single modified BMCC bottleneck link (T = 40 ms)

5.4.2 Modified BMCC Router

To retain the benefit of using ADPM in overload for BMCC traffic while preventing starvation of SACK and BMCC flows when traversing a common BMCC-enabled bottleneck, the following ingredients are desirable in a solution: (1) packets should be marked (or dropped) probabilistically rather than deterministically and (2) the fact that BMCC routers keep load factor estimates for one t_p should not impact other flows, which suggests that the marking procedure should be decoupled from load factor computation.

To achieve the above ends, we make the following modifications to the BMCC router. In over-

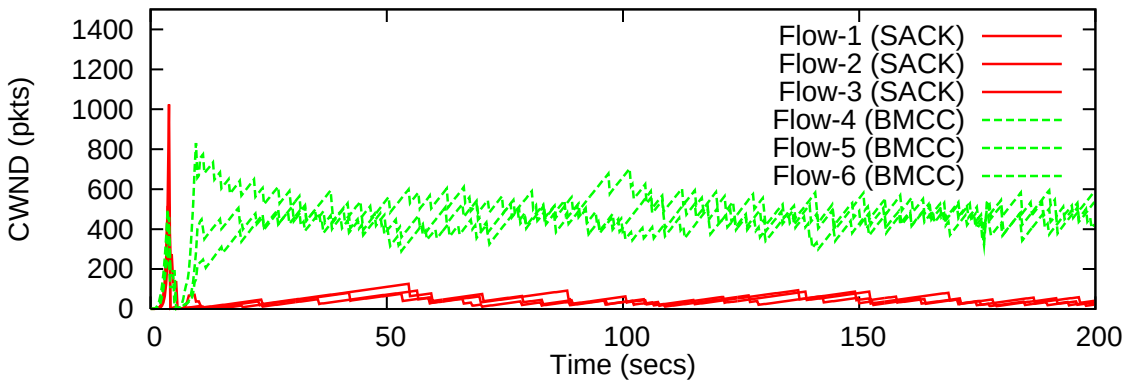


Figure 44: 3 SACK+RED/ECN and 3 BMCC flows sharing a single modified BMCC bottleneck link (T = 300 ms)

load, we replace the packet marking policy in BMCC routers with the Adaptive RED algorithm[3] while continuing to mark packets with ADPM using load factor computation. Recall, ADPM *only* marks packets with the $(01)_2$ symbol and therefore, doesn't interfere with the RED algorithm or the the standard ECN behavior. This ensures that BMCC sources continue to obtain precise load factor estimates. To retain the benefit of multiple back-off factors, BMCC sources are made to back-off only when they receive the $(11)_2$ symbol and $\hat{f} \geq 1$, where the back-off factor depends solely on the load factor estimate at the receiver. Now since the marking (or dropping) probability is the same for SACK and BMCC flows, they back-off roughly the same number of times leading to a more fair bandwidth sharing.

Figure 43 shows the congestion window sizes of three SACK flows sharing a BMCC-enabled bottleneck with a modified BMCC router. Observe that SACK flows are now able to effectively share bandwidth with BMCC traffic. The BMCC flows obtain the benefit of multiple MDs by applying small back-off factors in low overloads whereas SACK behavior remains largely unchanged. Figure 44 shows the congestion window sizes of flows on a path with $T = 300\,ms$. The bandwidth-delay product of this path is about 1700 packets, which is roughly eight times larger than the previous scenario. In this case, BMCC obtains much larger throughput than SACK flows

[3]We chose RED because it is the most widely deployed AQM scheme. However, any AQM could be used.

because it acquires the spare bandwidth much faster than SACK flows.

In order to achieve max-min fairness, BMCC flows scale their MI and AI parameters. This scaling depends on the ratios T/t_p and $(T/t_p)^2$ for MI and AI, respectively. This implies that on long RTT paths, BMCC flows will achieve higher throughput than SACK flows due to larger α, β, and ξ parameter values[4]. On the other hand, it is well-known that SACK flows achieve throughput proportional to $1/RTT^z$, where $1 \leq z \leq 2$ [113]. This implies that with SACK, short RTT flows can achieve much higher throughput than long RTT flows. We now compare the unfairness in these two cases. In the first case, we characterize the throughput achieved by BMCC and SACK flows when they share a bottleneck and in the second case, we characterize the throughput achieved by two SACK flows with different RTTs.

To quantify unfairness, we measure the gain, G, in the throughput of a more aggressive flow at the expense of a loss, L, in the throughput of a lesser aggressive flow. In the first experiment, called *BASELINE*, we run two SACK flows sharing a 45 Mbps link with a round-trip propagation delay of T. In the second experiment, we run one SACK flow and one BMCC flow, each RTT, T. In the third experiment, we run one SACK flow with RTT, T and another SACK flow with an RTT of $10\,ms$. The gain achieved by the aggressive flow and the loss incurred by a less aggressive flow is given by

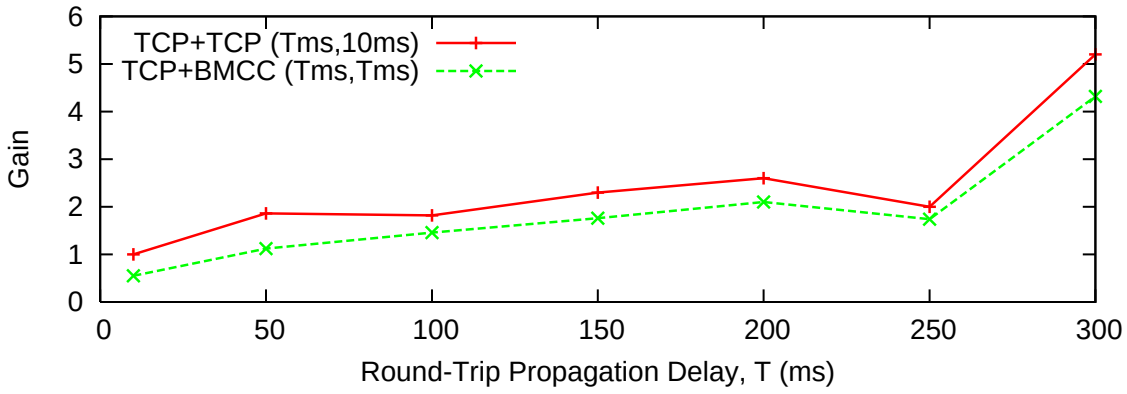
$$G = T(B)_{Mix}/T(B)_{Baseline},$$

$$L = T(A)_{Mix}/T(A)_{Baseline}$$

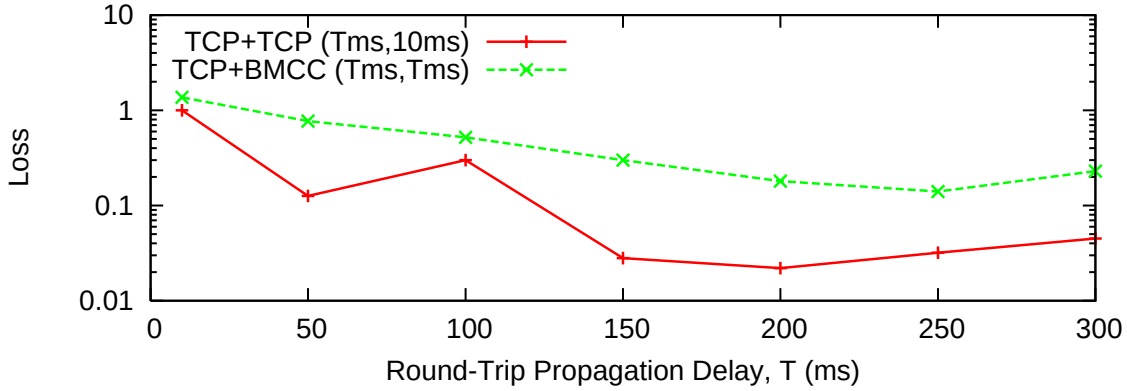
respectively, where $T(A)$ is the throughput of a less aggressive flow and $T(B)$ of a more aggressive flow [114].

Figure 45 shows that the bandwidth gained by a BMCC flow over a SACK flow is less than the bandwidth gained by a SACK with $T = 10\,ms$, across a range of round-trip times. Figure 45 shows that the bandwidth lost by SACK flows is roughly an order of magnitude larger when the bottleneck is shared by a $10\,ms$ RTT flow compared to a BMCC flow.

[4]Note that this does not necessarily mean that BMCC flows do not achieve higher than SACK throughput on very small RTT paths. The fast MI phase and high β values used by BMCC allows it to achieve high throughput in such cases.



(a) Bandwidth Gain



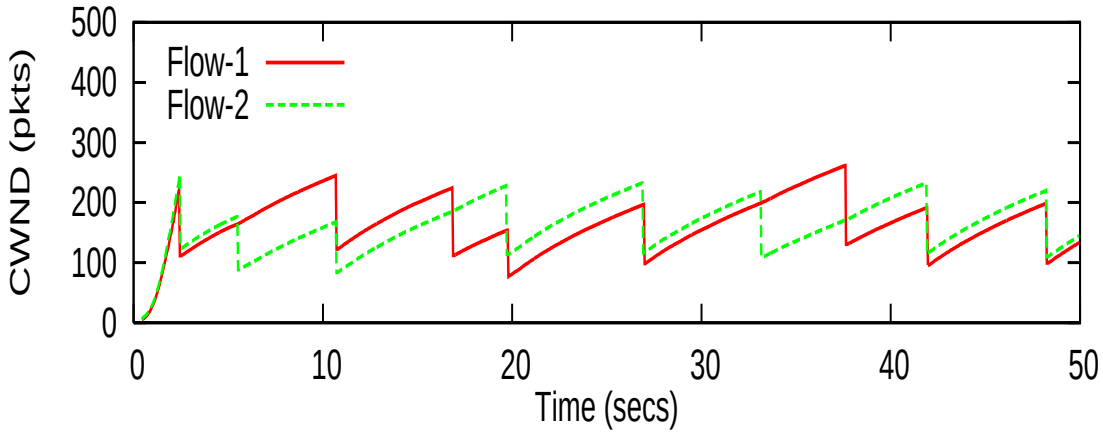
(b) Bandwidth Loss

Figure 45: Bandwidth Gain and Loss for a BMCC (RTT=T ms) and a SACK (RTT=10 ms) flow over a SACK flow (RTT=T ms), respectively

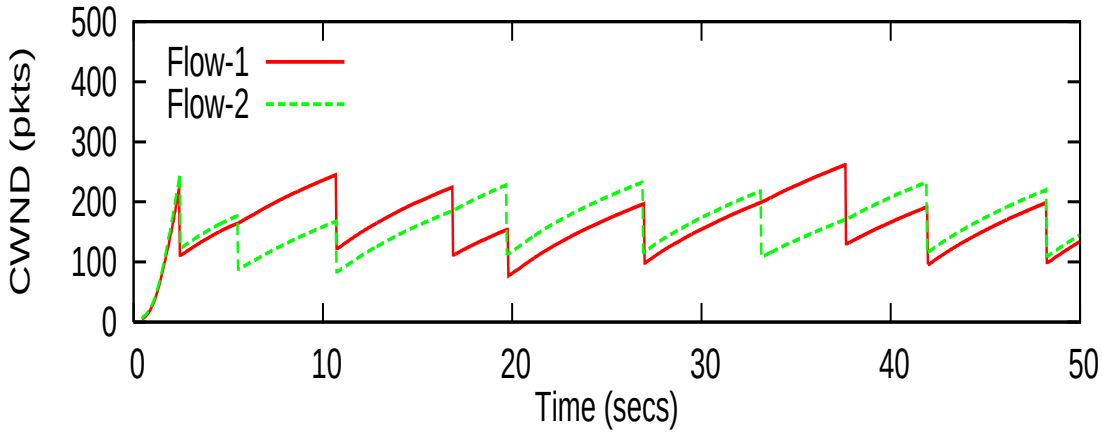
5.4.3 Deployment over non-BMCC bottlenecks

Simulations results in the previous section showed that when the path of a flow contained only non-BMCC routers, BMCC applies MIMD. This leads to degraded throughput and in some cases starvation for SACK flows. To remedy this situation, we now present heuristics for detecting non-BMCC bottlenecks for switching to SACK TCP.

