Measurement-Driven Algorithm and System Design for Wireless and Datacenter Networks

Varun Gupta

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Measurement-Driven Algorithm and System Design for Wireless and Datacenter Networks

Varun Gupta

The growing number of mobile devices and data-intensive applications pose unique challenges for wireless access networks as well as datacenter networks that enable modern cloud-based services. With the enormous increase in volume and complexity of traffic from applications such as video streaming and cloud computing, the interconnection networks have become a major performance bottleneck. In this thesis, we study algorithms and architectures spanning several layers of the networking protocol stack that enable and accelerate novel applications and that are easily deployable and scalable. The design of these algorithms and architectures is motivated by measurements and observations in real world or experimental testbeds.

In the first part of this thesis, we address the challenge of wireless content delivery in crowded areas. We present the *AMuSe* system, whose objective is to enable scalable and adaptive WiFi multicast. *AMuSe* is based on accurate receiver feedback and incurs a small control overhead. This feedback information can be used by the multicast sender to optimize multicast service quality, e.g., by dynamically adjusting transmission bitrate. Specifically, we develop an algorithm for dynamic selection of a subset of the multicast receivers as feedback nodes which periodically send information about the channel quality to the multicast sender. Further, we describe the Multicast Dynamic Rate Adaptation (*MuDRA*) algorithm that utilizes *AMuSe*'s feedback to optimally tune the physical layer multicast rate. *MuDRA* balances fast adaptation to channel conditions and stability, which is essential for multimedia applications.

We implemented the *AMuSe* system on the ORBIT testbed and evaluated its performance in large groups with approximately 200 WiFi nodes. Our extensive experiments demonstrate that *AMuSe* can provide accurate feedback in a dense multicast environment.

It outperforms several alternatives even in the case of external interference and changing network conditions. Further, our experimental evaluation of *MuDRA* on the ORBIT testbed shows that *MuDRA* outperforms other schemes and supports high throughput multicast flows to hundreds of nodes while meeting quality requirements. As an example application, *MuDRA* can support multiple high quality video streams, where 90% of the nodes report excellent or very good video quality.

Next, we specifically focus on ensuring high Quality of Experience (QoE) for video streaming over WiFi multicast. We formulate the problem of joint adaptation of multicast transmission rate and video rate for ensuring high video QoE as a utility maximization problem and propose an online control algorithm called *DYVR* which is based on Lyapunov optimization techniques. We evaluated the performance of *DYVR* through analysis, simulations, and experiments using a testbed composed of Android devices and off the shelf APs. Our evaluation shows that *DYVR* can ensure high video rates while guaranteeing a low but acceptable number of segment losses, buffer underflows, and video rate switches.

We leverage the lessons learnt from *AMuSe* for WiFi to address the performance issues with LTE *evolved Multimedia Broadcast/Multicast Service* (eMBMS). We present the *Dynamic Monitoring* (*DyMo*) system which provides low-overhead and real-time feedback about eMBMS performance. *DyMo* employs eMBMS for broadcasting instructions which indicate the reporting rates as a function of the observed Quality of Service (QoS) for each UE. This simple feedback mechanism collects very limited QoS reports which can be used for network optimization. We evaluated the performance of *DyMo* analytically and via simulations. *DyMo* infers the optimal eMBMS settings with extremely low overhead, while meeting strict QoS requirements under different UE mobility patterns and presence of network component failures.

In the second part of the thesis, we study datacenter networks which are key enablers of the end-user applications such as video streaming and storage. Datacenter applications such as distributed file systems, one-to-many virtual machine migrations, and large-scale data processing involve bulk multicast flows. We propose a hardware and software system for enabling physical layer optical multicast in datacenter networks using passive optical splitters. We built a prototype and developed a simulation environment to evaluate the

performance of the system for bulk multicasting. Our evaluation shows that the optical multicast architecture can achieve higher throughput and lower latency than IP multicast and peer-to-peer multicast schemes with lower switching energy consumption.

Finally, we study the problem of congestion control in datacenter networks. Quantized Congestion Control (QCN), a switch-supported standard, utilizes direct multi-bit feedback from the network for hardware rate limiting. Although QCN has been shown to be fast-reacting and effective, being a Layer-2 technology limits its adoption in IP-routed Layer 3 datacenters. We address several design challenges to overcome QCN feedback's Layer-2 limitation and use it to design window-based congestion control (QCN-CC) and load balancing (QCN-LB) schemes. Our extensive simulations, based on real world workloads, demonstrate the advantages of explicit, multi-bit congestion feedback, especially in a typical environment where intra-datacenter traffic with short Round Trip Times (RTT: tens of μs) run in conjunction with web-facing traffic with long RTTs (tens of milliseconds).

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my Ph.D. advisor, Professor Gil Zussman. His guidance, motivation, and high standards were instrumental in shaping me as a researcher. Gil provided me the freedom to find my own research path and I will be forever grateful for his mentorship. Additionally, I would like to express my appreciation to my entire thesis committee consisting of Professors Keren Bergman, Vishal Misra, Debasis Mitra, and Javad Ghaderi.

I would like to extend a special thanks to Yigal Bejerano and Craig Gutterman. Yigal has been a close collaborator, mentor, and a great friend throughout my time at Columbia. Craig has been a parnter in crime while working on papers, presentations, and attending meetings for the AMuSe project. Although I did not get to collaborate with everyone from the Wim.Net research group, I cherished worked alongside with extraordinary Ph.D. students Maria Gorlatova, Berk Birand, Robert Margolies, Tingjun Chen, Jelena Marasevic, Saleh Soltan, and Alex Loh. I thank them for their extraordinary support and feedback.

This thesis would not have been possible without several successful collaborations. Through my work on the the AMuSe project, I have had the unique opportunity to collaborate and learn from many people. I am thankful to have worked closely with Jaime Ferragut, Andy Xu, Bohan Wu, and Hannaneh Pasandi. I was especially fortunate to have the opportunity to mentor Sohan Kumar, David Alvarez, and Rodda John. The discussions with Payman Samadi and Howard Wang were extremely useful in helping me understand optics and develop new research ideas.

I would like to thank Abdul Kabbani, Sridhar Raman, and Terry Lam for hosting me during an internship at Google. The feedback and input from Rong Pan and Milad Sharif was invaluable in shaping my research on datacenter congestion control. Last, but certainly not least, I am fortunate to have worked with Katherine Guo, Chandru Raman, and Chun-

Nam Yu at Bell Labs.

Finally, and above all, I am grateful to my family. My family's encouragement and motivation paved the way for my Ph.D. The journey through graduate school itself would not have been possible without the uncondiitonal love and support of my wife Rupa who stood like a rock beside me at every step.

Financial Support: The research described in this thesis was supported in part by New York City Media Lab Combine program grant, NSF grants CNS-16-50685, CNS-14-23105, CNS-10-54856, and CIAN NSF ERC under grant EEC-0812072.

To Mummy, Papa, Neha, and Rupa.

Chapter 1

Introduction

The growing number of mobile devices and data-intensive applications pose unique challenges for wireless access networks as well as datacenter networks that enable modern cloud-based services. For instance, video is expected to contribute 75% of all the mobile traffic by 2020 [23] while constituting about 32% of the total cloud traffic [4]. This problem will only become more pressing as emerging technologies such as Internet of Things (IoT) devices, virtual reality, and distributed machine learning change communication patterns and impose stricter requirements on throughput and latency.

With the enormous increase in volume, variability, and complexity of traffic from applications such as video streaming and cloud computing, the wireless and wired interconnection networks have become a major performance bottleneck. One solution to address these traffic demands is to use more equipment. For example, wireless small-cell technologies enable deploying more wireless base stations, each with smaller ranges. In datacenters, high port count switches, 100Gb Ethernet links, or InfiniBand links could be used for high performance enterprise networks. However, these approaches are expensive, hard to scale, and unsuitable for all applications. Thus, novel approaches to scale and manage the networks of the future must be developed.

In this thesis, we study algorithms and architectures spanning several layers of the networking protocol stack that enable and accelerate novel applications and that are easily deployable and scalable. Our focus is on two different domains of wireless networks and datacenter networks, both of which are critical for the overall performance of end-user

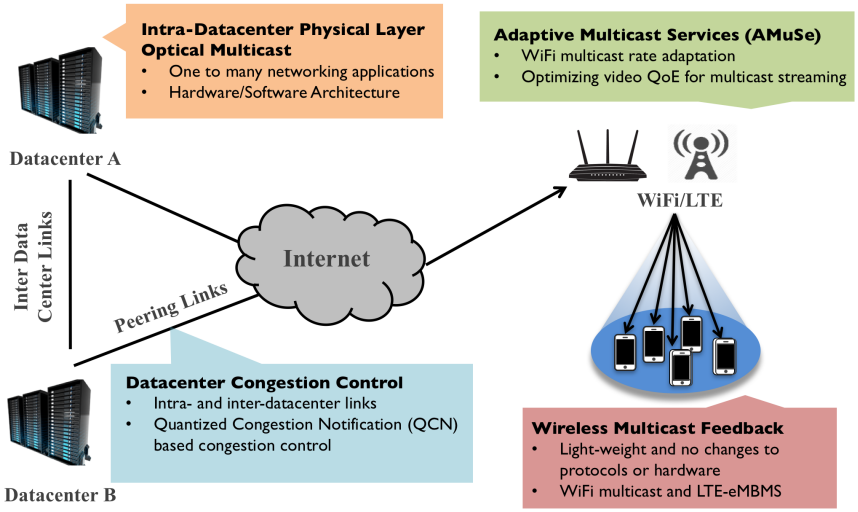


Figure 1.1: An overview of the wireless and datacenter networks

applications.

First, we focus on wireless networks that form a major source of user demand, and consider the problem of content delivery in crowded areas through wireless multicast both for WiFi and cellular networks. Our focus is on 3 key problems associated with multicast: (i) collecting reliable feedback with low overhead, (ii) enabling dynamic rate adaptation, and (iii) optimizing video Quality of Experience (QoE). We then turn to datacenter networks and study new architectures and algorithms that can enable efficient multicast at the physical layer using optical switches. Finally, we consider congestion control approaches for datacenter networks which satisfy the twin goals of ensuring low latencies while maintaining high throughput with the traffic consisting of a mix of intra-datacenter and inter-datacenter flows.

1.1 Background

We start by providing background information for each domain. In the following section, we summarize our contributions.

1.1.1 Content Delivery Through Wireless Multicast

Recent years have witnessed a rapid growth of mobile devices equipped with WiFi or LTE interfaces which allow users to access the Internet anywhere and any time. Mobile video is expected to contribute 75% of all the mobile traffic by 2020 [29]. The popularity of live video services such as Facebook Live and YouTube TV could severely stress wireless networks in crowded areas. The growing need to support larger demands for multimedia content using limited resources in dense areas has prompted the design of several solutions by both industry and academia.

Many of these solutions are typically based on dense deployments of WiFi Access Points (APs) [21, 24, 201] or cellular Base Stations (BSs) [87, 124, 201]. These dense deployments provide dedicated content delivery to each user. Such solutions, besides requiring considerable capital and operational expenditure, may not meet user expectations, due to extensive interference between adjacent APs/BSs.

Wireless multicast is an attractive approach for content delivery to large groups of users interested in venue specific content (e.g., in sports arenas and entertainment centers). However, WiFi networks provide limited multicast support where multicast is either handled through a series of unicast packets or transmission at a low rate (e.g., 6Mbps even for 802.11ac) without a feedback mechanism that guarantees service quality. Similarly, the evolved Multicast and Broadcast Services (eMBMS) standard [27] for LTE networks does not specify a mechanism for collecting real-time feedback from receivers which is important for tuning parameters such as transmission rates and error correction. Due to the limited ability to collect feedback, deployment of wireless multicast is very challenging.

In crowded areas with tens of thousands of receivers (e.g., [87]), even infrequent feedback reports by each receiver may result in high signaling overhead. This could translate to blocking of unicast traffic in cellular networks and high packet losses due to contention in WiFi. Existing approaches for tuning multicast parameters (e.g., transmission rate) rely on extensive radio frequency surveys which are not scalable, lead to low throughput, and are oblivious to environmental changes. Despite recent advances [67, 226], the practicality and

scalability of wireless multicast has been limited [152]. Thus, there is a need for a multicast system that dynamically adapts the transmission rates in response to receiver performance.

Further, since a key application for multicast is video streaming, ensuring high Quality of Experience (QoE) for video streaming is essential. Recent research has shown that videos which play at lower bitrates or freeze frequently lead to higher abandoment which translates to lost revenue for video providers, advertisers, and sub-optimal use of network resources [85]. Existing unicast video streaming techniques rely on segmenting the video in chunks of fixed duration, encoding each chunk in several bitrates, and transmitting a chunk whose bitrate matches the estimated throughput in each timeslot. However, applying techniques similar to unicast video streaming for multicast is not straightforward. Besides the lack of reliable feedback and a rate adaptation mechanism, tuning video rates is challenging in the presence of multiple receivers with diverse channel qualities.