When a BMCC router is congested, it signals a high load factor, f. If other signals of congestion, such as packet loss or $(11)_2$ marks are detected when the load factor is low, this indicates



(a) Bottleneck = Drop-Tail



(b) Bottleneck = RED+ECN

Figure 46: Two BMCC flows (with heuristic detection) sharing a non-BMCC bottleneck with T = 40 ms.

congestion at a non-BMCC router. We propose the following rules:

- If a BMCC source detects a packet loss or $(11)_2$ mark when its estimate of the bottleneck load factor is less than 100%, then it falls back to SACK operation.
- If a BMCC source detects a load factor in excess of 100%, it exits SACK compatibility mode and resumes normal operation.

Figure 46 shows the congestion window size of two BMCC flows (with this heuristic detection

rule) traversing a bottlenecked Drop-Tail router. Note that unlike as in Figure 37, BMCC quickly detects the condition and switches to SACK.

If a BMCC flow halves its window in response to the packet which caused it to enter SACK mode, this rule gives rise to the anomalous situation that the window reduction is *greater* if $\hat{f} < 1$ than if $\hat{f} > 1$. Thus, we propose that the loss or mark which triggers the entrance to SACK mode is not itself interpreted as a congestion signal. Note also that the estimate of f signalled by ADPM lags the value signalled by the routers [108], it is possible for BMCC router to estimate the load factor to be slightly below 100% when the router's value is actually greater, and its RED policy emits a $(11)_2$ mark. This will cause the flow to mistakenly enter SACK mode. However, it will to return to normal mode soon afterwards, as the load factor is already high.

5.5 SUMMARY

In this chapter, we showed that BMCC can be incrementally deployed on the Internet, without changing the IP Header or the addition of a "shim" layer. End-host and router softwares can be gradually updated in a similar way as ECN was deployed. End-hosts using BMCC can immediately get high throughput and faster downloads when traversing BMCC-enabled bottlenecks without significantly hurting standard TCP traffic. This is an important incentive for end-users and router vendors to deploy BMCC. Moreover, research studies have shown that the throughput of most Internet users is limited by their "last mile" or access links. This suggests that BMCC's deployment can begin by updating such links and users can immediately take advantage of BMCC's high performance.

We evaluated the performance of BMCC under different deployment scenarios and a mix a protocols. We showed that when the bottleneck is not BMCC-enabled, SACK flows achieve low throughput and can even get starved. On the other hand, when the bottleneck is BMCC-enabled, SACK flows starve BMCC flows. We proposed and evaluated solutions which prevent starvation and allow for a fairer bandwidth sharing between BMCC and SACK flows.

6.0 ADDRESSING PERFORMANCE CHALLENGES OVER 802.11 WIRELESS LANS

In the last few years, there has been a proliferation of wireless technologies e.g., 802.11-based Wireless LANS (or WiFi), WiMAX, and 3G/4G networks. This is evidenced by the fact that typical smartphones today are shipped with at least two wireless interfaces: a 3G interface and a WiFi interface [115, 116]. Moreover, hotspots, enterprises, and residential networks are increasingly using WiFi for providing last hop connectivity [117]. Therefore, a typical end-to-end path today is likely to contain a wireless segment.

While the capacity of a wired link is generally fixed, it is not the case in wireless networks such as WiFi. In wireless networks, the capacity of a wireless "link"[1] changes over time and depends on two factors: (1) radio propagation and (2) the level of interference/contention due to other links in the wireless neighborhood[2]. The radio propagation, in turn, depends on three factors: "*path loss*", a deterministic power law decay of the signal that depends on the distance between the sender and the receiver, "*shadowing*", that are variations in signal strength due to obstacles and reflections, and "*fading*", which are fine-grained signal variations in both space and frequency [118]. The level of interference depends on the signal strength and the offered load due to other nodes in the wireless neighborhood.

BMCC and other congestion control protocols such as XCP [6], RCP [8], MaxNet [72], and VCP [56] require an estimate of the capacity for feedback computation. When a wireless segment in the path of a flow becomes the bottleneck, the end-to-end performance depends on the accuracy of wireless capacity estimates. Inaccurate feedback can lead to under-utilization or network overload and unfairness between flows.

[1] We use the term *wireless link* to denote a transmitter-receiver pair.
[2] This is because the wireless medium is a shared resource

In this chapter, we present a simple model to understand the key factors that affect the available capacity of wireless nodes in a WLAN. We then analyze the performance of BMCC over end-to-end paths that contain a wireless segment and quantify the impact of inaccurate capacity estimates. In particular, we focus on WiFi networks, as they are the most widely used technology today. Finally, we present an accurate method for available capacity estimation and through extensive packet-level simulations, we show that it enables good end-to-end performance.

6.1 MODELING LINK CAPACITY

In this section, we discuss the key factors that affect the available capacity of a wireless node with the help of a simple model. We consider a 802.11 based Wireless LAN with multiple clients and a Base Station (BS). Each of these clients can have different channel conditions to the BS. The differences in channel conditions across clients mean that each client can have different capacities to the BS. We assume each node can use different physical layer (PHY) transmission rates to adapt to the time-varying and location-dependent channel quality, a capability that is mandated by the current 802.11 specifications. We start by providing a brief overview of the 802.11 Distributed Coordination Function (DCF).

6.1.1 802.11 Distributed Coordination Function (DCF)

The 802.11 DCF is a random access scheme based on the carrier sense multiple access with collision avoidance (CSMA/CA) protocol. When a station has a new packet to transmit, it monitors the channel activity for the DCF Interframe Space (DIFS) period. If the channel is sensed idle during this time, the station transmits. Otherwise, if the channel is sensed busy (either immediately or during the DIFS), the station persists to monitor the channel until it is measured idle for the DIFS period. At this point, the station generates a random backoff period $B_i \in [0, 2^i CW_{min}]$ slots[3] where $i \in \{0, 1, .., 10\}$ (initially set to zero) and CW_{min} is the minimum contention window size. The station then counts down this backoff timer. If the channel is sensed busy during this time,

[3]for 802.11b the slot duration is 20μs and CW_{min}= 32 slots.

Terms	Description
C_l	network layer capacity of link l
C_l^{MAC}	MAC layer capacity of link l
R_l^{MAC}	PHY transmission rate used on link l
p_l	channel loss rate on link l
$A_{S\setminus l}$	fraction of the air time used by transmissions on $S - l$
S	set of all links in the WLAN
T_l^{ov}	per-transmission MAC overhead on link l
T_l^{tr}	average transmission time of a MAC frame on link l
σ	802.11 slot duration
t_p	measurement interval
T^{ack}	transmission time of a MAC layer ACKs
T^{pr}	transmission time of the PLCP preamble and the header
T_l^{data}	transmission time of a data frame on link l
T_l^{BO}	average MAC backoff on link l
X^{data}	MAC data frame size in bits

Table 6: Constants and Variables

the backoff timer is frozen for as long as the channel is busy. When B_i reaches zero, the node transmits the packet. When a data packet is received, the receiver waits for the Short Interframe Space (SIFS) interval and then transmits an ACK. When an ACK is successfully transmitted, the maximum contention window size is set to CW_{min}. In addition, to avoid channel capture, a station must wait a random backoff time between two consecutive new packet transmissions, even if the medium is sensed idle in the DIFS period [119].

Upon every successive packet loss, senders double their maximum contention window size (or increase i by one) until the maximum possible of the contention window size is reached.. For broadcast packets, there are no link-layer ACKs and no exponential backoffs. Note that the 802.11

DCF introduces a variable amount of overhead on every packet that it transmits. To accurately model the link capacity, each of these overheads must be carefully accounted.

6.1.2 Link Capacity Representation

Link capacity is defined as the *maximum achievable throughput* on a BS-client link given certain channel conditions and offered load due to other nodes in the WLAN. Suppose the PHY transmission rate (or bitrate) used by a node on link l is R_l bps and the average channel loss rate is p_l. When there no competing transmissions in the wireless neighborhood, the MAC layer capacity C_l^{MAC} of link l is[4]

$$C_l^{MAC} = R_l(1 - p_l) \tag{6.1}$$

When there are competing transmissions from other nodes in the neighborhood, the air time available for transmissions on link l is lower. Suppose $A_{S\setminus l}$ is the fraction of the air time used by transmissions on other links in the WLAN, where S is the set of all links that are part of the WLAN. Then the MAC layer capacity of link l is given by

$$C_l^{MAC} = R_l(1 - p_l)(1 - A_{S\setminus l}) \tag{6.2}$$

In order to compute the network layer capacity, we need to account for all the overheads induced by the 802.11 MAC. These overheads are not static and depend on several factors. Suppose the transmission time of a MAC frame on link l is T_l^{tr} and the MAC layer overhead is T_l^{ov}, then the link capacity is given by

$$C_l = \frac{T_l^{tr}}{T_l^{tr} + T_l^{ov}} \cdot C_l^{MAC} \tag{6.3}$$

We now characterize T_l^{ov}, the MAC layer overhead incurred by each MAC frame.

[4] assuming zero per-packet overhead for now

Figure 47: A typical 802.11 unicast transmission

6.1.2.1 802.11 MAC Overhead Whenever a packet is sent to the 802.11 MAC for transmission, it incurs several overheads as shown in Figure 47. For unicast transmissions, these overheads correspond to the time needed to gain channel access (DIFS + Backoff), synchronize transmissions (PLCP), and transmit a link layer ACK (SIFS + ACK transmission time) [120, 121]. Therefore, the total time taken by the interface to send a data packet is given by $T_l^{tr} + T_l^{ov}$, where $T_l^{tr} = X/R_l$ is the MAC transmission time of the packet, X is the data packet size and T_l^{ov} is given by

$$T_l^{ov} = DIFS + T^{pr} + T_l^{BO} + SIFS + T^{pr} + T^{ack} \tag{6.4}$$

where T^{pr} is the transmission time for the PLCP preamble and header, T_l^{BO} is the average backoff experienced by transmissions on link l, T^{ack} is the ACK transmission time, and DIFS and SIFS are the interframe spacings. For 802.11b, T^{pr}=192 μs[5] (sent at the basic rate), DIFS=50 μs, SIFS=10 μs, ACK=112 μs (112 bits). Note that all the above terms are fixed except T_l^{BO}.

We can now use Equation 6.3 to derive capacity expressions. For instance, consider the download scenario and assume that X_{data}=1500 b, X_{ack}=118 b, R=11 Mbps, and T_l^{BO}=320 μs[6]. This gives us a capacity of 4.1 Mbps for traffic from the BS to the clients. Note that transport layer

[5]This is the transmission time when the long PLCP header is used. With short PLCP headers, T^{pr} is equal to 96 μs

[6]This is the average backoff value *i.e.*, $\sigma \cdot CW_{min}/2$ (= 320 μ) for 802.11b in the absence of frame losses.

ACKs are sent at the same bitrate as the corresponding DATA packets.

6.2 PERFORMANCE ISSUES DUE TO INACCURATE CAPACITY ESTIMATES

Recall that during every time interval t_p, each BMCC router computes the load factor as

$$f = \frac{\lambda + \kappa_1 q_{av}}{\gamma C_l t_p} \tag{6.5}$$

where λ is the amount of traffic received during t_p, C_l is the capacity of the link and $\gamma_l \leq 1$ is the target utilization, q_{av} is the exponentially weighted moving average length and κ_1 controls how fast to drain the queue. To accurately compute f, we need a stable estimate of C_l, the maximum achievable throughput on link l, which is fixed and known for wired links. In this section, we discuss performance issues that can arise due to inaccuracy of C_l.

Let C_a and C_e be the actual and estimated capacities of a wireless link, respectively. Each BMCC router computes the load factor based on C_e. When $C_e < C_a$, the network goes under-utilized by a factor of $1 - C_e/C_a$. When $C_e > C_a$, the network can get overloaded, the severity of which depends on the extra capacity that is advertised $E = C_e - C_a > 0$. As E increases, the amount of traffic that cannot be handled by the network increases, leading to increased packet drops and large delays. Moreover, it also increases the chances that few unlucky flows experience more losses than others that can lead to large throughput variations and unfairness. To validate this, we study the impact of varying the capacity estimate in two scenarios: (a) download scenario, where clients are downloading files from the BS and the (b) upload scenario, where the clients are uploading via the BS.

6.2.1 Simulation Setup

All the simulations are conducted in ns2 version 2.33 using an unmodified 802.11b MAC (DCF). We use the default parameters for 802.11b in ns2 unless stated otherwise. The MAC layer rate is fixed at 11 Mbps. All the wireless nodes are assumed to be within the transmission range of

each other, a typical scenario in a WLAN. We run bulk transfer flows with a data packet size of 1500 bytes. The interface queue size of each node is set to 64 packets. Unless stated otherwise, each client runs a single flow. We also assume that all clients have similar channel conditions to the BS. All simulations are run for at least 180 s and the results are averaged over 10 runs.

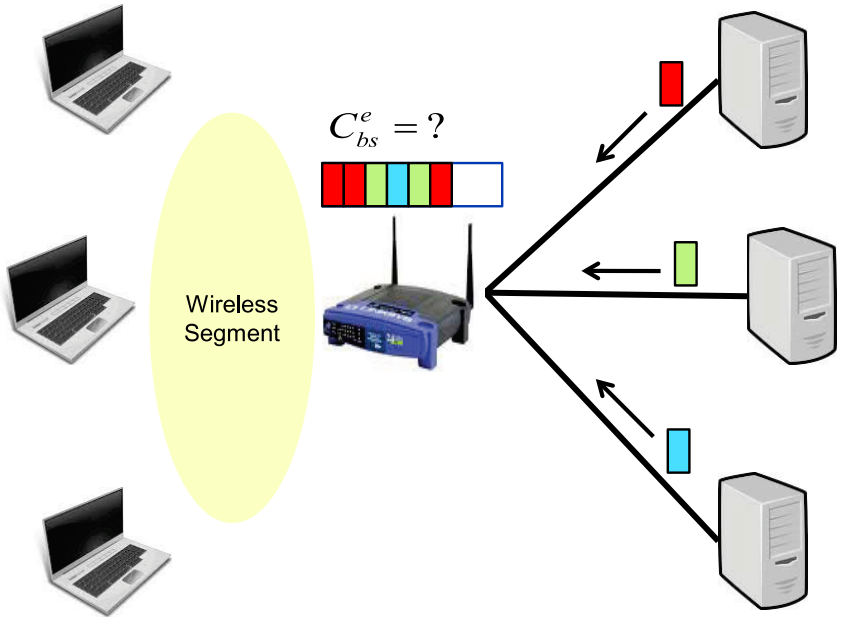
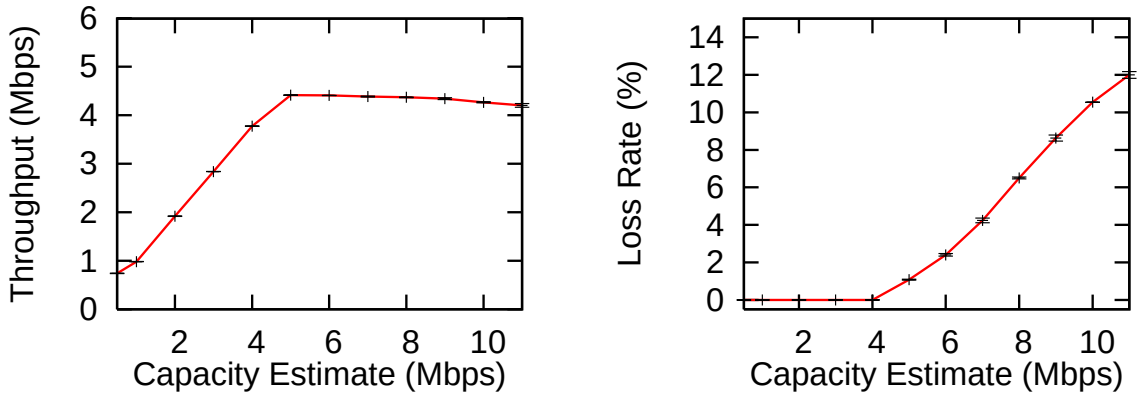


Figure 48: Download Scenario.

6.2.2 Download Case

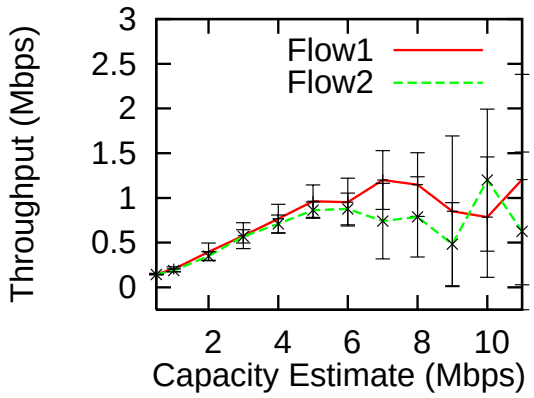
We first consider the scenario where N clients are downloading a long file via the BS as shown in Figure 48. In this case, the base station needs an estimate of the capacity to each client. When channel conditions do not differ significantly, a single estimate suffices. Note that in this scenario, the download traffic comprises of data packets whereas the upload traffic comprises of only ACKs.

6.2.2.1 Varying the Capacity Estimate We vary the capacity estimate at the BS and analyze the throughput and loss rate performance of (a) the aggregate of all flows and (b) per-flow performance.

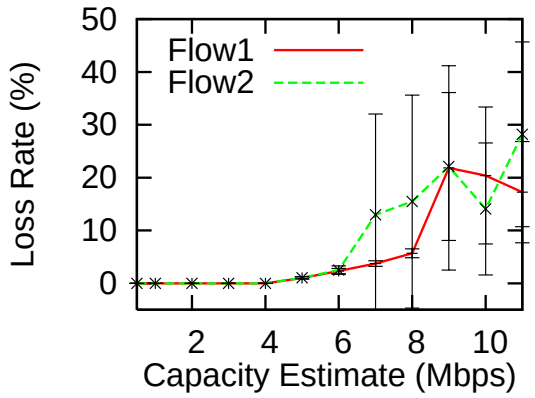


(a) Aggregate Throughput

(b) Aggregate Loss Rate



(c) Per-Flow Throughput



(d) Per-Flow Loss Rate

Figure 49: Throughput and loss rate of five BMCC flows as a function of the capacity estimate for the download scenario.

Impact on Aggregate Throughput and Loss Rate: Figures 49(a) and 49(b) show the aggregate throughput and loss rate as a function of C_e for five BMCC flows, respectively. In this scenario, according to Equation 6.3, $C_a \approx 4.5$ Mbps, after accounting for the MAC overhead and transport layer ACKs. Observe that when $C_e < 4.5$ Mbps, the aggregate loss rate is zero and aggregate throughput is equal to C_e. In these cases, the network goes under-utilized by a factor of $(1 - C_e/4.5)$. When the capacity estimate is increased beyond 4.5 Mbps, the BS starts advertising lower than actual load to the sources. This causes sources to send more data than what can be handled by the BS, resulting in packet losses. Observe that the aggregate loss rate increases roughly linearly

with increases in the capacity estimate, peaking at ≈12% for C_e=11 Mbps.

Impact on Per-flow Throughput and Loss Rate: Figures 49(c) and 49(d) show the throughput and loss rate of two (out of the five) randomly picked flows, respectively[7]. Observe that when $C_e < 4.5$ Mbps, each flow gets equal throughput, experience negligible loss rate and incur little throughput variations across the runs. However, when C_e is increased beyond 4.5 Mbps, flows starting experiencing high loss rate with large variations across the runs. This happens because as the capacity estimate increases, the BS starts advertising lower load values. First, this causes flows to send more data than what the BS can forward. Second, since not all flows are synchronized, some get to send more data than others at different points in time. The flows which send more data fill up the buffers at the BS, resulting in losses to other flows. This results in unfairness and causes large variations in the throughput of flows, which suggests that the throughput behavior is unstable.

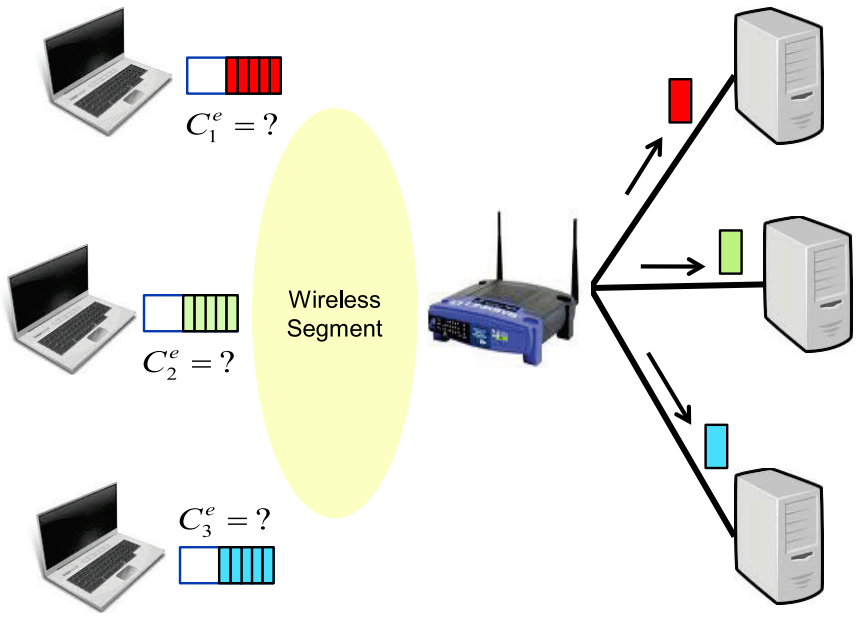
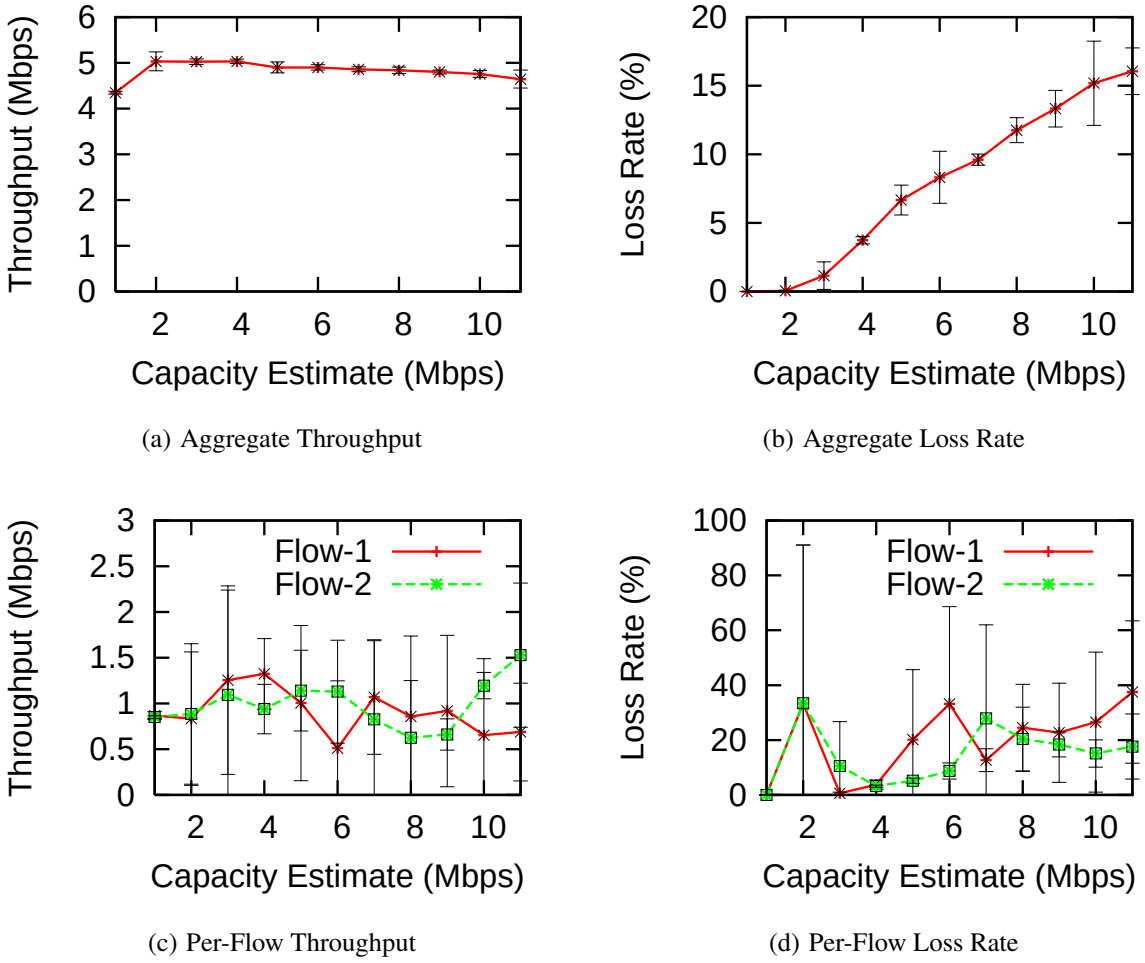


Figure 50: Upload Scenario.