Thus, our objectives are threefold: (i) to design efficient feedback schemes for wireless multicast - both WiFi and cellular, (ii) to develop dynamic multicast rate adaptation mechanisms, and (iii) to design schemes for optimizing video QoE for multicast.

1.1.2 Optical Multicast for Datacenter Networks

Datacenters are key enablers of user services such as video streaming discussed in the previous section. Similar to wireless networks, the workload in datacenter networks is evolving and a large fraction consists of one-to-many traffic patterns. Applications such as distributed file systems [92], one-to-many virtual machine migrations [65], and large-scale data processing [61] involve bulk multicast flows. The main barrier in deploying IP multicast is the requirement of complex configurations on all the switches and routers of the data center network. Due to this, most datacenters transmit such multicast traffic through a series of unicast transmissions. These methods are inherently inefficient since they send multiple copies of the same data. At the same time, datacenter networks are usually oversubscribed due to the enormous switching cost and cabling complexity associated with scaling Ethernet network. In such a scenario, these bulk transfers lead to congestion at the aggregation and core layers.

Recent solutions [88, 212] have proposes offloading high-volume traffic to an Optical Circuit Switched (OCS) network using an Optical Space Switch (OSS) for point-to-point traffic but this approach is ineffective for multicast traffic.. Faster delivery of complex traffic patterns such as multicast, incast and all-to-all-cast over an OCS substrate requires leveraging optics' advanced functionalities. For example, using passive optical splitters for multicast and time and wavelength multiplexing for incast. A key challenge in implementing an end-to-end system containing optical modules in data center networks, is the control and management integration with conventional data center packet-switched networks. Software Defined Networkin (SDN) along with cross-layer designs [60, 133] can overcome this critical challenge and provide the optical modules functionalities seamlessly to the higher layers. Therefore, our objective is to design a hardware and software architecture to enable optical multicast in datacenter networks.

1.1.3 Datacenter Congestion Control

While new physical layer technologies such as optical multicast can provide significant benefits in datacenter networks, improving the performance of existing packet-switched transport networks is critical. Extensive studies show that modern datacenter traffic are typically composed of a large fraction (as high as 80%) of short ($<$ 10KB) mice flows [56] with the rest being throughput intensive elephant flows. The mice flows, especially are latency sensitive and even a small fraction of late arrivals can cause a ripple effect that degrades application's overall performance [80]. Additionally, inter-datacenter or user bound traffic often traverses through WAN peering links. The switches facing such links typically include a large buffer to avoid packet losses, which in turn may lead to large delays. Furthermore, ensuring high throughput over peering links is essential since their cost is negotiated based on peak utilization (95th percentile).

To meet these requirements, datacenter transports must simultaneously deliver two competing aspects of performance: high throughput and low latency. Congestion control plays a crucial role in meeting these demands but unfortunately traditional loss-based TCP fails to achieve optimal performance. QCN (Quantized Congestion Notification) has been de-

veloped to provide congestion control at the Ethernet layer (L2) for the IEEE 802.1Qau standard. The QCN feedback signal is a multi-bit indication of congestion sent directly from the congestion point. Despite the obvious advantages of QCN feedback over Explicit Congestion Notifications (ECN) [232] or RTT [156], its practicality is limited due to L2 operation and the necessity to make changes to host hardware. Therefore, our objective is to design easily deployable congestion control algorithms that leverage the benefits of QCN feedback. .

1.2 Contributions

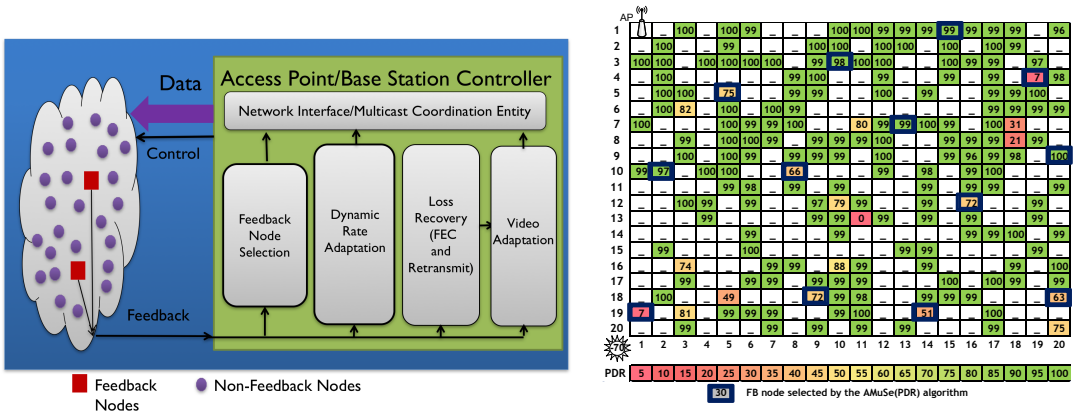
In this section, we describe the contributions made to each domain.

1.2.1 Adaptive Wireless Multicast

The first part of this thesis focuses on large scale content delivery via wireless multicast both for WiFi and cellular networks. We address the research challenges associated with several aspects of wireless multicast as shown in Fig. 1.2(a). For WiFi multicast, we address challenges related to feedback, rate adaptatation, and video quality optimization as part of the *AMuSe* (Adaptive Multicast Services) system [52]. For LTE-eMBMS, our focus is on efficient large-scale monitoring using light-weight feedback. Below, we describe these contributions in more detail.

Light-weight feedback mechanism: In Chapter 2 we study approaches for light-weight feedback for WiFi multicast. First, in order to better understand the performance of existing schemes, we conducted extensive experiments with over 200 WiFi nodes on the ORBIT testbed in [50, 51, 101] as shown in Fig. 1.2(b). Our observations show that some nodes, which we define as *abnormal nodes*, suffer from low *Packet Delivery Ratio (PDR)*, even when the AP is transmitting at a low bit-rate and there is no external interference. Furthermore, this set of abnormal nodes varies across experiments. Further, our evaluations showed that effects of external interference can be highly localized.

Existing feedback mechanisms are a variation of the *Leader-Based* feedback scheme



(a) (b)

Figure 1.2: (a) A block diagram of the contributions to adaptive wireless multicast for both WiFi and cellular networks: a light-weight feedback mechanism, multicast dynamic rate adaptation, loss recovery and Forward Error Correction (FEC), and video rate adaptation, (b) A heatmap of the average Packet Delivery Ratio (PDR) values for 200 nodes receiving multicast data from a single Access Point in the ORBIT testbed.

where feedback is provided by a few nodes, typically the nodes with the lowest channel quality. The above observations provide an intuitive explanation as to why leader-based feedback protocols may perform poorly since they may not accurately capture the network performance. Existing schemes cannot provide QoS guarantees or high throughput.

Next, we introduce a low-overhead *AMuSe* feedback mechanism [50, 101] that does not require changes to the existing IEEE 802.11 standard and can be implemented as a light-weight application on any WiFi enabled device with minor or no modifications. The *AMuSe* feedback mechanism dynamically divides the network into clusters based on the adjacency of nodes and maximum cluster size. In each cluster, one node is selected as the feedback node that updates the Access Point (AP) about its channel quality. We implemented the *AMuSe* feedback on the ORBIT testbed with more than 200 WiFi nodes as a distributed protocol with very low control overhead and evaluated its performance. Our results demonstrate the ability of the *AMuSe* feedback mechanism to provide feedback about the performance of wireless multicast. *AMuSe* feedback leads to lowest number of false

positives of received packets when compared to leader-based feedback mechanisms.

Dynamic rate adaptation: In Chapter 3, we propose the design and evaluation of the Multicast Dynamic Rate Adaptation (*MuDRA*) algorithm for WiFi. Our experiments on ORBIT show that when the multicast rate exceeds an optimal rate, called the *target-rate*, numerous receivers lose a large number of packets that cannot be recovered. *MuDRA* [53,99] addresses this issue and detects when the system operates at the target-rate.

We experimentally demonstrate that *MuDRA* can swiftly converge to the target rate while meeting QoS requirements, e.g., ensuring that more than 85% of packets are correctly received by at least 95% of the 200 nodes in our setup. The losses are recovered using application-level Forward Error Correction (FEC). *MuDRA* achieves 6x higher throughput than current state-of-the-art schemes in diverse conditions. We also show the feasibility of using *AMuSe* system, comprising of *AMuSe* feedback and *MuDRA* , for streaming video by emulating video transfers over experimental data. *AMuSe* can deliver 3 or 4 high definition H.264 videos (each one of 4Mbps) with over 90% of the nodes receiving video quality classified as excellent or good based on user perception.

We also present an interactive web-based application that illustrates the performance of the overall *AMuSe* system based on experimental traces collected on the ORBIT testbed [102, 103]. Each experimental trace consisted of channel measurements at 150-200 nodes using several metrics such Link Quality, Packet Delivery Ratio (PDR) etc. The application allows to compare the performance of *AMuSe* with other schemes in different scenarios such as different channel conditions and interfering transmissions. For each scenario, the application shows the dynamic conditions over a period of time on the testbed from the appropriate experimental traces as well as syntactic scenarios based on manipulating the measured data.

Optimizing Video QoE for Multicast Streaming: In Chapter 4, we address the problem optimizing QoE of video over WiFi multicast by jointly tuning multicast transmission and video rates. We formulate the problem of optimizing the QoE as a utility maximization problem and propose an online algorithm *DYVR* for solving utility maximization problem. We derive performance guarantees for the performance of the *DYVR* algorithm using the Lyapunov optimization framework [158]. Our analysis shows that *DYVR* can achieve

$O(W, \frac{1}{W})$ tradeoff between achieving the utility and satisfying the QoE constraints, where W is an algorithmic parameter.

Next, we describe an architecture for video streaming over WiFi multicast that can be easily integrated with existing Adaptive Bit Rate (ABR) services. We implement the architecture on a wireless testbed comprised of Android devices and a commercially available WiFi AP. We evaluate the performance of the *DYVR* algorithm both through simulations and experiments and compare it against other schemes. Our evaluations show that *DYVR* yields close to optimal performance while meeting the QoE constraints under a variety of conditions.

Dynamic Monitoring of LTE-eMBMS: In Chapter 5, we describe the Dynamic Monitoring (*DyMo*) system for low-overhead monitoring of LTE-eMBMS. The design of *DyMo* is based on the lessons learnt from WiFi. *DyMo* provides accurate QoS reports with low overhead in dense environments by identifying the maximum *SNR threshold* so that only a small number of UEs (User Equipments) with SNR below the threshold suffer from poor service. *DyMo* leverages the broadcast capabilities of eMBMS to quickly disseminate *stochastic group instructions* to a large number of receivers for adjusting their feedback frequency. Each instruction is targeted at a sub-group of UEs and the sub-group divisions are further refined based on the QoS reports.

We develop a *Two-step estimation* algorithm which can efficiently identify the SNR Threshold as a one time estimation. We also develop an *Iterative estimation* algorithm for estimating the SNR Threshold iteratively, when the distribution changes due to UE mobility or environmental changes, such as network component failures. Our analysis shows that the *Two-step estimation* and *Iterative estimation* algorithms can infer the SNR Threshold with a small error and limited number of QoS reports. It is also shown that they outperform the *Order-Statistics estimation* method, a well-known statistical method, which relies on sampling UEs with a fixed probability. For instance, the *Two-step estimation* requires only 400 reports when estimating the $1th$ percentile to limit the error to 0.3% for each re-estimation. The *Iterative estimation* algorithm performs even better than the *Two-step estimation* and the maximum estimation error can be bounded according to the maximum

change of SNR Threshold.

We conducted extensive at-scale simulations, based on real eMBMS radio survey measurements from a stadium and an urban area. Our simulations show that both in a stadium-like and urban area, *DyMo* detects the *eMBMS SNR value of the* 0.1% *percentile with Root Mean Square Error (RMSE) of* 0.05% with only 5 messages per second in total across the whole network. This is at least 8 times better than *Order-Statistics estimation* based methods. *DyMo* also infers the optimal SNR Threshold with RMSE of 0.3 dB regardless of the UE population size, while the error of the best *Order-Statistics estimation* method is above 1 dB. *DyMo* violates the outlier bound (of 0.1%) with RMSE of at most 0.35 while the best *Order-Statistics estimation* method incurs RMSE of over 4 times as compared to *DyMo*. The simulations also show that after a failure, *DyMo* converges instantly (i.e., in a single reporting interval) to the optimal SNR Threshold. Thus, *DyMo* is able to infer the maximum MCS while preserving QoS constraints.

1.2.2 Optical Multicast System for Datacenter Networks

In Chapter 6, we present the design and experimental evaluation of an Optical Multicast System for Data Center Networks - an integrated hardware-software system architecture that enables native physical layer optical multicast in data center networks. The hardware architecture is built on a hybrid network, i.e. the Top-of-Rack switches are simultaneously aggregated by a L2/L3 packet-switched network and an optical circuit-switched network provided by an Optical Space Switch (OSS) (OSS is a switching substrate that provides an optical circuit between any idle input and output ports, without optical to electronic conversion [3, 17]). The OSS is also the substrate to connect passive optical splitters to the optical network. The control plane software runs on the SDN controller and communicates with the hosts through the packet-switched network. The control plane manages the routing operations at the electronic and optical switches, connectivity of optical splitters, and optimally assigns optical splitters to flows using a resource allocation algorithm.

We evaluated the performance of the system through simulations and experiments on a prototype testbed. Experimental and simulation results show that optical multicast provides

similar throughput for delivering multicast flows as IP multicast but (i) does not require applying complex configurations on all the switches/routers of the data center to enable IP multicast since multicast trees are directly created by the SDN controller, (ii) has superior energy efficiency since it is built on an OCS network that consumes less energy than an EPS network, (iii) is future-proof due to the data rate transparency of the system. Compared to unicast transmissions where the throughput is inversely proportional to the number of receivers, optical multicast have steady performance irrespective to the multicast group size. Compared to peer-to-peer multicast, it provides at minimum an order of magnitude higher throughput for flows with sizes under 250 MB. Adding the optical multicast system to a data center with a sole non-blocking packet-swtiched network decreases the total energy consumption by 50% while delivering 20 TB of data containing 15% multicast flows. The latency also drops by 55%. The improvements are more significant in the case of over-subscribed EPS networks and larger volumes of multicast flows.

1.2.3 QCN-Based Congestion Control for Datacenter Networks

In Chapter 7, we describe a novel congestion control algorithm *QCN-CC* which is based on simple modifications of existing TCP implementations and utilizes QCN feedback messages at the transport layer, i.e. the TCP layer. *QCN-CC* is readily deployable without changes in commodity Network Interface Cards (NICs) and precludes the need of high-performance hardware or software based timers. We illustrate how to make QCN feedback messages across the boundary between Layer 2 and Layer 3 domains and our proposed changes can be readily incorporated in most current commercial switches

We compared the performance of *QCN-CC* against other state-of-the-art congestion control mechanisms using simulations based on realistic datacenter workloads. A key aspect of these evaluations is including both intra and inter-datacenter traffic which poses significant challenges for congestion control due to mismtach of flow RTTs by orders of magnitudes. Our simulations show that *QCN-CC* significantly reduces the tail latency of short flows by as much as 6x as compared to TCP, DCTCP, and DCQCN while incurring no penalty on the overall throughput. *QCN-CC* can also provide high utilization at WAN

peering links while minimizing packet drops which is an important performance metric for datacenter network operators.

1.3 Contributions to Literature

The work about efficient feedback collection for WiFi multicast, described in Chapter 2 was published in the proceedings of IEEE ICNP'13 [50] while an extended journal version appeared in IEEE/ACM Transactions on Networking [101]. Besides this, a summary of lessons learnt from the large scale experimentation on the ORBIT testbed were summarized in an invited paper in the proceedings of GENI Research and Educational Experiment Workshop'14 (GREE) [51]. A demo of the concepts described in this work was presented at IEEE LCN'15 [102].

The design and experimental evaluation of *MuDRA* for dynamic rate adaptation for WiFi multicast in Chapter 3 appeared in the proceedings of IEEE INFOCOM'16 [99]. A technical report can be found in [100] and an extended version was submitted to a journal. A demo of the rate adaptation process was presented at and appeared in the proceedings of IEEE INFOCOM'16 [103][1].

A demo of the algorithm and the platform described in Chapter 4 was presented and appeared in the proceedings of IEEE INFOCOM'17 [106].

The description and evaluation of the DyMo system for efficient monitoring of large scale eMBMS deployments as described in Chapter 5 appeared in IEEE INFOCOM'17 [54]. An extended version with additional results and proofs that could not be included in the conference version was fast-tracked to IEEE/ACM Transactions on Networking and the technical report can be found in [55].

The work on wireless multicast described in the thesis was performed as a part of *AMuSe* project at Columbia University. The overview of the results spanning the entire project (including the work presented here) appeared in the proceedings of IEEE IC-

[1]The same demo was presented in the NYC Media Lab Annual Summit'15 and won the second prize among more than 100 demos.

CCN'16 [52] as an invited paper.

The concepts of using Software Defined Networking (SDN) and optics for enabling data-center multicast as explained in Chapter 6 appeared as a poster in the proceedings of ACM SIGCOMM'14 [181] and in the proceedings of European Conference on Optical Communication'14 (ECOC) [182]. A paper with details about system design and evaluation appeared in the journal Optics Express [183].

Part I

Adaptive Multicast Services

Chapter 2

LIGHT-WEIGHT FEEDBACK FOR WIRELESS MULTICAST

2.1 Introduction

Current state of the art techniques using IEEE 802.11 for content delivery leverage either unicast or multicast data delivery. Commercial products [21,24] rely on unicast for streaming the content to individual users. With standards such as 802.11ac promising total speeds up to 800 Mbps using multi-user MIMO, it is theoretically possible to serve video streams to hundreds of users. However, recent studies [107,164] throw cold water on this promise. A large number of neighboring APs leads to hidden terminal problems and this coupled with increased interference sensitivity due to channel bonding, makes the entire approach highly susceptible to interference. Extrapolating from studies on 802.11n [107,164], it seems that 802.11ac-based unicast to multiple receivers may not be able to support more than a hundred users, assuming all of them have 802.11ac capable devices.

On the other hand, WiFi multicast services are rarely used by practical content delivery applications. Standard WiFi broadcast/multicast frames are transmitted at a fixed and low bitrate without any feedback. This raises several known reliability and efficiency issues. While some commercial products [24] are experimenting with WiFi multicast deployments

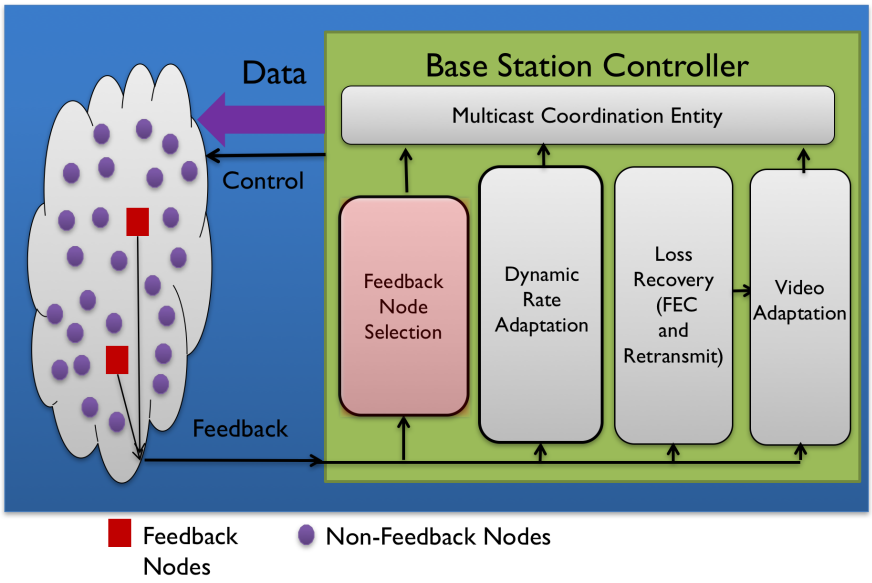


Figure 2.1: The *AMuSe* feedback mechanism (highlighted in red) as a part of the overall *AMuSe* system.

for crowded environments, there remain several challenges to its widespread adoption. In particular, a recently published IETF Internet Draft highlights several open technical problems for WiFi multicast [152]. High packet loss due to interference and the hidden node problem can significantly degrade service quality. On the other hand, transmitting at low bitrates leads to low network utilization. As described in Section 2.2, there are numerous studies that propose solutions for overcoming these limitations from two aspects. One aims to reduce the overhead of feedback information to the multicast sender. The other aims to improve message reliability based on available feedback information. All the existing schemes, however, suffer from one or more issues including lack of scalability, inability to guarantee high service quality, or compliance with existing standards. Further, *none of the schemes have been tested experimentally at scale.*

We have been developing the *AMuSe* (**A**daptive **Mu**lticast **Se**rvices) system [52] for scalable and efficient delivery of multimedia content to a very large number of WiFi nodes in crowded venues (e.g., sport arenas, lecture halls, and transportation hubs). *AMuSe* does not require changes to the IEEE 802.11 protocol or wireless hardware. Therefore, it can be

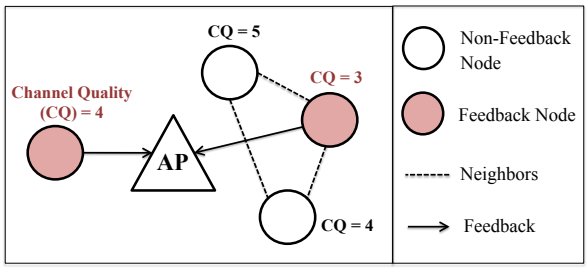


Figure 2.2: Feedback node selection by *AMuSe*. A node with the poorest channel quality in every neighborhood is selected as a Feedback node. Each feedback node periodically sends updates about the service quality to the Access Point.

deployed as an overlay network on existing wireless infrastructure. This overlay network is comprised of *AMuSe* server on the network side and light-weight application-layer software on the mobile devices. This makes *AMuSe* attractive for delivering live video content to a dense user population that shares common interests (e.g., providing simultaneous video feeds of multiple camera angles in a sports arena).

The *AMuSe* system consists of the following components: (i) an efficient feedback mechanism, (ii) dynamic rate adaptation algorithm, and (iii) loss recovery and content control. In this chapter, we focus on design and evaluation of the *AMuSe* feedback mechanism as shown in Fig. 2.1. In subsequent chapters, we will describe the other components.

The work on *AMuSe* project started in collaboration with Bell Labs, Nokia with Yigal Bejerano and Katherine Guo. Jaime Ferragut, Craig Gutterman, and Thyaga Nandagopal made numerous important contributions in the design and data analysis of experiments.

2.1.1 *AMuSe* feedback

We consider the use of WiFi multicast to address the challenge of providing scalable and efficient delivery of multimedia content to a very large number of WiFi nodes in a small geographical region (e.g., sport arenas, lecture halls, and transportation hubs). This is an attractive approach for delivering live video content to a dense user population that shares

common interests (e.g., providing simultaneous video feeds of multiple camera angles in a sports arena).

The core challenge in providing such a service is collecting limited yet sufficient feedback from the users for optimizing the network performance. To address this challenge, *we introduce AMuSe (**A**daptive **Mu**lticast **Se**rvices), a low-overhead feedback mechanism which leverages the existing WiFi standards for tuning the network parameters, i.e., optimizing the network utilization while preserving Quality of Service (QoS) requirements. AMuSe* is based on the following hypothesis, which was reported in [32] and is validated in this chapter.
Main Hypothesis: A cluster of adjacent nodes experience similar channel quality and suffer from similar interference levels. Hence, a node v with a worse channel condition than its adjacent neighbors can represent the service quality observed by the nodes in the cluster.

AMuSe dynamically divides the nodes in a network into clusters based on the adjacency of nodes and maximum cluster size (D m). In each cluster, one node is selected as a *Feedback (FB) node* and the FB node updates the AP about its service quality, e.g., channel quality (an example is shown in Fig. 2.2). The AP, in response, may take several actions such as[1]:
(i) **Rate Adaptation:** *AMuSe* can allow the APs to transmit multicast traffic at the highest possible bitrate while meeting constraints set by a network operator, i.e. ensuring high *Packet Delivery Ratio* (PDR) for a large fraction of the nodes.
(ii) **Tuning FEC:** We demonstrate in this chapter that ensuring 100% packet deliveries to all nodes is challenging. In large multicast groups, even a small amount of packet losses at nodes could lead to large packet retransmissions. In such situations, dynamically tuning application-level FEC might be a more suitable option. Feedback from *AMuSe* can be used to adjust the amount of FEC dynamically.
(iii) **Detecting Interference:** *AMuSe* collects detailed packet statistics which can be used to identify causes of packet loss in the network such as collisions and noise. For instance, packet losses that occur at the same time at multiple nodes can help pinpoint the location of the interference.

[1]The actions of the AP will require changes only at the AP side which is relatively straightforward.