[7]The throughput and loss rate for other flows were similar. We do not show them here for clarity of presentation.

6.2.3 Upload Case

We now consider a scenario where N clients are uploading long files to the Internet via the BS. As shown in Figure 50, in this case, each client needs an estimate of the available capacity to the BS. Note that in this scenario, the upload traffic comprises of data packets whereas the download traffic comprises of only ACKs.



(a) Aggregate Throughput

(b) Aggregate Loss Rate

(c) Per-Flow Throughput

(d) Per-Flow Loss Rate

Figure 51: Throughput and loss rate performance of five long-lived BMCC flows as a function of the capacity estimate for the Upload scenario.

6.2.3.1 Varying the Capacity Estimate We now vary the capacity estimate of each client-BS pair and analyze the throughput and loss rate performance of (a) the aggregate of all clients and (b)

per-client performance.

Impact on Aggregate Throughput and Loss Rate: Figures 51(a) and 51(b) show the aggregate throughput and loss rate as a function of C_e for five BMCC flows, respectively. In this scenario, C_a for each client is equal to ≈1 Mbps. Therefore, as the capacity estimate increases beyond 1 Mbps, the aggregate loss rate starts increasing. When the capacity estimate of each client is 11 Mbps, the loss rate is ≈16%. Observe that the aggregate throughput in the upload case is slightly higher than in the download case. The reason is that when clients are uploading files, the transport-layer ACKs to each client are sent only by the BS. However, in the download case, each client needs to compete for the channel to send these ACKs. This wastes bandwidth and therefore reduces the aggregate throughput for the download traffic.

Impact on Per-flow Throughput and Loss Rate: Figures 51(c) and 51(d) show the throughput and loss rate of two (out of the five) randomly picked flows, respectively. Observe that flows incur large throughput and loss rate variations across a range of capacity estimates. Note that some flows experience loss rates that are as high as 35%. When C_e=1 Mbps for both the clients, each achieve the same throughput and experience negligible loss rates.

Based on the above results, we conclude that each node in a WLAN needs to dynamically estimate the available capacity in order to achieve good end-to-end performance.

6.3 ESTIMATING AVAILABLE CAPACITY

One possible way to estimate the available capacity is to make each wireless node measure the average values of p_l, $A_{S\backslash l}$, R_l, T^{BO}, and X over a given interval T and then use Equation 6.3 to compute the available capacity. However, this involves keeping track of several variables. Another approach is to use the average transmission time of successful network layer packets in a given interval T for this purpose. This accounts for all possible delays and requires the keeping of only the average transmission time of packets. Therefore, we employ the latter approach.

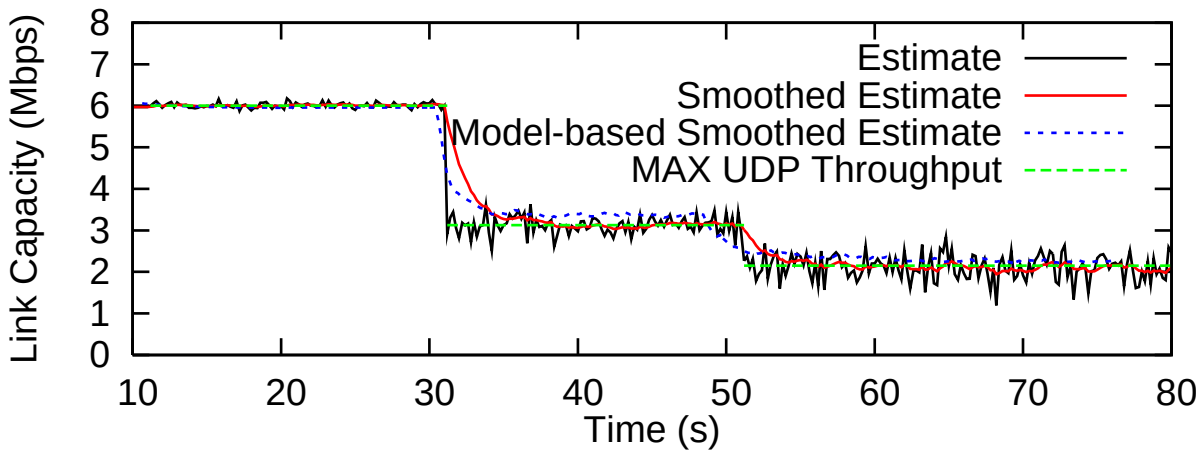


Figure 52: Maximum UDP throughput for three upload transfers as a function of time.

6.3.1 Using Packet Transmission Times for Available Capacity Estimation

The transmission time of a network layer packet is the difference between t_s, the time when a packet is released to the MAC and t_e, when the corresponding MAC layer ACK is received. Note that $t_s - t_e$ includes MAC overhead delays, deferrals due to channel contention, retransmissions due to packet losses, the transmission delay, and the propagation delay. Given a packet of size X bits, the network-layer transmission rate is equal to $X/(t_e - t_s)$. In order to reduce the burstiness in the rate estimates, we measure the available capacity over an interval t_p =200 ms and then use exponential averaging as follows:

$$C_e^{av} = a \cdot C_e^{av} + (1 - a) \cdot C_e^{sample}$$

where $a = 0.875$. To assess the efficacy of using the average transmission rate for approximating the available capacity, we ran ns2 simulations. We consider three clients that are uploading data over UDP such that each client is able to saturate the path. These clients start their transfers with an inter-arrival time of 20 s. Figure 52 shows the Maximum UDP throughput, the estimated available capacity based on transmission rates, an exponentially weighted moving average of the capacity

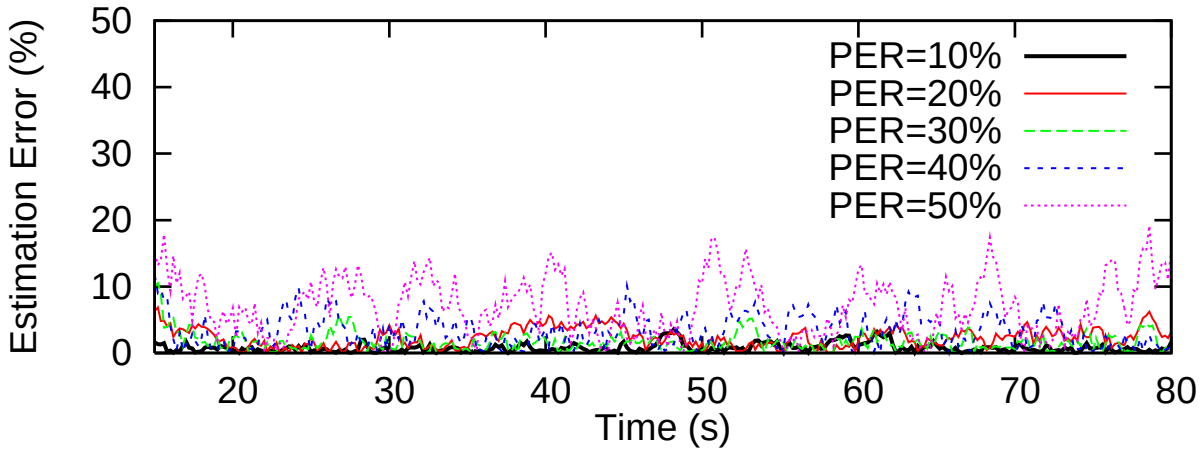


Figure 53: Capacity estimation error as a function of time for different PERs.

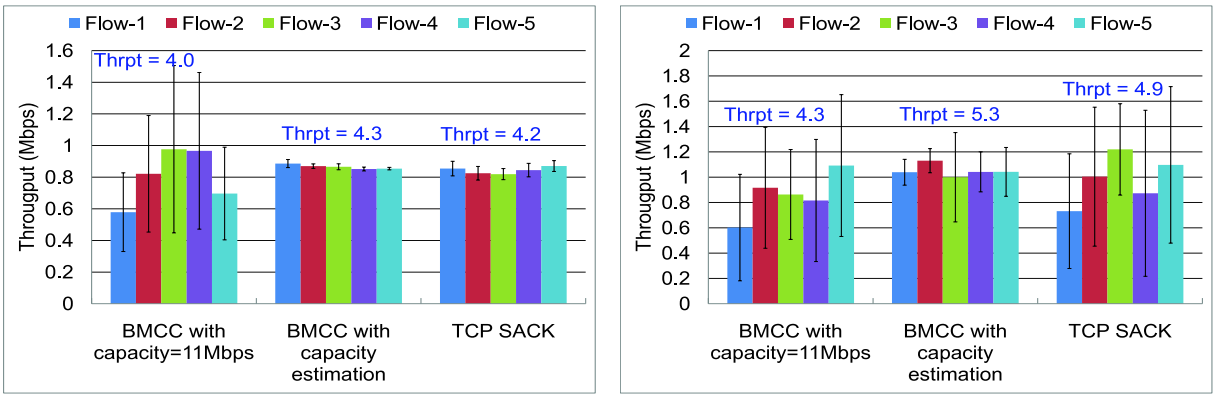
estimates, and the estimated available capacity based on Equation 6.3 for the three clients as a function of time[8]. Observe that as new clients arrive, the average transmission rate is able to track the available capacity quite accurately. Also note that the model-based estimates are very similar to the transmission rate estimates.

6.3.2 Impact of Channel Losses

Channel losses often occur in a wireless medium due to signal attenuation, fading, etc. Therefore, it is important to study the impact of channel losses on capacity estimation. Assuming an error model that introduces packet errors uniformly at random, we vary the Packet Error Rate (PER) from 10% to 50% and analyze the capacity estimation error[9]. Figure 53 shows the estimation error as a function of time for different PERs with UDP traffic. Observe that for PERs less than 30%, the average estimation error remains below 5%. As the PER increases, the average estimation error also increases, reaching to an average value of ≈10% (and a maximum value of 20%) for PER=50%. This happens because as the PER increases, packets start experiencing burst losses

[8]Note that in the model-based approach, we measure the average values of each quantity as well as the free air time to determine the available capacity.

[9]=$|C_e - C_a|/C_a$



(a) Download scenario

(b) Upload scenario

Figure 54: Throughput of five download and upload transfers for BMCC (with and without capacity estimation) and TCP SACK.

that result in successive retransmissions at the MAC layer and an increase in the average backoff period. This increases the variability in packet transmission times, resulting in larger estimation errors.

6.3.3 Handling Heterogenous Packet Sizes

The available capacity also depends on the average packet size as shown in Equation 6.3. Packets with different sizes generally incur the same MAC overhead. Therefore, large packets can achieve a higher maximum throughput than small packets. In a typical WLAN, applications may employ different packets sizes. To account for this heterogeneity, each wireless node maintains the average of all packet sizes seen in a moving window. The average packet size is then used to determine the available capacity by normalizing the actual transmission rates to the average packet size. Observe that the transmission rate of two packets only differ in their times to transmit a packet at the link layer (including retransmissions). Suppose the actual packet size, the average packet size, and the transmission rate are equal to X, X_{av}, and T_X, respectively. Then the normalized transmission rate equals $X_{av}/(T_X - X_{av}/R_l + X/R_l)$.