AMuSe can be implemented as a light-weight application on any WiFi enabled device with minor or no modifications to the receiver devices and does not require changes to the existing 802.11 standard. The *AMuSe* feedback mechanism allows multicast service operators to balance between the number of FB nodes, the accuracy of the feedback, and the overall convergence time by controlling *AMuSe* parameters, such as the cluster radius D. *AMuSe* ensures that every node is at most D m away from an FB node with similar or weaker channel quality. To ensure sparse FB node density, any pair of FB nodes are at least D m apart which results in low communication overhead. The problem of selecting FB nodes which meet the above requirements is a variant of the well known *Minimal Independent Dominating Set* problem [150]. Although this problem is NP-hard, we prove that *AMuSe* can find a solution with a small constant approximation ratio.

2.1.2 Experimental Evaluation

We evaluated *AMuSe* on the large-scale ORBIT testbed [14] using over 200 WiFi nodes by implementing *AMuSe* on the application layer at each device. In all of our experiments, one node served as the AP and it sent a continuous multicast flow to all the other nodes, which acted as receivers. We first study the variation of channel quality metrics in different scenarios, (e.g., varying external interference levels, different transmission bit rates). The observations from these experiments serve as guiding principles for the design of *AMuSe*.

We observe that during any experiment, some nodes, which will be defined as *abnormal nodes*, suffer from low PDR, even when the AP is transmitting at a low bitrate and there is no external interference. Furthermore, this set of abnormal nodes varies across experiments.

We collected detailed channel and service statistics from all the nodes. They include the *Link Quality*[2] (LQ) reported by each node's WiFi card as representative of its observed received signal strength (RSS), its PDR, and its distance from the AP. Our preliminary evaluations show only moderate correlation between the nodes' LQ and the experienced

[2]Although LQ is not a standard measurement metric, we observed that the reported LQ by the Atheros chipsets indicates the RSS in db normalized to a reference value of -110 dBm (thermal noise).

PDR and a weak correlation between the nodes' distance from the multicast AP and the PDR values.

To validate the Main Hypothesis, we consider all the possible clusters with radius 3 and 6 m and calculate the Standard Deviation (STD) of the LQ and PDR values in the clusters at different bitrate and noise-levels. Our experiments indeed show low LQ and PDR STDs between the nodes in a cluster. However, as we increase the transmission bitrate or the noise level, we observe an increase in STD for the PDR values. We also notice that clusters with a small radius have lower LQ and PDR STDs than larger clusters.

We assess the feedback reports produced by *AMuSe* when the channel quality is evaluated according to the nodes' LQ, PDR, or a combination of them. These variants are denoted as AMuSe-LQ, AMuSe-PDR, and AMuSe-Mix respectively. We compare their performance to other feedback node selection schemes; *K-Worst* [73, 215], which selects the receivers with the worst channel condition as FB nodes, and *Random*, which selects a fixed number of random FB nodes. To evaluate the quality of an FB node selection, we compute the number of non-FB nodes that experience PDR value strictly lower than their respective FB node. We refer to these nodes as *Poorly Represented Nodes* (PRNs). We show that AMuSe-PDR and AMuSe-Mix produce a negligible number of PRNs and they outperform the other schemes when evaluated with different multicast bitrates and various noise levels. AMuSe-LQ and K-Worst have comparable performance, and are significantly better than the Random scheme.

Furthermore, we assess the performance of *AMuSe* as a service quality predictor in the event of environment changes. More specifically, we first select the FB nodes of the different variants at a given network setting. We then, compute the number of poorly represented nodes when using the same FB nodes, but after changing the multicast bitrate or the noise-level. We observe that at low bitrates AMuSe-LQ has slightly less PRNs than AMuSe-PDR, while AMuSe-PDR has similar performance to K-Worst. We notice a different trend when operating at a high multicast bitrate, in which AMuSe-PDR outperformed AMuSe-LQ and K-Worst. In all evaluations AMuSe-Mix was the best variant while Random, suffered from a very high number of PRNs. We explain these observations and provide additional results

in Section 2.6.

Our experimental results demonstrate the ability of *AMuSe* to effectively provide feedback about the performance and quality of wireless multicast services. In turn, this feedback can be used for tuning the network parameters (e.g., rate adaptation, FEC configuration, and interference classification) to optimize multimedia content delivery.

2.1.3 Chapter Organization

We describe the network settings and our objectives in Sections 2.3 and 2.4 respectively. We present testbed evaluation of the design of *AMuSe* in Section 2.5 and the experimental results of evaluating channel quality metrics in Section 2.6. Finally, the evaluation of the performance of *AMuSe* is presented in Section 2.7 for both the static and dynamic cases.

2.2 Related work

Various methods have been proposed for multimedia content dissemination to multiple receivers. They leverage either unicast or multicast data delivery. This brief overview describes the most relevant studies Commercial products [21,24] rely on unicast for streaming content to individual users. This approach requires deployment of numerous APs and it does not scale to crowded areas. Alternatively, the basic 802.11 multicast mechanism without any node feedback simply sets the transmission bitrate to the lowest rate. Cellular networks also operate without any node feedback and set the transmission bitrate to a low value, assuming some nodes are located near the cell edge. Any multicast mechanism without feedback results in low network utilization.

Many of the schemes to improve multicast services are based on integrating Automatic Repeat Request (ARQ) mechanisms into the protocol architecture [67, 73, 131, 199, 215], adding Forward Error Correction (FEC) packets to the multicast stream [36, 71], or both [221]. Other studies propose rate adaptation mechanisms for improved network utilization [144].

In all cases, a key requirement is having appropriate feedback from the receivers re-

garding their observed service quality. These feedback mechanisms can be classified as follows: (i) *Individual Feedback* from multicast receivers, (ii) Leader-Based Protocol with acknowledgements (LBP-ACK), (iii) *Pseudo-Broadcast*, and (iv) *Leader-Based Protocol* with negative acknowledgements (LBP-NACK).

Individual Feedback mechanisms require all receivers to send acknowledgements of received packets either at the link layer [98, 196, 199, 214, 215], the application layer [221], or using periodic updates [36]. With *More Reliable Groupcast* (MRG) [89, 115] from IEEE 802.11 working group, each receiver transmits a bit-map of correctly received packets. Using this feedback, the sender determines lost packets and retransmits them to the group. This approach offers reliability but incurs high feedback overhead with large groups. The other three approaches reduce this overhead as follows.

The LBP-ACK approach [208, 215] provides scalability by selecting a subset of the receivers to provide feedback. The Pseudo-Broadcast approach [67, 73, 163], converts the multicast feed to a unicast flow and sends it to one leader, typically, the receiver with the weakest channel. The leader acknowledges the reception of the unicast flow. The other receivers receive packets by listening to the channel in promiscuous mode. The LBP-NACK approach [131, 142, 144] improves Pseudo-Broadcast by allowing the other receivers to send NACKs for lost packets. After receiving the ACK from the leader, the sender can infer successful transmission to all receivers since an NACK would collide with the leader's ACK.

With LBP-ACK and Pseudo-Broadcast, the selection of the leader(s) or subset of the receivers to provide feedback, can compromise service reliability. In Fig. 2.3(a), the leader v acknowledges a packet on behalf of node u, even though node u suffers from external interference that prevents correct reception of the packet. In Fig. 2.3(b), the node u might have an uplink transmission collide with the multicast packet from the AP, but since the leader correctly receives the multicast packet, the AP thinks the transmission has succeeded.

The LBP-NACK scheme requires changes to the standard and suffers from lack of reliability since a non-leader cannot reply with a NACK if it cannot identify a corrupted packet. Furthermore, due to the capture effect, the AP may be able to decode the ACK and ignore NACK messages. A major drawback of the LBP-NACK scheme is lack of fine-grained in-

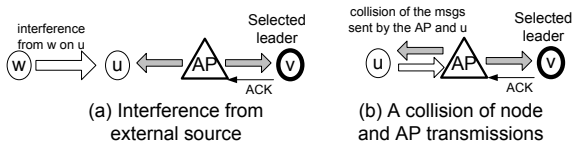


Figure 2.3: Unreliable packet delivery by the LBP and the Pseudo-Broadcast approach.

Table 2.1: Multicast: Features of related work

	Scalable **(a)**	**QoS Guarantees** **(b)**	**High Util.** **(c)**	**Standards Compatible** **(d)**	**Low Cost** **(e)**
Unicast	x	√	x	√	x
Basic multicast	√	x	x	√	√
Individual Feedback	x	√	x	x	√
Pseudo Broadcast	√	x	x	√	√
LBP-NACK	√	x	x	x	√
AMuSe	√	√	√	√	√

formation about packet losses. Consider an example with 100 nodes in a multicast group, each with PDR of 99%. The expected fraction of packets for which NACK messages are received is $1 - .99^{100}$, which translates to roughly 63% of the packets. Thus, even in the case of network performing well, the AP observes poor performance.

Table 2.1 summarizes the main features of existing approaches. In summary, at least one of the following weaknesses hinders their performance: (i) requirement of feedback from a large number of receivers, (ii) ignorance of AP to interference-related packet loss, (ii) low network utilization to compensate for lack of feedback information or due to abnormal nodes, (iv) requirement of changes to standard WiFi protocol, or (v) expensive deployment of numerous APs. This motivates our desire for a scalable solution that improves reliability

of multimedia delivery for WiFi deployments.

2.3 Network Setting

We consider an IEEE 802.11 WLAN and focus on a single AP serving a dense deployment of WiFi devices or *nodes*. A multicast server sends data to the AP and the AP transmits this data using multicast to all the nodes in its transmission range. There could be several sources of external interference in the network including transmissions from nodes within the network, adjacent APs, and nodes outside the network.

We follow the model where a node may report its service quality (e.g., channel quality) to an AP or multicast server. The AP or the multicast server, in response, may decide to adjust the FEC, adjust the transmission bitrate, retransmit lost packets, or execute a combination of the above. In practice, the AP and the multicast server are two separate logical entities and may reside in multiple network layers. Only the AP, however, is responsible for adjusting the network layer parameters. To simplify presentation, in the rest of the chapter we refer to AP as a representation of the combination of an AP and a multicast server.

At any given time, each node is associated with a single AP and nodes are assumed to have a quasi-static mobility pattern. In other words, nodes are free to move from place to place, but they tend to stay in the same physical locations for several minutes or more. This is a reasonable assumption for various crowded venues, such as sports arenas or transportation hubs. We assume that mobile devices can estimate their locations (e.g., by using one of the methods in [155]) with an accuracy of a few meters, and also determine if they are *static*[3] or *mobile*.

[3]We consider a node static, if its movement is restricted to a few meters.

2.4 Objective

We focus on designing a light-weight feedback mechanism for supporting scalable WiFi multicast services for a very large number of nodes. This allows APs[4] to monitor the network conditions and to take appropriate actions for improving the multicast service quality while meeting various service delivery constraints. We rely on the following observation reported in [32]:

Observation: *A cluster of adjacent nodes experience similar channel quality and suffer from similar interference levels. Hence, a node v with worse channel condition than its adjacent neighbors can represent the service quality observed by the nodes in the cluster.*

Based on this observation, the nodes can be grouped into clusters of adjacent nodes and a single *Feedback (FB) node* from each cluster can represent that particular cluster. The FB node can be used to report the channel quality of the cluster to the AP. Our feedback mechanism should ensure the following requirements:

(i) The FB nodes should accurately represent the network conditions in their neighborhood. This implies that the channel state experienced by non-FB nodes should not be significantly worse than the channel state reported by FB nodes.

(ii) The FB nodes should be well distributed throughout the network. In other words, the distance between the FB and non-FB nodes should be small. This ensures that the AP is informed about any interference even if it affects a small area.

(iii) The FB nodes should be responsive to changes of the service condition and should accurately report the impact of environmental changes, such as the multicast bitrate or external interference.

We now provide a formal definition of our objective. Given any FB node selection scheme and assume that every non-FB-node is represented by a single FB-node, typically

[4]To simplify our presentation, we assume that *AMuSe* is implemented as a software module on the APs. In practice, *AMuSe* can be realized as an independent server or even a cloud service.

the closest FB-node. A non-FB-node is considered a *Poorly Represented Node* (PRN) if its PDR is $\epsilon > 0$ below the PDR of its representing FB-node. We refer to ϵ as the *PRN Gap*. Consequently, our objective can be defined as follows;

Objective: *Consider an upper bound on the number of FB nodes or their density*[5] *as well as a fixed PRN-Gap* $\epsilon > 0$. *Design a low-communication FB node selection mechanism that minimizes the following metrics:*

- *Number of PRNs in normal operation as well as after environment changes, e.g. bi-trate or noise level changes.*
- *Maximum distance between a non-FB-node and its representing FB node.*

2.5 The *AMuSe* Feedback Mechanism

This section provides an overview of the *AMuSe* feedback mechanism. For any given D we define two nodes to be *D-adjacent* if they are separated by a distance of at most D. In order to find a small set of FB nodes that can provide accurate reports, *AMuSe* should satisfy the following requirements.

(i) Each node should be D-adjacent to an FB node.

(ii) An FB node must have similar or weaker channel quality than its D-adjacent nodes.

(iii) Any two FB nodes cannot be D-adjacent.

In order to evaluate the channel quality, various metrics can be considered, including *Received Signal Strength* (RSS), *Signal-to-Noise Ratio* (SNR) and *Packet Delivery Ratio* (PDR). We experimentally compare LQ^2 and PDR as channel quality metrics in Section 2.6.

2.5.1 The Feedback Node Selection Algorithm

We present a semi-distributed process for FB node selection, where some nodes volunteer to serve as FB nodes, and the AP selects the best candidates. If node location information

[5]The FB node density can be enforced by requiring a minimal distance D between any two FB nodes.

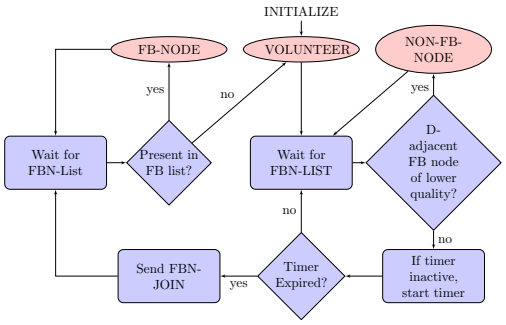


Figure 2.4: State diagram of the *AMuSe* FB node selection algorithm at each node. All nodes initialize in the VOLUNTEER state.

and observed channel quality are known, then the AP can easily select the ideal set of FB nodes. Yet, this is not feasible in practice for large groups. Hence, we seek to minimize the number of nodes that send their information to the AP as part of the FB node selection process, while ensuring that a small set of FB nodes meeting the above requirements is selected.

The AP periodically (e.g., once every $\tau_{AP} = 500$ ms in our experiments) multicasts an FBN-LIST message with a list of FB nodes (these messages can be sent multiple times for reliable transmissions and do not incur overhead, since they are 1-2 packets long). Each entry in the FBN-LIST contains the node ID[6], its reported location[7], its reported channel quality, and a measure of the PDR[8].

Each node is in one of three states:

- **FB-NODE -** A node that has been selected as FB node.
- **VOLUNTEER -** A node that is not aware of any D-adjacent FB node with lower or similar channel quality and can serve as an FB node.

[6]Nodes can be assigned temporary virtual IDs to maintain privacy.

[7]Relying on a user to be truthful about its location/channel quality could lead to denial-of-service attacks. Yet, we shelve this orthogonal discussion.

[8]This can be easily changed to report the last acknowledged packet sequence number to support finer granularity of message reliability.

- **NON-FB-NODE -** A node that either is in a transient state or is aware of a D-adjacent FB node with similar or lower channel quality.

Fig. 2.4 presents the state transition diagram for each node. When a node v joins the network, it is in the VOLUNTEER state. The node waits for an FBN-LIST message, and checks if there are any D-adjacent FB nodes in this list with similar or weaker channel quality. If there are any such nodes, node v switches to the NON-FB-NODE state and records the list of D-adjacent FB nodes in the FBN-LIST message with similar or weaker channel quality.

If there are no such nodes, node v starts a random back-off timer for a period chosen in the interval $[0, T]$ (our experiments use the maximum receiver back-off timer $T = 5$ seconds). The random timer solves the problem of many nodes overwhelming the WiFi channel and AP with FBN-JOIN messages in the situation of changes in channel condition. During this countdown, if node v learns of a D-adjacent FB node from a FBN-LIST message, then it cancels its countdown, and switches to a NON-FB-NODE state. Otherwise, upon expiry of the timer, it sends a FBN-JOIN message to the AP, and waits to see if its ID appears on the next FBN-LIST. The FBN-JOIN message contains the node ID, node location, and the observed channel quality (e.g., the node PDR and LQ). If node v appears on the FBN-LIST, it switches to the FB-NODE state. If not, it repeats the back-off process again until it leaves the VOLUNTEER state. At any time, upon receipt of an FBN-LIST message, if an FB node v does not find itself on the FBN-LIST, it ceases to be in the FB-NODE state. In this case, the node returns to the VOLUNTEER state and waits for the next FBN-LIST to either (i) switch to the NON-FB-NODE state due to the existence of a D-adjacent node of lower quality, or (ii) send the FBN-JOIN message again after the back-off timer expires.

An important property of this FB node selection algorithm is that the FB node selection is done in a semi-distributed manner, since a node volunteers to serve as an FB node, only if there is no other FB node in its vicinity with weaker channel quality. Thus, the AP is only responsible to resolve conflicts when several D-adjacent nodes volunteer simultaneously and to prune unnecessary FB nodes. Consequently, after receiving FBN-JOIN messages and

before sending a FBN-LIST message, the AP runs the *node pruning algorithm*, described in Section 2.5.3 to decide which nodes are FB nodes.

Each FB node periodically (e.g., once every $\tau_{FB} = 500$ ms in our experiments) sends REPORT messages to update the AP about the channel and service quality experienced by the node, and thus its representative cluster. If the AP does not receive any message from one of the FB nodes for a given duration, (for example, $3\tau_{FB}$ used in our experiments), then the AP removes it from the list of FB nodes.

A few aspects of the *AMuSe* feedback are worth pointing out.

(i) *AMuSe* does not require the nodes to listen to all the traffic on the network. All they have to do is listen to the AP on the multicast group address. This conserves energy at the receivers.

(ii) *AMuSe* does not require the location information for nodes to be very precise. As mentioned in Section 2.3, coarse granularity is acceptable, as long as the accuracy is in the order of few meters, which has been demonstrated by some studies as feasible and practical [72].

(iii) *AMuSe* provides variable levels of reliability by fine-tuning the combination of AP node selection frequency τ_{AP}, the receiver reporting frequency, τ_{FB}, the maximum receiver back-off timer T, and the node adjacency distance D. *AMuSe* can ensure more reliable and frequent reports at a cost of more overhead. Instead of a single control, *AMuSe* provides multiple control knobs, giving greater flexibility to the operator to provide different types of service for various multicast streams.

(iv) Fourth, as described above, *AMuSe* reports can be used for optimizing different aspects of WiFi multicast services, such as rate-adaptation, FEC configuration and interference classification. To this end, the REPORT messages may carry different information. For instance, in [50] we showed that PDR and LQ information is sufficient for performing rate adaptation, while reporting about received and lost packets is required for interference classification.

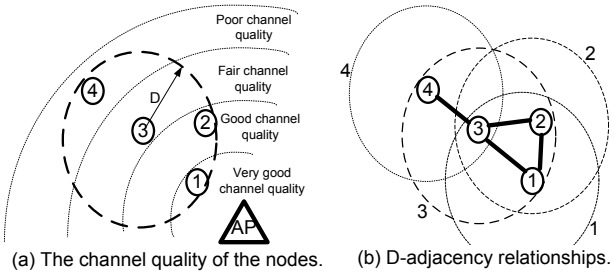


Figure 2.5: An example of a wireless network a single AP and 4 receivers. All 3 requirements described in Section 2.5 for an accurate feedback selection are important for this example.

2.5.2 Illustrative Example

Consider the network shown in Fig. 2.5(a) with a single AP and four receivers. Assume that numbers labeling the nodes denote their IDs and the order in which they join the multicast service at this AP. There are four different channel quality levels: very good, good, fair and poor as experienced by node 1, 2, 3, and 4 respectively. Fig. 2.5(b) shows a circle with radius D around every node, say node v, where each node, u, inside the circle of v is D-adjacent to node v. Hence, nodes u and v are considered neighbors to one another.

In this example, we demonstrate the importance of all three requirements mentioned at the beginning of this section on the quality and density of the set of FB nodes. Assume first that the FB nodes have to meet only requirement (i) and (ii), but not (iii). Under these guidelines, at the moment each node joins the multicast, it has a weaker channel quality than all its neighbors, and therefore, it is selected as an FB node. At the end of the process, the network contains four FB nodes. It is easy to see that this approach does not scale for large groups.

Now, let us assume that requirement (iii) is enforced. Right after a node joins the network, the set of FB nodes is optimized. When node 1 joins, it becomes the FB node. After node 2 joins, node 2 becomes the FB node, while node 1 becomes a non-FB node because of (iii). After node 3 joins, it becomes an FB node while both node 1 and 2 become non-FB nodes because all three nodes are D-adjacent to one another. After node 4 joins, it becomes an FB node, while node 3 becomes a non-FB node. In addition, node

2 becomes an FB node again. Notice that node 2 switches state twice, after node 3 and 4 joins respectively. However, after each node joins the multicast group, the set of FB nodes is optimal.

This example shows that while *AMuSe* FB node selection algorithm satisfies all three requirements, it may cause churn as nodes enter and leave the FB-NODE state. We show next that the selected set of FB nodes is near-optimal when the set of nodes receiving the multicast do not change.

2.5.3 The Node Pruning Algorithm

As described above, the FB node selection process ensures that every receiver is D-adjacent to a *candidate* node with similar or weaker channel condition. The list of candidates at the AP contains the current FB nodes as well as the nodes in the VOLUNTEER state. Thus, the AP is responsible to trim unnecessary candidates to select a small set of FB nodes such that any pair of nodes in the set are not D-adjacent.

The problem of finding the minimum set of FB nodes that meets the three requirements above is a variant of the *minimum dominating set problem*, which is a known NP-complete problem even in the case of unit disk graph [150]. Below we present a heuristic algorithm that selects a near optimal set of candidates that meet our three requirements.

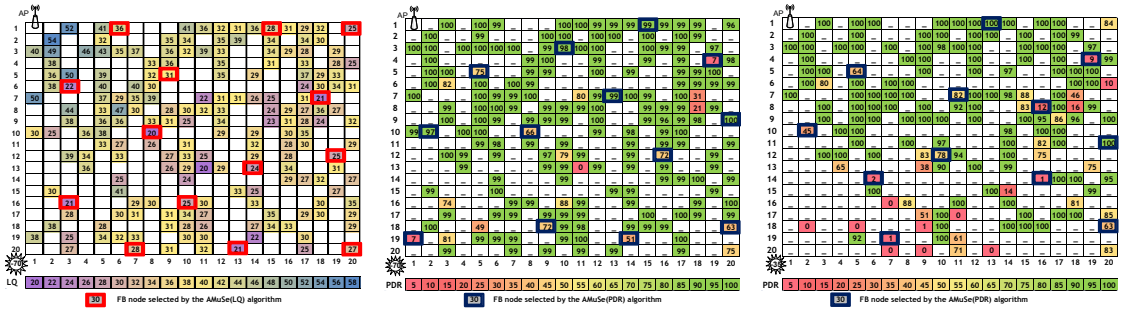
The heuristic algorithm: The AP creates a list L of the candidates sorted in increasing order according to their channel quality. Then, it iteratively selects the first candidate v in L as an FB node and remove v and all its D-adjacent nodes from L. The algorithm ends when L is empty.

Let F denote the FB nodes selected by the heuristic algorithm and OPT denote the optimal set of FB nodes among all nodes, our algorithm ensures the following property:

Proposition 1. $|F| \leq 5 \cdot |OPT|$. *If the channel quality is a monotonic decreasing function with the distance from the AP then* $|F| \leq 3 \cdot |OPT|$

For proof see Appendix 2.A.

Stability vs. optimality trade-off: As illustrated in Section 2.5.2, a naive implemen-



(a) Link Quality Heatmap, noise = -70 dBm.

(b) Packet Delivery Ratio Heatmap, noise = -70 dBm.

(c) Packet Delivery Ratio Heatmap, noise = -35 dBm.

Figure 2.6: Link Quality (LQ) and Packet Delivery Ratio (PDR) heatmaps at the AP for $D = 6$ meters with transmission bitrate of 12 Mbps and noise level of -70 dBm and -35 dBm. The FB nodes are highlighted with a thick border in red in the LQ heatmap and in blue in the PDR heatmap. Empty locations represent nodes that did not produce LQ or PDR reports and they are excluded from our experiments. Nodes with $PDR = 0$ are active nodes that reported LQ values but were unable to decode packets. These nodes are excluded from the FB node selection process. Note that the minimum threshold below which a node does not become an FB node is configurable.

tation of the heuristic algorithm may cause churn of FB nodes, which obstructs system stability. Since node pruning is done by the AP, the algorithm can be easily modified to prevent churn, for instance by giving higher priorities to already selected FB nodes or relaxing the distance constraint between FB nodes. In our experiments, we also observed rapid switching of FB nodes due to minor variations in channel qualities. In this case, ensuring that the difference between channel quality of a non-FB and FB node is greater than some value greater than zero before a non-FB node volunteers is an effective solution. Although striking a proper balance between system stability and optimality of the FB node selection is a central topic in the design of *AMuSe*, it is beyond the scope of this thesis.