6.4 EVALUATION

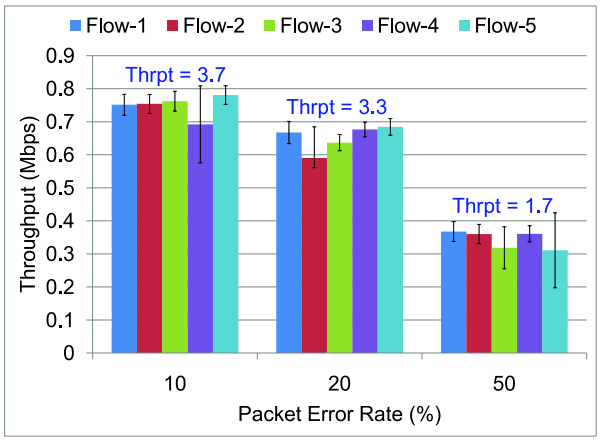
In this section, we evaluate the performance of BMCC by allowing each wireless node to estimate the load factor by using the average packet transmission rates as estimates for capacity. We study the impact of channel losses, number of clients, and MAC bitrates on the performance of BMCC for both the download and upload cases. We also compare BMCC's performance with TCP SACK.

6.4.1 Performance with Zero Channel Losses

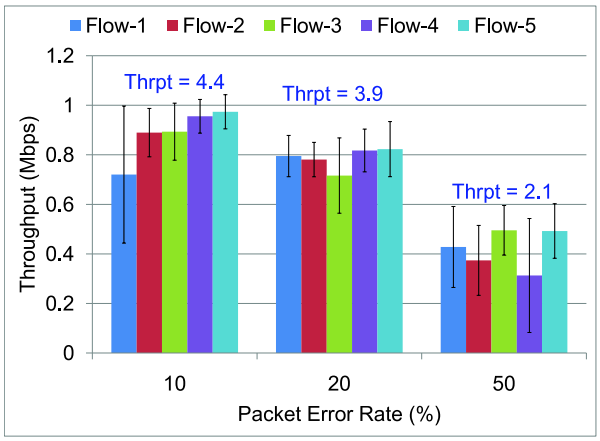
We now analyze the performance of BMCC in the presence and absence of capacity estimation for the download and upload scenarios while considering an error-free channel. We also compare the resulting performance with TCP SACK. Figures 54(a) and 54(b) show the throughput of five BMCC and five TCP SACK flows (one flow per client). Observe that when the capacity estimate is 11 Mbps, BMCC flows achieve unequal throughput and incur large variations in them. However, when capacity estimation is used, each wireless node is able to convey accurate load factor values to end-hosts. This causes BMCC flows to achieve a fairer throughput distribution in both the scenarios. Compared to TCP SACK, BMCC flows achieve higher throughout and incur smaller throughput variations. In particular, BMCC achieves $\approx$400 kbps higher throughput than TCP SACK in the upload case. This happens because BMCC avoids packet losses by using accurate load factor estimates whereas TCP SACK introduces packet drops by overflowing buffers which reduces its throughput.

6.4.2 Impact of Channel Losses

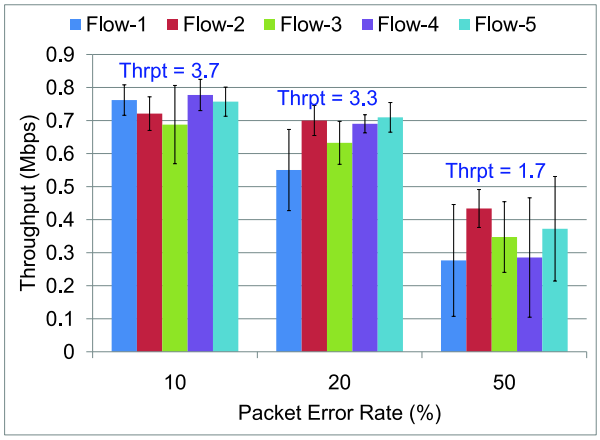
We now vary the PER and evaluate the performance of BMCC and TCP SACK in the two scenarios. By introducing channel losses, each network layer packet may now require retransmissions at the MAC layer. This leads to variations in packet transmission times and thus in the measured available capacity. Figure 55 shows the throughput achieved by download and upload transfers as a function of PER. In the download scenario, BMCC flows achieve similar throughputs across a range of PERs. While BMCC and TCP SACK flows achieve the same aggregate throughput, BMCC flows incur smaller throughput variations. On the other hand, in the upload scenario,



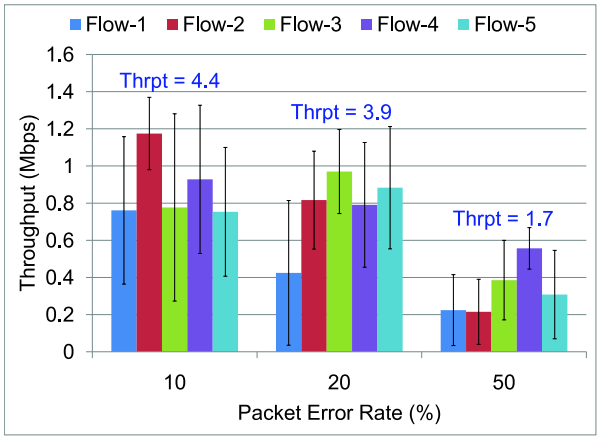
(a) BMCC under the download scenario



(b) BMCC under the upload scenario



(c) TCP SACK under the download scenario

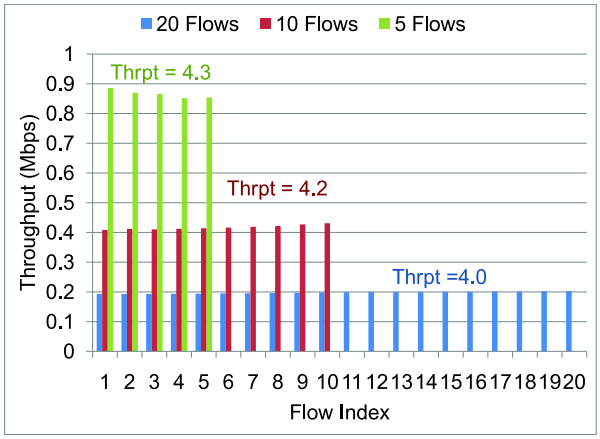


(d) TCP SACK under the upload scenario

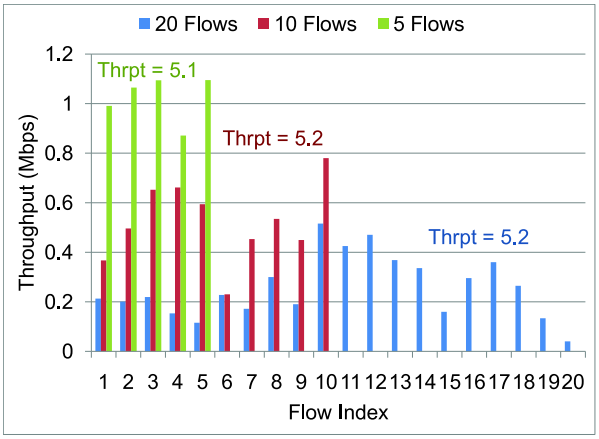
Figure 55: Comparison of throughput of five BMCC and TCP SACK flows under the upload and download scenarios for different channel loss rates (with nodes using capacity estimation in case of BMCC).

BMCC flows achieve a fairer throughput distribution than SACK. While they achieve the same aggregate throughput for PERs of 10% and 20%, however, when PER is 50%, BMCC flows achieve higher throughput. This happens because BMCC is more efficient at ramping up after a packet loss. Observe that in the upload scenario, the variations are much larger compared to the download case. This difference is due to the fact that in the download case, packets from all the flows serve as samples for estimating the BS-to-client(s) capacity, which is the same across all clients. At high

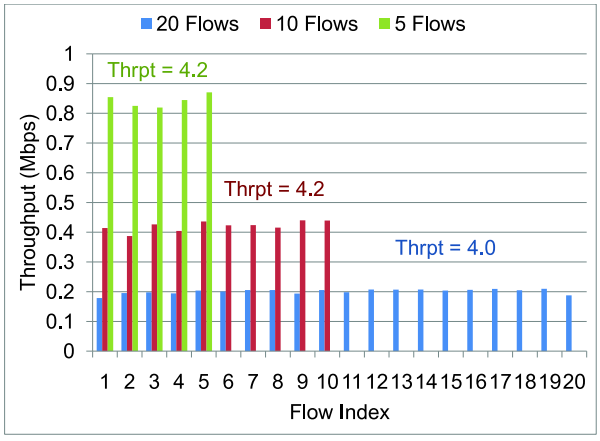
PERs, packets experience burst losses, which increases the transmission time of successful packets as well the variability in it. However, since the number of samples are large in the download case, capacity estimation is quite accurate. On the other hand, in the upload scenario, each wireless node has to estimate the capacity individually and, with burst losses, they are unable to accurately determine the channel loss rate as well as channel contention due to fewer packet transmissions in a given interval. This introduces variations in the available capacity estimate which leads to throughput variations.



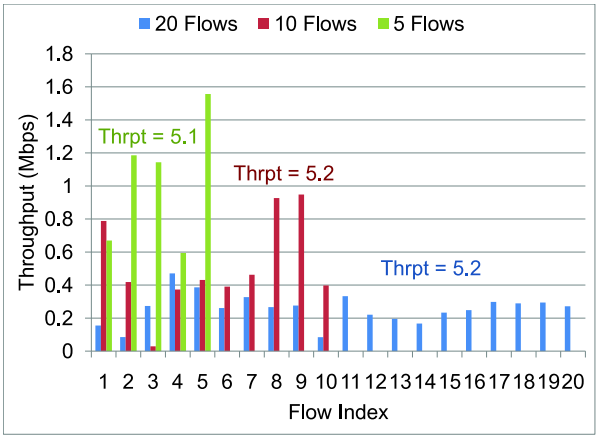
(a) BMCC under the download scenario



(b) BMCC under the upload scenario



(c) TCP SACK under the download scenario



(d) TCP SACK under the upload scenario

Figure 56: Throughput of download and upload transfers as a function of the number of clients.

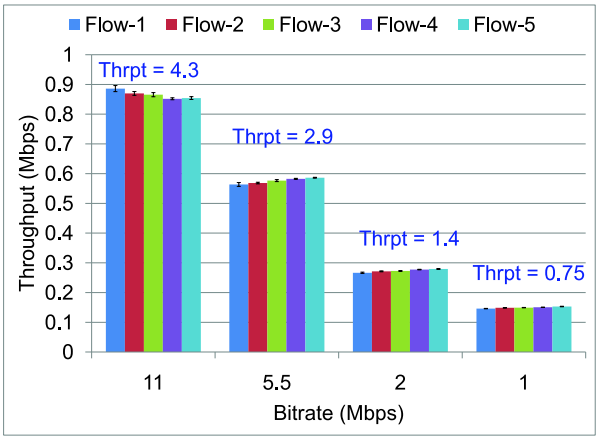
6.4.3 Impact of the Number of Clients

We now study the impact of increasing the number of clients that download or upload files. Increasing the number of clients increases the amount of wireless capacity needed for transmitting transport layer ACKs, which reduces the available capacity for data packets thus the number of successfully transmitted packets in a given interval.