2.6 Experimental Evaluation of Testbed Environment

We validated *AMuSe* experimentally using the 400-node ORBIT testbed [14]. We describe these experiments in this section. We use the *Link Quality*[2] (LQ) metric reported by a

Table 2.2: Evaluation Parameters

Parameter	Definition
LQ_i	Link Quality of node i with the AP.
P_i^{vec}	A vector of the packets received by node i.
(x_i, y_i)	(row, column) location of node i.
TX_{AP}	Broadcast/Multicast transmission rate at the AP.

node's WiFi card as representative of its observed RSS. We first consider the following set of auxiliary hypotheses used to validate our main hypotheses in Section 2.1.1.

H1: There is a correlation between the PDR and LQ values observed by a node.

H2: Clustered nodes experience similar LQ and PDR.

H3: Clustered nodes suffer from similar interference.

2.6.1 The ORBIT Testbed and Experiment Settings

The ORBIT testbed [14] consists of a dynamically configurable grid of 20×20 (400 overall) nodes each with an 802.11 radio. The grid separation between nodes is 1 meter and in addition, the testbed provides a noise generator with four noise antennas at the corners of the grid whose attenuation can be independently controlled, permitting the emulation of a richer topology. In order to avoid performance artifacts stemming from a mismatch of WiFi hardware and software, we select the subset of nodes equipped with *Atheros 5212/5213* wireless cards with *ath5k* wireless driver. Furthermore, we remove unresponsive nodes (nodes with hardware issues) in the grid before every experiment. This results in approximately 200 nodes participating in each experiment.

We implemented the *AMuSe* system as an application layer program for the AP and the clients, running on all nodes. Each node is identified by its (row, column) location.

The node at the corner $(1, 1)$ serves as a single multicast AP, configured in master mode, and it uses channel 40 of 802.11a[9] to send a multicast UDP flow with a transmission power of 1 mW= 0 dBm. The other nodes are the multicast receivers, configured in managed mode. This means that in practice our experiments consider *at most a quarter of the transmission range of an AP*. Each UDP packet is 1400 bytes in payload length and the payload data contains sequence number for each packet in order to identify missing packets at the nodes. While we consider a single multicast group in our experiments, *AMuSe* can allow for monitoring of several multicast groups individually. If several multicast groups should be monitored together, then a control multicast group can be setup.

Every node keeps track of the parameters described in Table 2.2, which we process off-line after each experiment. The received or dropped packets are marked by 1s or 0s respectively in a boolean vector P_i^{vec} stored at each node i. The *packet delivery ratio* (PDR) value of each node i is calculated from its P_i^{vec} vector. Note that the throughput measured at each node is a function of the PDR as well as the bitrate and is different from the transmission throughput at the AP. The testbed hardware and software allows us to measure the LQ or RSS values from the user-space. The PDR values can be measured on any commodity hardware by measuring the received packets. It is possible that some environments such as iOS do not provide LQ or RSS information to the user-space. In such cases, *AMuSe* can rely on PDR measurements alone. As we show later, *AMuSe* with PDR measurements alone can provide reliable feedback.

2.6.2 Experiment Description

We now describe the types of experiments conducted to validate our hypotheses presented earlier in this section.

Different Bitrates: We fix the AP multicast transmission bitrate, denoted by TX_{AP}, to different values allowed by the card (6, 9, 12, 18, 24, 36, 48, 54 Mbps), each bitrate for a

[9]We observed that channel 40 at the 5 Ghz band suffers from lower external interference levels on the ORBIT grid than the channels at 2.4Ghz band.

duration of 10 seconds. We repeat these experiments 10 times at different times of the day without any external noise.

Different Noise Levels: We fix the AP multicast transmission bitrate to 12 Mbps and turn on the noise generator near node (20, 1). The noise generator is configured to provide AWGN noise for the entire spectrum of channel 40. Starting with -70 dBm (low noise), we vary noise power in steps of 5 dBm up to -35 dBm (high noise).

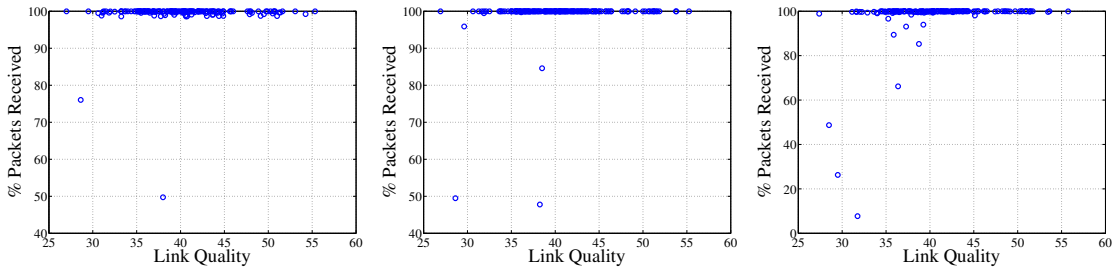
Fig. 2.6 presents three sample heatmaps of one run of the experiments, when $TX_{AP} = 12$ Mbps and external noise of -70 dBm and -35 dBm generated near node (20,1). Each heatmap shows the active nodes used in the experiment and either the LQ or PDR values that they experienced, in addition to the FB nodes that the AP has selected with D-adjacency parameter of 6 meters. Nodes marked with thick red or blue border are FB nodes selected by the *AMuSe* scheme. Nodes with $PDR = 0$ are active nodes that reported LQ values but unable to decode packets in the experiment run. For example, node (13,11) with $LQ = 20$ and $PDR = 0$ in Fig. 2.6(a) and 2.6(b) for a noise level at -70 dBm. These nodes are excluded from the FB node selection algorithm.

An interesting observation is that a selected FB node v may have higher PDR (or LQ) values than an adjacent non-FB node, say u. Such a situation results from the independent-set property of the selected FB nodes and it may occur if u is D-adjacent to another FB node with even lower PDR (or LQ) values. For instance, in Fig. 2.6(b) Node $(7, 13)$ with PDR of 99% was selected as FB node although it has a neighbor, Node $(7, 11)$, with PDR of 80%. The reason is that Node $(7, 11)$ is 6-adjacent to FB node $(10, 8)$ with PDR of 66%.

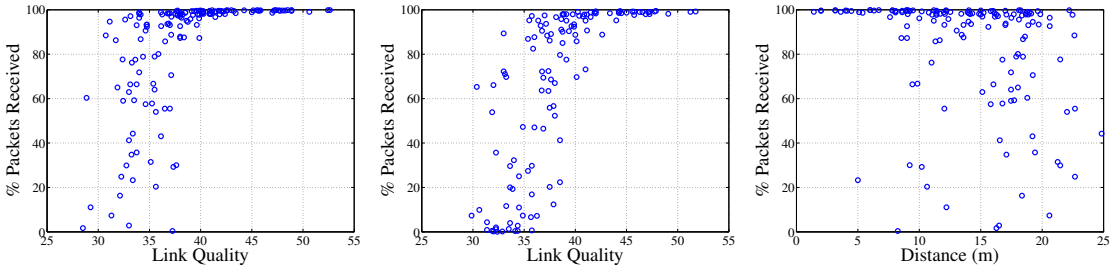
2.6.3 Hypotheses Testing

We turn to test our hypotheses based on the information collected from the experiments described in Section 2.6.2.

H1 - Correlation between PDR and LQ: Fig. 2.7(a)-2.7(e) demonstrate the correlation between the PDR of a node with respect to its LQ for different transmission bitrates without external noise, whereas, Fig. 2.7(f) shows the correlation between the PDR of a node with respect to its distance from the AP at a transmission rate of 48 Mbps. PDR values are



(a) PDR vs. LQ, $TX_{AP} = 6$ Mbps. (b) PDR vs. LQ, $TX_{AP} = 24$ Mbps. (c) PDR vs. LQ, $TX_{AP} = 36$ Mbps.



(d) PDR vs. LQ, $TX_{AP} = 48$ Mbps. (e) PDR vs. LQ, $TX_{AP} = 54$ Mbps. (f) PDR vs. distance, $TX_{AP} = 48$ Mbps.

Figure 2.7: Experimental results for testing hypothesis H1 and verifying the presence of abnormal nodes.

close to 100% for almost all nodes for bitrates up to 24 Mbps (Fig. 2.7(a)-2.7(b)). Some degradation of PDR values is observed for bitrates of 36 Mbps (Fig. 2.7(c)) and even higher variance of PDR values are seen for 48 Mbps (Fig. 2.7(d)) and above.

Fig. 2.7(d) and Fig. 2.7(e) show that the correlation between the PDR and LQ is not very strong, suggesting that nodes with the same LQ value may have significantly different PDR. Fig. 2.7(f) illustrates very weak correlation between the PDR of a node and its proximity to the AP (with $TX_{AP} = 48$ Mbps), and some of the nodes adjacent to the AP suffer from low PDR. For instance, Fig. 2.7(f) shows that one of the nodes with distance of 5 meters from the AP suffers from PDR of 25%. This observed variation of PDR with LQ as well as variation of PDR with distance to the AP is consistent with prior work, e.g., [170], [209], [123] and [97].

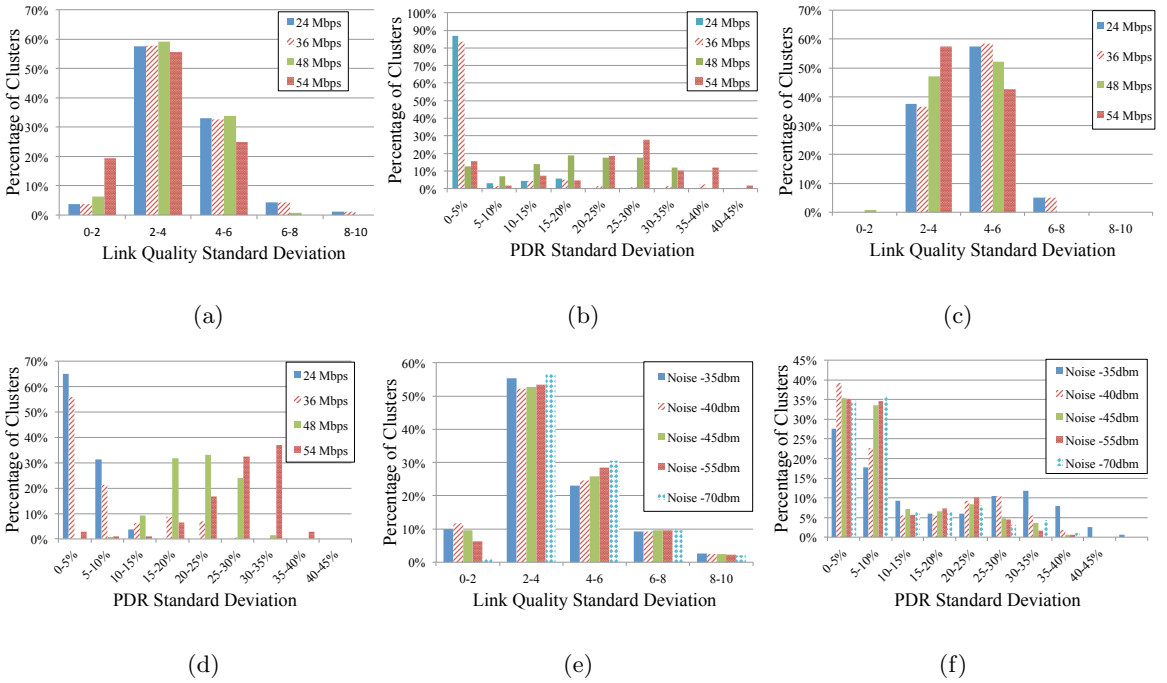
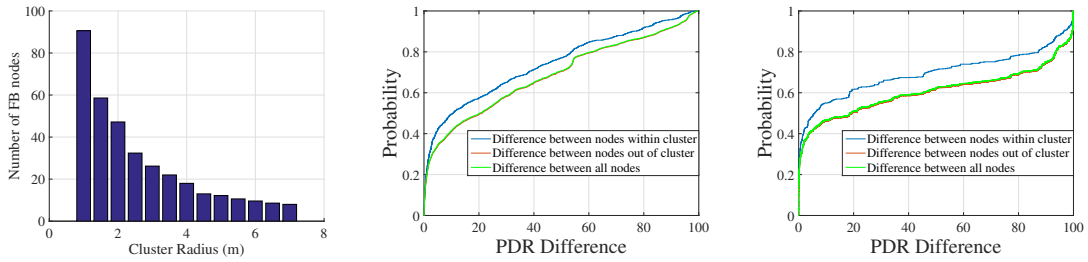


Figure 2.8: Experimental results for testing hypotheses H2–H3: (a) LQ STD: varying TX_{AP} without noise, cluster size = $3m$, (b) PDR STD: varying TX_{AP} without noise, cluster size = $3m$, (c) LQ STD: varying TX_{AP} without noise, cluster size = $6m$, (d) PDR STD: varying TX_{AP} without noise, cluster size = $6m$, (e) LQ STD: varying noise, TX_{AP} = 12 Mbps, cluster size = $3m$, and (f) PDR STD: varying noise, TX_{AP} = 12 Mbps, cluster size = $3m$.