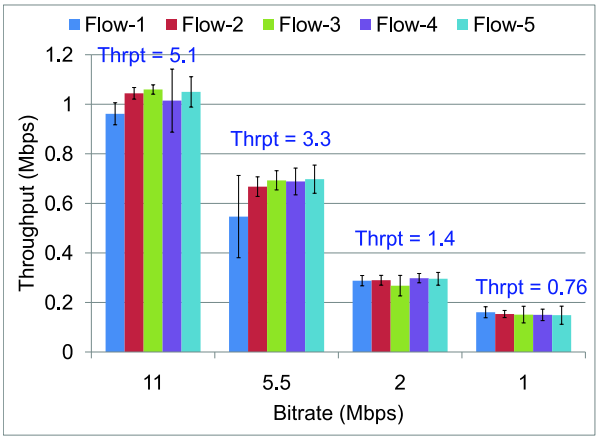
Figure 56 shows the throughput of 5, 10, and 20 clients for the two scenarios. Observe that in the download scenario, BMCC as well as TCP SACK flows achieve the same throughput across different number of clients. However, the throughput distribution is more variable in case of the upload scenario, especially when there 10 and 20 clients in the WLAN. This is so because in the upload scenario, each client needs to estimate the *channel contention* to be able to accurate estimate the available capacity. As the number of clients increases, each client is able to send fewer successful packets in the measurement interval. This increases the inaccuracy in estimates and also introduces the variability in the throughput of flows This can be improved by increasing the length of the measurement interval at the cost of reduced responsiveness.

6.4.4 Impact of MAC Bitrate

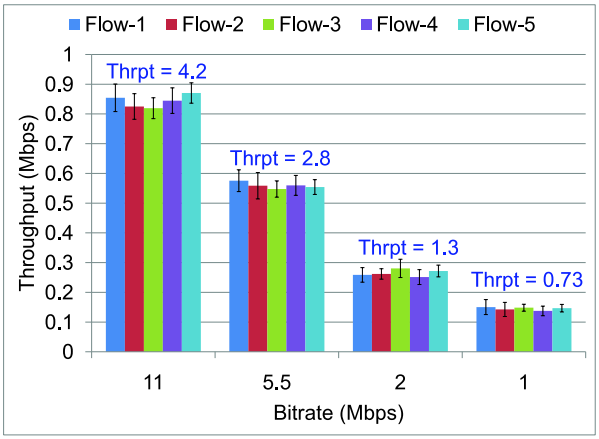
Packets sent at higher bitrates require higher Signal-to-Noise Ratio (SNR) for them to be decoded by the receiver. Therefore, higher bitrates can be sustained for shorter distances compared to lower bitrates. To adjust to different channel conditions, 802.11 cards can use different bitrates. We now analyze the impact of MAC bitrate on the performance of BMCC and SACK flows. Lowering the bitrate increases the transmission time of packets which in turn decreases the number of packets that can be successfully transmitted in a given interval. This can potentially impact capacity estimation in case of BMCC. Moreover, it can reduce the BDP of the path, which, in case of TCP SACK, is likely to result in more packet drops in a given interval of time. At small window sizes, this can lead to more timeouts. Figure 57 shows the throughput of five BMCC and SACK flows as a function of the MAC bitrate for the two scenarios. Observe that as we reduce the bitrate, the average throughput decreases, however, BMCC flows achieve higher throughput than SACK flows. Moreover, BMCC flows incur smaller throughput variations. In the upload scenario, BMCC



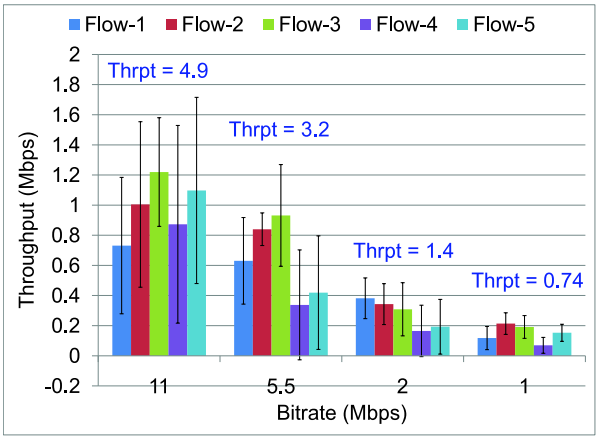
(a) BMCC under the download scenario



(b) BMCC under the upload scenario



(c) TCP SACK under the download scenario



(d) TCP SACK under the upload scenario

Figure 57: Throughput of five download and upload transfers with different MAC bitrates.

achieves a more fair throughput distribution than SACK. SACK flows incur large variations in their throughput across all the bitrates.

6.5 DISCUSSION

In this section we discuss various aspects of this work and suggest possible extensions.

Per-Client Queues at the Base Station: We considered cases where the channel conditions between the BS and the clients are similar. However, there can be cases where the BS-client channel conditions are different resulting in different channel loss rates. This would require the BS to maintain one queue per-client because the available capacity of each BS-client link would be different. In this case, the capacity of each link can be estimated by measuring the average transmission rate of only those packets that are transmitted on that link (or served by the corresponding queue). Note that the BS may be able to maintain one queue for all clients, however, if the channel conditions differ significantly then the performance would be determined by the weakest link which will cause all clients to experience degraded performance.

Passive Approach to Capacity Estimation: Using the average transmission rate as an estimate of available capacity requires packets to be sent before an estimate becomes available. Hence, at the beginning of a transfer flows have to start with a possibly inaccurate estimate. However, as flows start sending packets they will be able to estimate the capacity better. This is an *active* approach because flows' packets also serve as probe packets for capacity estimation Another possibility is to follow a more *passive* approach. This derives from Equation 6.3, in which each node monitors the channel around the transmitter and uses the air free time to guide the capacity estimate. In this approach, however, clients still need an estimate of the channel loss rate, p_l. If the channel loss rates are stable, they can be measured from previous transfers' packets and then Equation 6.3 can provide good estimates without solely relying on flows' packets. If the channel loss rates vary frequently (for instance when the clients are mobile) then the loss rate would need to be measured either through flows' packets or through active probing which consumes wireless bandwidth. One possible limitation of this approach is that it will overestimate capacity in the presence of hidden terminals. The reason is that hidden terminals cannot be overhead by the transmitter but they cause packet collisions at the receiver, which reduces the effective capacity.

Extension to Multi-hop Wireless Mesh Networks: This work focussed only on WLANs, however, the presented approach has broader applicability. A similar approach can be used in multi-hop wireless mesh networks where each access point maintains the average transmission rate to every other access point and uses these estimates to provide feedback to end-hosts.

Performance with Rate Adaptation Algorithms: For performance evaluation, we only consid-

ered fixed MAC bitrates. However, most 802.11 cards today employ some form of rate adaptation to adjust to different channel conditions. It would be interesting to see how dynamics variations in the MAC rate impact the available capacity estimates.

6.6 RELATED WORK

The idea of using the measured throughput of network layer packets for bandwidth estimation has been used in earlier works in a variety contexts [122, 90, 123]. [122] uses it to improve the performance of adaptive multimedia applications in AODV MANETs whereas [123] uses it in an admission control and dynamic bandwidth management scheme for WLANs. However, in our work, the focus is to use these estimates to compute the load factor for congestion control purposes. Most recently, [90] uses the average throughput of UDP packets along with a binary interference model to estimate the capacity of a static multi-hop wireless mesh networks. There are three differences between [90] and our work. First, they compute the available capacity by estimating the channel loss rate rather than directly measuring the transmission rate of network layer packets. Second, they use separate probe packets for channel loss estimation. We use flows' packets for capacity estimation because estimates are useful when new flows arrive. As more packets are sent, flows are able to accurately determine their available capacity. Lastly, they focus on multi-hop wireless mesh networks and their goal is to apply long term source rate limits whereas our focus is to use these estimates for congestion control that operate on RTT timescales.

Sundaresan et. al. uses a combination of the average queueing delay and the transmission delay experienced by nodes in an Ad hoc network to guide the source rates [32]. Bianchi [119] and others [124, 90] propose 802.11 models for characterizing the maximum throughput of stations in 802.11-based wireless networks. Our work can be viewed as a practical implementation strategy for these models. WCPCap [80] uses the 802.11 model presented in [125] to compute the wireless neighborhood capacity in multi-hop wireless mesh network. For use in multi-hop wireless mesh networks, a similar approach can also be used in case of BMCC.

6.7 SUMMARY

In this chapter, we presented a simple model to study the key factors that affect available capacity of wireless nodes in a 802.11-based WLAN. We then analyzed the impact of inaccurate capacity estimates on the end-to-end performance of BMCC when a wireless segment becomes the bottleneck. We showed that inaccurate capacity estimates can either result in under-utilization when the actual capacity is higher or overload and unfairness when the actual capacity is lower. To address these issues, we proposed the use of the average transmission rate experienced network layer packets (excluding the idle time) as estimates for available capacity. We evaluated the performance of BMCC using extensive packet-level simulations and showed that the mechanism works well across a range of scenarios.

7.0 CONCLUSION AND FUTURE WORK

In this chapter, we summarize our contributions and propose possible future directions. We conclude this dissertation by offering final remarks.

7.1 CONTRIBUTIONS

This dissertation makes four contributions. We summarize each of these contributions in turn.

First, we presented a systematic analysis of the interplay between performance and feedback. We showed that in order to achieve near-optimal convergence to an efficient bandwidth allocation, 3-bit load feedback is sufficient. When more bits are used, performance follows the law of diminishing returns. Further, we showed that multiple backoff factors improve responsiveness as well as fairness during times of high statistical multiplexing. Recent studies on data centers reinforce the importance of this property [9], whereby problems such as TCP incast [11] can be avoided if sources can react more aggressively during times of high congestion. This property also allows for smoother throughput variations during times of low congestion, a desirable property for real-time applications [23]. We showed that in order to achieve the right balance between responsiveness, fairness, and throughput variations, it suffices to use 3 bits to represent the overload region. Based on the insights gained, we designed the Multi-Level Feedback Congestion control Protocol (MLCP).

Second, we presented the design, analysis, and evaluation of an efficient framework for congestion control called Binary Marking Congestion Control (BMCC) that uses the existing ECN bits to achieve high performance. With BMCC, long flows are able to achieve high utilization and

fairness on large BDP networks, whereas short flows finish much quicker than with TCP. This happens because BMCC flows maintain close to zero queueing delay. We presented analytical models of convergence to fairness and efficient bandwidth allocations and studied the impact of using a packet marking scheme on the resulting protocol. Using extensive packet-level simulations, we evaluated BMCC and showed that it outperforms several other schemes.

Third, we addressed deployment issues of BMCC and presented a complete deployment path of the proposed mechanism. In particular, we analyzed the performance of BMCC over different kinds of bottleneck routers and studied its bandwidth sharing properties with TCP. We showed that BMCC flows do not perform well when they run over non-BMCC bottleneck routers. In this case, it is easy to detect non-BMCC bottlenecks and shift to SACK mode. Further, we showed that when TCP and BMCC share a non-BMCC bottleneck, TCP flows can get starved, whereas over BMCC bottlenecks, BMCC can get starved. To address these issues, we presented simple algorithms that enable fairer bandwidth sharing between BMCC and TCP flows. The proposed solutions have applicability beyond BMCC.

Finally, we considered 802.11-based WLANs and presented a detailed treatment of the issues involved. In particular, we showed that inaccurate capacity estimates can lead to either underutilization or overload and unfairness between flows. Using a simple model, we showed that available capacity estimation methods need to account for a variety of factors including variations in bitrate, channel loss rate, backoff factors, packet sizes, and the air time consumed by other nodes. We proposed to use the average transmission rate of network layer packets as an approximation of the available capacity. Through extensive evaluation we showed that it results in good performance.

7.2 FUTURE WORK

7.2.1 Adjusting the AI factor based on Load

In BMCC, we do not adapt α or the AI factor based on load, even though the BMCC framework provides high accuracy estimates of f during high load. This was a design choice we made in order to allow for fairer bandwidth sharing between TCP and BMCC. However, we believe it

would useful to determine if there exists an optimal adaptation of α based on load. Using larger values of α when $f \ll 100\%$ can lead to improved convergence on high BDP links. However, care must be taken to ensure that performance does not get worse on small BDP links.

7.2.2 Non-linear Mapping of Load Values

In the design of BMCC, we only considered a linear mapping of load factor values to the $[0, 1]$ interval. Specifically, the mapping from load factor f to $[0, 1]$ interpolates linearly between the points (f, c) = $(0, 0)$, $(0.15, 0)$, $(0.75, 0.25)$, $(1, 0.5)$, $(1.2, 1)$, $(1, 1)$. It would be interesting to see how non-linear mapping would impact performance and to determine if there exists an optimal mapping function that optimizes the convergence properties of the protocol.