H2 - Clustered nodes experience similar LQ and PDR: We measure the standard deviation (STD) of LQ and PDR without noise in each cluster radius of 3 and 6 meters on the grid, where each cluster contains an FB node and all its neighbors Histograms of the distribution of the LQ and PDR STD in different clusters are shown in Fig. 2.8(a)-2.8(d). We measure the same distributions in the presence of various noise levels with a cluster radius of 3 meters, and plot the results in Fig. 2.8(e) and Fig. 2.8(f). We expect the STD across clusters to be a good measurement of how similar the PDR and the LQ values are.

Fig. 2.8(a), Fig. 2.8(c), and Fig. 2.8(e) show that the LQ STD is very similar across all the bitrates regardless of the noise levels. This indicates that although adjacent nodes experience similar LQ (and similar RSS), the LQ metrics do not capture the effect of external



(a) (b) (c)

Figure 2.9: The impact of clustering: (a) the number of FB nodes for different cluster sizes, (b) CDF of PDR differences of pairs of nodes within and across clusters for no external noise and bitrate of 54Mbps, and (c) CDF of PDR differences of pairs of nodes within and across clusters for external noise of -30dBm and bitrate of 12Mbps.

interference and bitrate variation. By comparing Fig. 2.8(a) and Fig. 2.8(c), we see that a higher percentage of clusters report higher LQ STD for cluster size 6 m than with cluster size 3 m.

We now consider the distribution of the PDR STD values. Fig. 2.8(b) shows that with $TX_{AP} \leq 36$ Mbps, only very few clusters show significant deviations ($> 5\%$) in PDR. This is because most nodes have PDR above 99% when $TX_{AP} \leq 36$ Mbps as shown in Fig. 2.7. However, the variability of the PDR becomes evident at higher bitrates. By comparing Fig. 2.8(b) and Fig. 2.8(d), we observe that a higher percentage of clusters report higher PDR STD for cluster size 6 m as compared to cluster size 3 m. Further, we see in Fig. 2.8(d) that at higher bitrates, PDR STD is higher for a significant number of clusters.

As shown in Fig. 2.8(f), interference introduces noticeable deviations ($> 5\%$) in PDR across nearly two-thirds of the clusters. To understand this, we revisit the heatmaps in Fig. 2.6(c). It is clear that the PDR values are decreasing for nodes near the bottom-left corner where the noise generator is located. The nodes which are not able to decode the AP beacons (at a bitrate of 6 Mbps) disconnect from the AP, are not shown on the heatmap, and are not included in the variance calculations. The nodes which report a 0 PDR value are the ones that fail to receive any multicast packet. These nodes are shown in the heatmap red with a 0 value. At higher noise levels, many more nodes report PDR values of 0. This

explains the high levels of PDR variance observed in Fig. 2.8(f).

The increase in LQ and PDR STD with the cluster size point to the inherent tradeoff in FB node selection process using both LQ and PDR as the quality metrics. The system should ideally operate in a mode where a large fraction of the nodes experience high PDR and the PDR STD is very low. Increasing the cluster size reduces the number of FB nodes, however, leads to increased STD of quality metrics in clusters, particularly the PDR STD at higher bitrates. The average number of FB nodes for different cluster sizes is shown in Fig. 2.9(a). The FB overhead of *AMuSe* is directly proportional to the number of FB nodes. Each FB node, periodically sends an FB message which is roughly 100 bytes long. The frequency of feedback messages is application-specific e.g., for multicast rate adaptation application, 1s could be sufficient [99]. This implies that 50 FB nodes will add an overhead of 40Kbps. In our case, 50 FB nodes correspond to a cluster radius of 3m from Fig. 2.9(a). The FB overhead is much smaller than the multicast throughput measured at the AP (order of Mbps even for bitrate of 6Mbps). The above observations serve as a good motivation to carefully set the parameters for the FB node selection algorithm.

Finally, we demonstrate that clustering is not redundant by comparing the proximity of channel quality values within and across clusters. Fig. 2.9(b) shows the CDF of the PDR differences between pairs of nodes inside and across clusters for bitrate of 54Mbps and no noise for a cluster radius of 3m. We chose bitrate of 54Mbps for ease of exposition. Roughly 60% of the node pairs have PDR differences less than 20% within a cluster while fewer than 50% of pairs have differences less than 20% across clusters. Similarly, Fig. 2.9(c) shows the CDF of the PDR differences between pairs of nodes inside and across clusters for bitrate of 12Mbps and external noise of -30dBm for a cluster radius of 3m. In this case also, the differences are similar. These results show that clustering is effective in grouping nodes with similar channel qualities.

H3 - Clustered nodes suffer from similar interference: Fig. 2.6 shows that external noise has a largely local effect near the noise source. Moreoever, Fig. 2.8(f) shows that even with a small cluster size of 3 meters, the PDR STD can be high due to external interference. The above two observations validate the need for a well-distributed and non-sparse set of FB

nodes to report the values of quality metrics in order to reflect the interference experienced by receivers.

Our experiments also show that increasing TX_{AP} has an impact on all nodes, and that beyond a certain bitrate, the PDR of many nodes drops below 90%, as shown in Fig. 2.7(d) and Fig. 2.7(f). Thus, it is critical to assign TX_{AP} appropriate values in order to improve the multicast service.

2.6.4 Abnormal Nodes

In general, we refer to a node with low PDR as *abnormal*. Specifically, in our experiments, a node is *abnormal* if its PDR is below the abnormal threshold $H = 90\%$. In contrast, a node is *normal* if its PDR is at least $H = 90\%$. In this section, we study the number of abnormal nodes as a function of the TX_{AP} and the link quality (LQ). Fig. 2.7(a)-2.7(d) show how PDR varies with LQ for each node in a single experiment run with TX_{AP} bitrates of $6, 24, 36$ and 48 Mbps respectively. Results from all values of TX_{AP} (including ones not shown here) show that the number of abnormal nodes increases with the increase of TX_{AP}.

In Fig. 2.7(a)-2.7(c), PDR values are close to 100% for a large fraction of the nodes for bitrates up to 36 Mbps. However, Fig. 2.7(a) demonstrates that even in the extreme case of very low TX_{AP} without any interference some of the nodes (two in this case) are abnormal and suffer from low PDR.

The set of abnormal nodes remained small when we increase TX_{AP} to higher bitrates until 36 Mbps, as shown in Fig. 2.7(b) and Fig. 2.7(c). The number of abnormal nodes increases significantly once TX_{AP} reaches 48 Mbps. Surprisingly, the set of abnormal nodes is not the same in all experiments.

2.7 Feedback Node Selection

The primary objective of this section is to study the performance of feedback node selection schemes. We compare *AMuSe* FB node selection scheme with other schemes and in the process, validate our *main hypothesis* from Section 2.1.1. We consider the following schemes

including the three flavors of *AMuSe* that select either the LQ, the PDR or a mix as the metric which is used by the AP for selecting FB nodes.

(i) AMuSe-LQ – *AMuSe* based on LQ.

(ii) AMuSe-PDR – *AMuSe* based on PDR.

(iii) AMuSe-Mix – *AMuSe* based on mix of LQ and PDR.

(iv) K-worst-LQ – K nodes with lowest LQ are FB nodes.

(v) K-worst-PDR – K nodes with lowest PDR are FB nodes.

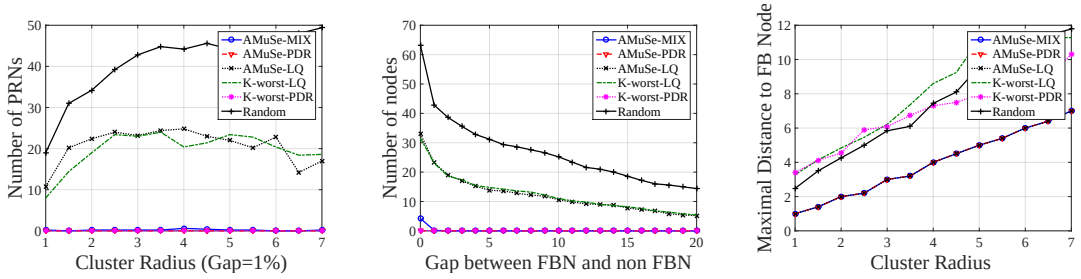
(vi) K-random – K random nodes as FB nodes.

The AMuSe-Mix scheme relies on lexicographic ordering of PDR and LQ values for comparing channel quality. For nodes with PDR $> 98\%$, the ordering is based on LQ. For nodes with PDR $\leq 98\%$, the ordering is based on PDR. Thus, the channel quality is defined by the following tuple in lexicographic order: $(\min(PDR, 98), LQ)$ The motivation behind AMuSe-Mix lies in our observation that LQ is weakly correlated with PDR in Section 2.6. Very high PDR values ($> 98\%$) could result from random packet losses and small PDR variations above this value are unreliable indicators of difference in channel quality. Thus, we use AMuSe-Mix to study if LQ can be a better metric to distinguish nodes which have high PDR values.

Moreover, we study the parameter choices for cluster radius (represented by the adjacency parameter, D). When we refer to cluster radius D as a parameter for the Random, K-worst-LQ, or K-worst-PDR schemes, we select as many FB nodes as *AMuSe* feedback schemes have (for a fair comparison).

We study the performance of different feedback nodes selection schemes under two network settings:

- **Static Settings:** The multicast bitrate and the external interference level are fixed.
- **Dynamic Settings:** In a dynamic environment of either (i) changing multicast bitrate, (ii) changing external interference, or (iii) emulated mobility.



(a) (b) (c)

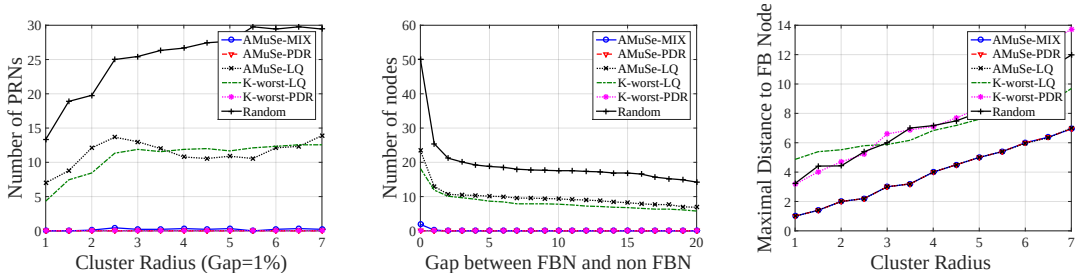
Figure 2.10: Static settings with bitrate of 48Mbps: (a) the number of Poorly Represented Nodes (PRN) vs. the cluster radius with fixed PRN-Gap of 1%, (b) PRN for different PRN-Gap and fixed cluster size of $D = 3$ m, and (c) maximal distance between an FB and non-FB node for various cluster radius.

For all our evaluations in both the static as well as the dynamic settings, we collected detailed packet traces at each node in the testbed for several bitrate and interference conditions. The number of nodes in the experiments was kept similar between 170 to 200 to avoid any performance mismatch. All the results for varying bitrate conditions were averaged over five runs of 10s at each bitrate. We ensured the appropriate setting of controlled interference by measuring the interference on a spectrum-analyzer on the testbed. During our experiments we observed sporadic spikes of uncontrolled interference. For mitigating their impact, we consider only time instants when there was no uncontrolled noise in our evaluations.

2.7.1 Static Settings

We first study the performance of different feedback schemes while the multicast bitrate and the generated external noise level are fixed. This setting allows us to evaluate the various schemes under normal network operation in stable conditions. We repeat our experiments with different bitrates and noise levels. We present our results for 3 different cases.

(i) **Fixed bitrate of 36 Mbps** – The optimal bitrate at which most of the nodes experience PDR close to 100 and only a few nodes suffer from low PDRs, as shown in Fig. 2.7(c).



(a) (b) (c)

Figure 2.11: Static settings with external noise: (a) the number of Poorly Represented Nodes (PRN) vs. the cluster radius with fixed PRN-Gap of 1%, (b) PRN for different PRN-Gap and fixed cluster size of $D = 3$ m, and (c) maximal distance between an FB and non-FB node for various cluster radius.

(ii) **Fixed bitrate of** 48 **Mbps** – Above the optimal bitrate many nodes experience low PDR, as shown in Fig. 2.7(f).

(iii) **External Noise** – The bitrate is set to 12 Mbps and the receivers suffer from different interference levels between -70dBm to -35dBm. The interference is concentrated on one corner of the grid as in Section 2.7.1.