7.2.3 Using the Rate of Change in Load to Estimate the Available Capacity

We used load as a congestion signal because (a) it accurately characterizes congestion and therefore allows excellent performance to be achieved and (b) it is a scale-free parameter that can be encoded using few bits, which is particularly helpful for deployment purposes. Raw load factor values, however, hide the absolute bottleneck capacities from the senders. Protocols can use the rate of change in load factor values to estimate the available capacity and further improve performance. Using this, instead of directly using available capacity as a congestion signal helps in retaining the scale-free nature of the signal while achieving the benefits of the other.

7.2.4 Real Implementation

As part of the future work, it would useful to evaluate the performance of BMCC using a real implementation. BMCC end-host functionality can be implemented as a loadable kernel module. New congestion control protocols are often implemented as kernel modules to avoid the need to patch the kernel and the subsequent kernel recompilation. The BMCC router functionality can be implemented in Linux, NetFPGAs, etc. The data plane operations of the router include updating a byte counter, comparing the link load with the value in the IPID field, and based on this, setting of the ECN bits in the IP header. The control plane operations take place at a much larger timescale.

Specifically, link utilization is computed every 200 ms and the average queue length is measured every 10 ms. Load factor (computed every 200 ms) is computed as a weighted average of these two values. We believe testbeds like GENI [126] provide the scale, heterogeneity, and realism needed for extensive evaluation of BMCC.

7.2.5 Extension to other Wireless Networks

We used the available capacity method (see Chapter 6) in 802.11-based WLANs. However, we believe it can also perform well over multi-hop wireless mesh networks. In this case, each mesh node can maintain a single queue or a multiple queues corresponding to each possible destination. Available capacity can then be estimate for each queue using the average service rate experienced by packets.

7.3 FINAL REMARKS

In this dissertation, we presented the design, analysis, and evaluation of an efficient framework of congestion control for next-generation networked called Binary Marking Congestion Control (BMCC). BMCC uses aggregate load as a congestion signal and achieves efficient and fair bandwidth allocations on high bandwidth-delay paths while maintaining low persistent queue length and negligible loss rates. Moreover, it considerably reduces the average flow completions times. We showed that BMCC is particularly amenable to deployment as it uses the existing ECN bits to achieve high performance. We presented models for characterizing the performance of BMCC and performed extensive simulation evaluation with several other proposals. We presented a complete deployment path and showed that BMCC can incrementally deployed on the Internet. Finally, we presented methods for available capacity estimation in 802.11 WLANs and showed that using them enables BMCC to perform well under diverse network settings. We believe BMCC is a practical congestion control protocol that can be readily deployed. Moreover, because its signalling is compatible with the IP Header, its deployment can immediately begin in specialized networks such as data centers, satellite networks, storage area networks, etc.

8.0 BMCC IMPLEMENTATION

A BMCC router maintains three variables: λ_l, a byte counter that is updated when a packet arrives, q_{av}, an exponentially weighted moving average of the queue length that is updated every 10 ms, and f, the load factor that is computed every 200 ms. We describe how each of these variables are updated:

On Packet Arrival:

$\lambda = \lambda + X$

where X is the packet size.

On Packet Departure:

if (P_{ECN}=$(10)_2$ and $H(f) > P_{IPID}$)

then P_{ECN}=$(01)_2$

where P_{ECN} and P_{IPID} are the ECN and IPID fields of the IP Header of the packet to be dequeued and $H(f) \in [0, 1]$ is the mapping of the load factor f into the $[0, 1]$ interval.

When the Queue Sampling Timer Expires:

$q_{sum} = q_{sum} + q_{length}$

q_{times} ++

timer.reschedule(t_q)

where q_{sum} is the sum of the bytes in the queue until now, q_{times} is the number of times the queue length has been counted in q_{sum}. These variables are updated every t_q s, (the queueing sampling period),

---

When the Load Factor Measurement Timer Expires:

q_{sample} = $\frac{q_{sum}}{q_{times}}$

q_{av} = $\alpha \cdot q_{av}$ + $(1 - \alpha) \cdot q_{sample}$

f = $\frac{\lambda + \kappa_1 q_{av}}{\gamma C_l t_p}$

λ_l = 0

q_{sum} = 0

q_{times} = 0

timer.reschedule(t_p)

where λ is the amount of traffic received during t_p, C_l is the capacity of the link and $\gamma_l \leq 1$ is the target utilization, κ_1 controls how fast to drain the queue, and $\alpha = 0.875$.

BIBLIOGRAPHY

[1] J. Postel. Transmission Control Protocol. RFC 793, 1981.

[2] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM*, Aug 1988.

[3] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581, April 1999.

[4] C. Partridge and T. Shepard. Tcp/ip performance over satellite links. In *IEEE Network Magazine*, pages 44–49, 1997.

[5] M. Allman, D. Glover, and L. Sanchez. Enhancing TCP over satellite channels using standard mehanisms. RFC 2488, January 1999.

[6] D. Katabi, M. Handley, and C. Rohrs. Internet Congestion Control for High Bandwidth-Delay Product Networks. In *ACM SIGCOMM*, Aug 2002.

[7] Yee-Ting Li, Douglas Leith, and Robert N. Shorten. Experimental evaluation of TCP protocols for high-speed networks. *IEEE/ACM Trans. Netw.*, 15(5):1109–1122, 2007.

[8] Nandita Dukkipati, Masayoshi Kobayashi, Rui Zhang-Shen, and Nick McKeown. Processor Sharing Flows in the Internet. In *IWQoS*, Jun 2005.

[9] Mohammad Alizadeh, Albert Greenberg, Dave Maltz, Jitu Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. DCTCP: Efficient Packet Transport for the Commoditized Data Center. In *ACM SIGCOMM*, 2010.

[10] Ihsan Ayyub Qazi, Lachlan L. H. Andrew, and Taieb Znati. Congestion Control using Efficient Explicit Feedback. In *IEEE INFOCOM*, Apr 2009.

[11] Vijay Vasudevan, Amar Phanishayee, Hiral Shah, Elie Krevat, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Brian Mueller. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *Proc. ACM SIGCOMM*, Barcelona, Spain, August 2009.

[12] Optical carrier transmission rates. http://en.wikipedia.org/wiki/Optical_Carrier_transmission_rates.

[13] R. Hui, B. Zhu, R. Huang, C. Allen, K. Demarest, Senior Member, Senior Member, Senior Member, and D. Richards. 10-Gb/s SCM Fiber System Using Optical SSB Modulation. *IEEE Photonics Technology Letters*, 13, 2001.

[14] Vern Paxson. End-to-End Internet Packet Dynamics. In *ACM SIGCOMM*, Sep 1997.

[15] 14 Tbps over a Single Optical Fiber: Successful Demonstration of World's Largest Capacity. http://www.ntt.co.jp/news/news06e/0609/060929a.html.

[16] IEEE 802.11. http://en.wikipedia.org/wiki/IEEE_802.11.

[17] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 31–42, New York, NY, USA, 2005. ACM.

[18] 10 Gigabit Ethernet. http://en.wikipedia.org/wiki/10_Gigabit_Ethernet.

[19] Aruna Balasubramanian ane Ratul Mahajan and Arun Vankataramani. Augmenting Mobile 3G Using WiFi. In *MobiSys*, 2010.

[20] C. Partridge. Isochronous Applications Do Not Require Jitter-Controlled Networks. RFC 1257, 1991.

[21] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, Oliver Spatscheck, and Xiaodong Zhang. Delving into internet streaming media delivery: a quality and resource utilization perspective. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 217–230, New York, NY, USA, 2006. ACM.

[22] Eli Brosh, Salman Abdul Baset, Dan Rubenstein, and Henning Schulzrinne. The delay-friendliness of tcp. In *SIGMETRICS '08: Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 49–60, New York, NY, USA, 2008. ACM.

[23] W.-T. Tan and A. Zakhor. Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol. In *IEEE Trans. on Multimedia*, Jun 1999.

[24] Deepak Bansal and Hari Balakrishnan. Binomial Congestion Control Algorithms. In *IEEE Infocom 2001*, Anchorage, AK, April 2001.

[25] T. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. In *IEEE/ACM Trans. Networking, 5(3):336-350*, Jun 1997.

[26] Self-Organizing Neighborhood Wireless Mesh Networks. http://research.microsoft.com/mesh/.

[27] Roofnet. http://pdos.csail.mit.edu/roofnet/.

[28] Ye Tian, Kai Xu, and Nirwan Ansari. TCP in wireless environments: Problems and solutions. *IEEE Communications Magazine*, 43:27–32, 2005.

[29] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP performance over wireless networks. In *MobiCom*, 1995.

[30] Srinivasan Seshan, Hari Balakrishnan, and Y H. Katz. Handoffs in cellular wireless networks: The Daedalus implementation and experience. *Kluwer Journal on Wireless Personal Communications*, 4:141–162, 1997.

[31] Sangtae Ha and Injong Rhee. Hybrid Slow Start for High-Bandwidth and Long-Distance Networks. In *PFLDnet*, 2008.

[32] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: A reliable transport protocol for ad hoc networks. ieee. *In IEEE Transactions on Mobile Computing*, 2005.

[33] ITU's Asia-Pacific Telecommunication and ICT Indicators Report focuses on broadband connectivity: Too much or too little? http://www.itu.int/newsroom/press_releases/2008/25.html.

[34] Rayadurgam Srikant. *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*. SpringerVerlag, 2004.

[35] Dina Katabi. *Decoupling Congestion Control and Bandwidth Allocation Policy With Application to High Bandwidth-Delay Product Networks*. PhD thesis, Department of Electrical Engineering and Computer Science, 2003.

[36] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, December 2003.

[37] C. Jin, D. Wei, and S. Low. FAST TCP: Motivation, Architecture, Algorithms and Performance. In *IEEE INFOCOM*, Mar 2004.

[38] I. Rhee and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. In *PFLDNet'05*, February 2005.

[39] H. Bullot and R. Les Cottrell. Evaluation of advanced TCP stacks on fast long-distance production networks. http://www.slac.stanford.edu/grp/scs/net/talk03/tcp-slac-nov03.pdf.

[40] A. Falk, D. Katabi, and Y. Pryadkin. Specification for the explicit control protocol (xcp). In *draft-falk-xcp-03.txt*, 2007.

[41] Nedeljko Vasic, Srinidhi Kuntimaddi, and Dejan Kostic. One Bit Is Enough: a Framework for Deploying Explicit Feedback Congestion Control Protocols. In *Proceedings of The First International Conference on COMmunication Systems and NETworkS (COMSNETS)*, 2009.

[42] Xiaolong Li and Homayoun Yousefi'zadeh. MPCP: multi packet congestion-control protocol. *SIGCOMM Comput. Commun. Rev.*, 39(5):5–11, 2009.

[43] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Mdard, and Jon Crowcroft. XORs in The Air: Practical Wireless Network Coding. In *ACM SIGCOMM*, 2006.

[44] Jon C. R. Bennett, Craig Partridge, and Nicholas Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Trans. Netw.*, 7(6):789–798, 1999.

[45] D-M. Chiu and R. Jain. Analysis of Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. In *Computer Networks and ISDN Systems*, Jun 1989.

[46] J. Kulik, R. Coulter, D. Rockwell, and C. Partridge. Paced tcp for high delay-bandwidth networks. In *IEEE Workshop on Satellite Based Information Systems*, 1999.

[47] S. Bhandarkar, S. Jain, and A. Reddy. Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control. In *PFLDNet*, Feb 2005.

[48] M. Mathis, J. Semke, and J. Mahdavi. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review*, 27(3), 1997.

[49] Rajesh Krishnan, James P. G. Sterbenz, Wesley M. Eddy, Craig Partridge, and Mark Allman. Explicit transport error notification (eten) for error-prone wireless and satellite networks. *Comput. Netw.*, 46(3):343–362, 2004.

[50] T. V. Lakshman, U. Madhow, and Bernhard Suter. TCP/IP performance with random loss and bidirectional congestion. *IEEE/ACM Transactions on Networking*, 8:541–555, 2000.

[51] L. Brakmo and L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. In *IEEE J. Selected Areas in Communications*, Oct 1995.

[52] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. In *IEEE INFOCOM*, Mar 2004.

[53] Thomas Anderson, Andy Collins, Arvind Krishnamurthy, and John Zahorjan. PCP: Efficient Endpoint Congestion Control. In *NSDI'06*, 2006.

[54] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A Compound TCP Approach for High-Speed and Long Distance Networks. In *IEEE INFOCOM*, Apr 2006.

[55] K. K. Ramakrishnan and S. Floyd. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, September 2001.

[56] Yong Xia, Lakshminarayanan Subramanian, Ion Stoica, and Shivkumar Kalyanaraman. One More Bit Is Enough. In *ACM SIGCOMM*, Aug 2005.

[57] Ihsan Ayyub Qazi and Taieb Znati. On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks. In *IEEE INFOCOM*, Apr 2008.

[58] Ihsan Ayyub Qazi, Lachlan L. H. Andrew, and Taieb Znati. Two bits are enough. In *ACM SIGCOMM*, Seattle, WA, 17-22 Aug 2008.

[59] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore. The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks. In *IEEE/ACM Trans. Networking, 8(1):87-98*, Feb 2000.

[60] S. Biaz and N. Vaidya. Is the round-trip time correlated with the number of packets in flight. In *USENIX/ACM IMC*, Oct 2003.

[61] R. S. Prasad, M. Jain, and C. Dovrolis. On the effectiveness of delay-based congestion avoidance. In *PFLDNet*, Feb 2004.

[62] S. Rewaskar, J. Kaur, and D. Smith. Why don't delay-based congestion estimators work in the real world? Technical Report TR06-001, Department of Computer Science, UNC Chapel Hill, July 2005.

[63] Van Jacobson. Modified TCP Congestion Avoidance Algorithm. Technical report, 30 Apr 1990. Email to the end2end-interest Mailing List, URL: ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[64] Janey C. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. In *ACM SIGCOMM*, Aug 1996.

[65] Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno, and Sack TCP. *SIGCOMM Comput. Commun. Rev.*, 26:5–21, 1996.

[66] Matthew Mathis and Jamshid Mahdavi. Forward acknowledgement: Refining TCP congestion control. In *ACM SIGCOMM*, Aug 1996.

[67] T. Kelly. Scalable TCP; Improving performance in highspeed wide area networks. *SIGCOMM Comput. Commun. Rev.*, 33(2):83–91, 2003.

[68] R.N.Shorten and D.J.Leith. H-TCP: TCP for high-speed and long-distance networks. In *PFLDnet*, Feb 2004.

[69] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. In *ACM SIGCOMM*, Aug 2002.

[70] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion Control Without Reliability. In *ACM SIGCOMM*, 2006.

[71] Sumitha Bhandarkar, A. L. Narasimha Reddy, Yueping Zhang, and Dimitri Loguinov. Emulating AQM from end hosts. In *ACM SIGCOMM*, Aug 2007.

[72] Bartek Wydrowski, Lachlan L. H. Andrew, and Moshe Zukerman. MaxNet: A Congestion Control Architecture for Scalable Networks. *IEEE Commun. Lett.*, 7(10):511–513, October 2003.

[73] S. Jain, Yueping Zhang, and D. Loguinov. Towards Experimental Evaluation of Explicit Congestion Control. In *IWQoS 2008*, June 2008.

[74] Martin Suchara, Lachlan L. H. Andrew, Ryan Whitt, Krister Jacobsson, Bartek P. Wydrowski, and Steven H. Low. Implementation of provably-stable explicit congestion control. In *Broadnets*, 2008.

[75] C. H. Tai, J. Zhu, and N. Dukkipati. Making large scale deployment of RCP practical for real networks. In *IEEE INFOCOM Minisymposium*, 2008.

[76] S. Athuraliya, V. Li, S. Low, and Q. Yin. REM: Active Queue Management. In *IEEE Network, 15(3):48-53*, May 2001.

[77] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. In *IEEE/ACM Trans. Networking, 1(4):397-413*, Aug 1993.

[78] C. Hollot, V. Misra, D. Towsley, and W. Gong. Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. In *IEEE/ACM Trans. Automatic Control, 47(6):945-959*, Jun 2002.

[79] M. Garetto, T. Salonidis, and E. Knightly. Modeling Per-flow Throughput and Capturing Starvation in CSMA Multi-hop Wireless Networks. In *IEEE INFOCOM*, 2006.

[80] Sumit Rangwala, Apoorva Jindal, Ki-Young Jang, Konstantinos Psounis, and Ramesh Govindan. Understanding Congestion Control in Multi-hop Wireless Mesh Networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, San Francisco, September 2008.

[81] K. Xu, M. Gerla, L. Qi, , and Y. Shu. Enhancing TCP fairness in adhoc wireless networks using neighborhood RED. In *MobiCom*, 2003.

[82] Filipe Abrantes and Manuel Ricardo. XCP for shared-access multi-rate media. *SIGCOMM Comput. Commun. Rev.*, 36(3):27–38, 2006.

[83] Yang Su and Thomas Gross. WXCP: Explicit Congestion Control for Wireless Multi-Hop Networks. In *In IWQoS 05: Proceedings of the 12th International Workshop on Quality of Service*, 2005.

[84] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on TCP performance. *In IEEE Transactions on Mobile Computing*, 2005.

[85] K. Tan, F. Jiang, Q. Zhang, and X. Shen. Congestion Control in Multihop Wireless Networks. *IEEE Transactions on Vehicular Technology*, 2006.

[86] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughout in multihop radio networks. In *IEEE Transactions on Automatic Control*, 1992.

[87] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. In *MobiCom*, 2003.

[88] Yi Li, Lili Qiu, Yin Zhang, Ratul Mahajan, and Eric Rozner. Predictable performance optimization for wireless networks. In *ACM SIGCOMM*, 2008.

[89] Ajay Bakre and B. R. Badrinath. I-tcp: Indirect tcp for mobile hosts. In *ICDCS*, pages 136–143, 1995.

[90] Theodoros Salonidis, Georgios Sotiropoulos, Roch Guerin, and Ramesh Govindan. Online optimization of 802.11 mesh networks. In *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 61–72, New York, NY, USA, 2009. ACM.

[91] Lili Qiu, Yin Zhang, Feng Wang, Mi Kyung Han, and Ratul Mahajan. A general model of wireless interference. In *MobiCom*, 2007.

[92] R. Jain, S. Kalyanaraman, and R. Viswanathan. The OSU Scheme for Congestion Avoidance in ATM Networks: Lessons Learnt and Extensions. In *Performance Evaluation, 31(1):67-88*, Nov 1997.

[93] Robert Shorten, Fabian Wirth, and Douglas Leith. A positive systems model of TCP-like congestion control: Asymptotic results. *IEEE/ACM Trans. Networking*, 14:616–629, 2006.

[94] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *ACM SIGCOMM*, Aug 2001.

[95] Nandita Dukkipati. *Rate Control Protocol (RCP): Congestion control to make flows complete quickly*. PhD thesis, Department of Electrical Engineering, Stanford University, 2008.

[96] ns-2 Network Simulator. http://www.isi.edu/nsnam/ns/.

[97] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *IEEE/ACM Trans. Networking*, Dec 1997.

[98] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing router buffers. *SIGCOMM Comput. Commun. Rev.*, 34(4):281–292, 2004.

[99] Xiaowei Yang, Yanbin Lu, and Lei Zan. Improving xcp to achieve max-min fair bandwidth allocation. *Comput. Netw.*, 54(3):442–461, 2010.

[100] Mihaela Enachescu, Yashar Ganjali, Ashish Goel, Nick McKeown, and Tim Roughgarden. Routers with Very Small Buffers. In *IEEE INFOCOM*, Apr 2006.

[101] Robert N. Shorten and Douglas J. Leith. On queue provisioning, network efficiency and the transmission control protocol. *IEEE/ACM Trans. Netw.*, 15(4):866–877, 2007.

[102] Y. Gu, D. Towsley, C. V. Hollot, and H. Zhang. Congestion Control for Small Buffer High Speed Networks. *IEEE INFOCOM*, 2007.

[103] Amogh Dhamdhere and Constantine Dovrolis. Open issues in router buffer sizing. *SIGCOMM Comput. Commun. Rev.*, 36(1):87–92, 2006.

[104] S. Low, F. Paganini, J. Wang, and J. Doyle. Linear Stability of TCP/RED and a Scalable Control. In *Computer Networks Journal, 43(5):633-647*, Dec 2003.

[105] Lachlan L. H. Andrew, Stephen V. Hanly, Sammy Chan, and Tony Cui. Adaptive Deterministic Packet Marking. *IEEE Comm. Letters*, 10(11):790–792, November 2006.

[106] Micah Adler, Jin yi Cai, Jonathan K. Shapiro, and Don Towsley. Estimation of Congestion Price Using Probabilistic Packet Marking. In *in Proc. IEEE INFOCOM*, Mar-Apr 2003.

[107] R. W. Thommes and M. J. Coates. Deterministic Packet Marking for Time-Varying Congestion Price Estimation. *IEEE/ACM Trans. Networking*, 14(3):592–602, June 2006.

[108] Lachlan L. H. Andrew and Stephen V. Hanly. The Estimation Error of Adaptive Deterministic Packet Marking. In *Proc. Allerton Conf. Communication, Control and Computing*, Urbana-Champaign, IL, September 2006.

[109] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. On the characteristics and origins of internet flow rates. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 309–322, New York, NY, USA, 2002. ACM.

[110] Frank Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[111] S. H. Low and D. E. Lapsley. Optimization Flow Control I: Basic Algorithm and Convergence. *IEEE/ACM Trans. Networking*, 7:861–875, 1999.

[112] H.-K Ryu and S. Chong. Deterministic Packet Marking for Max-Min Flow Control. *IEEE Comm. Letters*, 9(9):856–858, Sep 2005.

[113] T. V. Lakshman, U. Madhow, and Bernhard Suter. TCP/IP Performance with Random Loss and Bidirectional Congestion. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 8:541–555, 2000.

[114] L. L. H. Andrew, S. Floyd, and W. Gang. Common TCP Evaluation Suite. In *Internet draft (work in progress)*, 2008. http://netlab.caltech.edu/lachlan/abstract/draft-irtf-tmrg-tests-00.html.

[115] http://www.apple.com/. http://www.apple.com/iphone/.

[116] www.blackberry.com/. www.blackberry.com/.

[117] Gartner Survey Shows that Corporate Wireless LAN Deployment is Increasing, But Security is a Major Concern, 2006. http://www.gartner.com/it/page.jsp?id=493658.

[118] Micah Z. Brodsky and Robert T. Morris. In defense of wireless carrier sense. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 147–158, New York, NY, USA, 2009. ACM.

[119] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications, year = 2000, volume = 18, pages = 535–547.*

[120] Matthew Gast. *802.11 Wireless Networks: The Definitive Guide, Second Edition (Definitive Guide)*. O'Reilly Media, 2002.

[121] Martin Heusse Franck, Franck Rousseau, Gilles Berger-sabbatel, and Andrzej Duda. Performance Anomaly of 802.11b. In *IEEE INFOCOM*, pages 836–843, 2003.

[122] Manthos Kazantzidis, Mario Gerla, and Sung-Ju Lee. Permissible Throughput Network Feedback for Adaptive Multimedia in AODV MANETs. In *In IEEE International Conference of Communications (ICC*, pages 1352–1356, 2001.

[123] Samarth H. Shah, Kai Chen, and Klara Nahrstedt. Dynamic bandwidth management in single-hop ad hoc wireless networks. *Mob. Netw. Appl.*, 10(1-2):199–217, 2005.

[124] Anand Kashyap, Samrat Ganguly, and Samir R. Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 242–253, New York, NY, USA, 2007. ACM.

[125] Apoorva Jindal and Konstantinos Psounis. Achievable Rate Region of Wireless Multi-hop Networks with 802.11 Scheduling. *IEEE Transactions on Networking*, 2008. (to appear).

[126] GENI. http://www.geni.net.