The results of our evaluation are presented in Figs. 2.10-2.11. Figs. 2.10(a) and 2.11(a) show the number of PRNs as the cluster radius D increases at bitrate 48Mbps without external noise and at bitrate of 12Mbps wit external noise respectively. We only show the nodes with minimum PRN-Gap of 1% to avoid counting non-FB nodes with PDRs lower than their associated FB nodes by a small margin as PRN. Both AMuSe-Mix and AMuSe-PDR yield close to 0 PRNs since both schemes select nodes with lowest PDR in each cluster. K-worst-PDR also yields 0 PRNs, since it selects nodes with overall lowest PDR values. The link quality based schemes AMuSe-LQ and K-worst-LQ have similar performance which could be explained due to the weak correlation between LQ and PDR. As expected, the Random feedback selection scheme performs the worst and as the number of feedback nodes decreases (increase in cluster size), the number of PRNs increases due to fewer feedback nodes. We omit the results at lower bitrates since they are qualitatively similar but yield fewer overall PRNs since the vast majority of the nodes experience PDR above 99%. The

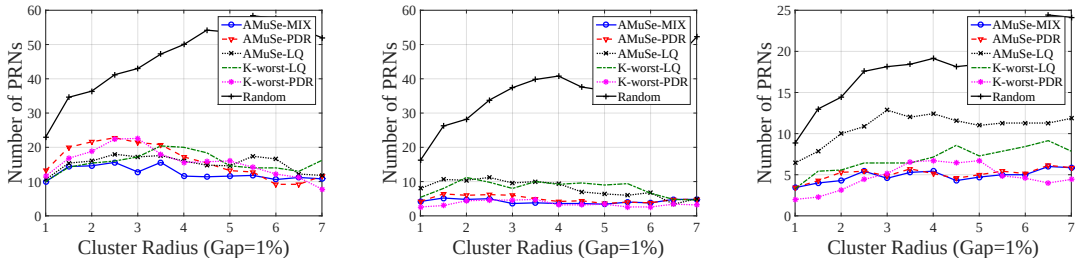
Random scheme yields much higher number of PRNs that increases with the cluster radius.

Figs. 2.10(b) and 2.11(b) present the number of PRNs at different values of PRN-Gap at bitrate 48Mbps without external noise and at bitrate of 12Mbps wit external noise respectively. The Random, K-worst-LQ, and AMuSe-LQ schemes result in a considerable number of PRNs. This number is high even for a PRN-Gap of 20% (e.g., Fig. 2.10(b) and 2.11(b) show that the K-worst-LQ and AMuSe-LQ schemes have between 5 to 10 PRNs with PRN-Gap of 20%). This means that the PDR value of each one of these nodes is at least 20% lower than its representative FB node. The situation is even worse for the Random scheme. We again omit the results at lower bitrates due to very low number of PRNs.

Finally, Figs. 2.10(c) and 2.11(c) show the maximum distance between an FB and non-FB node as D increases at bitrate 48Mbps without external noise and at bitrate of 12Mbps wit external noise respectively. As expected, for *AMuSe* schemes, this distance scales linearly with D. The maximum distance between an FB and non-FB node is significantly higher for the Random scheme and it is about twice for the K-worst-LQ and K-worst-PDR schemes. This indicates that FB nodes might be concentrated in areas of high losses. Thus, even though K-worst-PDR scheme leads to low number of PRNs, it does not obtain a well-distributed set of FB nodes. The distribution of FB nodes could be especially important in case of rapid network changes.

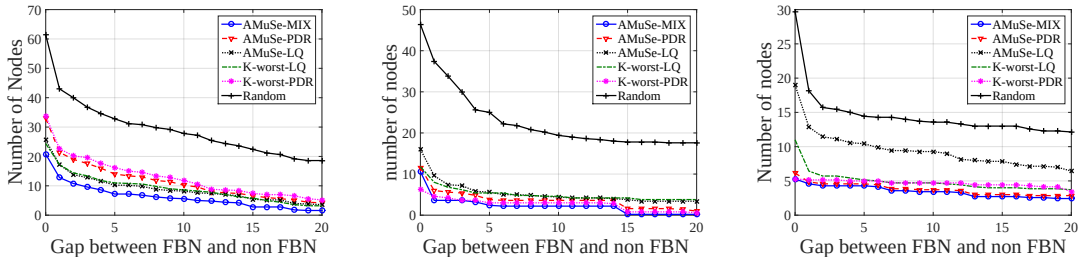
2.7.2 Dynamic Settings

Next, we emulate a dynamic environment of either: (i) changing AP bitrate, (ii) changing external interference, (iii) emulating node mobility. The methodology of the dynamic evaluations of (i) and (ii) relies on selecting a feedback set at one bitrate or external interference value and studying the performance of that set at a different value of bitrate or interference. Since the ORBIT environment is relatively static, we emulate mobility by exchanging positions of nodes but keeping their channel quality values fixed. The FB nodes are selected at a particular setting and a fixed percentage of non-FB nodes exchange locations with each other within a certain radius. The PRNs are then evaluated with the same FB nodes and clustering as the initial conditions. The dynamic setting helps to evaluate the performance



(a) Switching from 36 to 48 Mbps (b) Switching from 48 to 54 Mbps (c) Increasing noise by 10 dB

Figure 2.12: Dynamic Settings: The number of Poorly Represented Nodes (PRN) vs. the cluster radius with fixed PRN-Gap of 1%.



(a) Switching from 36 to 48 Mbps (b) Switching from 48 to 54 Mbps (c) Increasing noise by 10 dB

Figure 2.13: Dynamic Settings: The number of Poorly Represented Nodes (PRN) for different PRN-Gap and fixed cluster size of $D = 3$ m.

of the considered schemes under changes in the network.

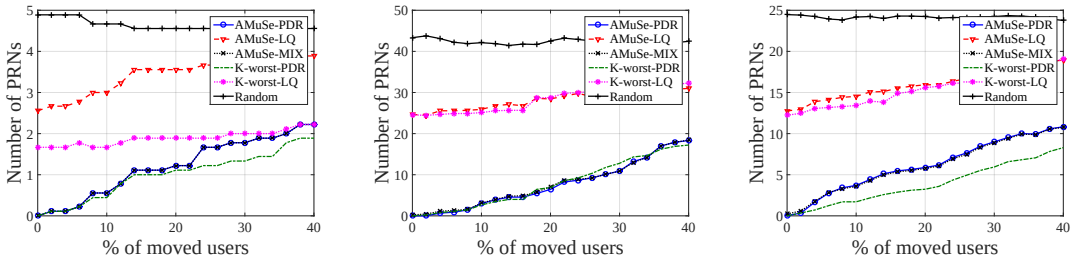
Obviously, under such dynamic changes, the feedback node selection process may choose a new set of FB nodes. However, this process may require noticeable convergence time (depending on several parameters, such as τ_{AP} and τ_{FB}) of up to a few seconds. During this time the system may not receive accurate reports about the service quality. Thus, it is essential that the selected FB nodes continue to provide accurate FB reports in the event of such changes. For instance, during any interference episode, the AP should receive the accurate feedback information without delays to take appropriate interference mitigation actions, such as adding more FEC, reducing bitrate, etc. Similarly, if the AP increases the multicast bitrate using a rate adaptation algorithm, the FB nodes should provide accurate state information about the change to the AP. For the dynamic setting we consider the

following cases: (a) Switching from bitrate of 36 Mbps to 48 Mbps, (b) Switching from bitrate of 48 Mbps to 54 Mbps, (c) Increasing the noise level by 10 dB, and (d) Emulated mobility.

Fig. 2.12 presents the number of PRNs vs. the cluster radius (D) for the three cases where the PRN-Gap is 1%. Fig. 2.12(a) shows the number of PRNs when switching the bitrate from 36 to 48 Mbps. In this case, the AMuSe-LQ and K-worst-LQ have comparable performance to the static case with bitrate of 48 Mbps. This is an expected result since LQ is a measure of the received signal strength and is not affected from changing the bitrate. However, AMuSe-PDR performs significantly worse than the static case. To understand this trend, recall that at bitrate of 36 Mbps most of receivers experience PDR close to 100%, as shown in Fig. 2.7(c). Therefore, when the cluster size is small and large number of receivers are selected as FB nodes, most of the FB nodes have PDR above 99%. With such high PDR, a selected FB node may not be affected by increasing the bitrate. Observe that the number of PRNs decreases by increasing the cluster size. This is not surprising since now most of the selected FB nodes have PDR below 98%, which indicates that they experience only moderate channel quality and therefore they are more susceptible to a bitrate increase. A similar explanation holds true for the K-worst-PDR scheme. AMuSe-Mix outperforms the other schemes since it considers both the PDR and the LQ of the receivers and uses the LQ values when the PDR is very high. Like the static setting, the Random scheme suffers from very high number of PRNs.

Fig. 2.12(b) shows the number of PRNs for bitrate increases from 48 to 54Mbps. In this case AMuSe-Mix , AMuSe-PDR , and K-worst-PDR outperform the LQ based solutions. By revisiting Fig. 2.7(f), we see that many receivers suffer from low PDR due to a weak channel condition at a bitrate of 48 Mbps. Since these nodes are selected as FB nodes, they provide good lower bound reports of the service quality observed by the nodes in their clusters. We notice a similar situation in Fig. 2.12(c) when increasing the noise level by 10 dB.

The distribution of PRNs vs the PRN-Gap is shown in Fig. 2.13 for a cluster radius $D = 3$ m. The figure supports our observations from Fig. 2.12 and demonstrates that



(a) (b) (c)

Figure 2.14: The number of Poorly Represented Nodes (PRNs) vs. percentage of moved nodes for (a) fixed bitrate of 36Mbps, (b) fixed bitrate of 48Mbps, and (c) bitrate of 12Mbps and noise of 5dBm.

AMuSe-Mix outperforms the other alternatives in all cases. Since the feedback node set is not changed when increasing the bitrate or noise level, the maximum distance between an FB and non-FB node remains the same as shown in Figs. 2.10(c) or 2.11(c).

The results for emulated node mobility are shown in Fig. 2.14. Fig. 2.14(a) shows the number of PRNs vs. the percentage of moved nodes within a radius of 2m at a fixed bitrate of 36Mbps. Similar results at bitrate of 48Mbps are in Fig. 2.14(b) and with external noise in Fig. 2.14(c). The Random scheme yields the largest number of PRNs and is not affected by increasing number of moved nodes. The AMuSe-Mix, AMuSe-PDR, and K-worst-PDR schemes perform quite similarly and the PRNs for all of them increase with increase in the number of moved nodes. The LQ based schemes AMuSe-LQ and K-worst-LQ perform worse than the PDR based schemes.

We also evaluate the sensitivity of *AMuSe* to errors in node location estimation by injecting errors into reported node locations. The errors are picked from a Gaussian distribution with $\mu = 0, \sigma = 7$ meters. However, we observed only insignificant increases in the number of PRNs for the *AMuSe* schemes.

Our experiments on the ORBIT testbed with approximately 200 nodes validate the practicality of AMuSe-Mix as an excellent scheme for reporting the provided quality of an ongoing WiFi multicast services for both static and dynamic settings. The K-worst-PDR scheme also peforms quite well but does not yield a well-distributed set of FB nodes. Our evaluation

shows that a relative small number of FB nodes is sufficient to provide accurate reports. Yet, the number of required FB nodes will also depend on the application.

2.A Proof of Proposition 1

Proposition 1: $|F| \leq 5 \cdot |OPT|$. If the channel quality is a monotonic decreasing function with the distance from the AP then $|F| \leq 3 \cdot |OPT|$

Proof of Proposition 1. We prove the general proposition of $|F| \leq 5 \cdot |OPT|$, which is based on Lemma 3.1 in [150]. The special case of $|F| \leq 3 \cdot |OPT|$, where the channel quality is a monotonic decreasing function with the distance from the AP, can be proved by using similar arguments and Lemma 3.3 in [150].

Consider a point x in the plane and let Z be an independent set of points in the circle with radius r around point x. i.e, the distance between any two points in Z is more than r. Then according to Lemma 3.1 in [150], $|Z| \leq 5$.

To prove that *AMuSe* guarantees approximation ratio of 5, we just need to show that for any given multicast group there is a mapping from F to OPT such that at most 5 nodes in F are mapped to the same node in OPT. To this end, we map every FB node $v \in F$ to its nearest node $u \in OPT$, which may be node v itself. Recall that both OPT and F are dominating independent sets such that each node has an adjacent FB node with distance at most D and the minimal distance between any pair of FB nodes is at least D. From this it is implied that any FB node v is either in OPT or it is D-adjacent to at least one node in OPT.

Now, consider an FB node $u \in OPT$ and let $W \subseteq F$ be the set of FB nodes selected by our scheme that are D-adjacent to u. Since F is an independent set it holds that W is also an independent set, i.e., the minimal distance between any pair of FB nodes $x, y \in W$ is $d_{x,y} > D$. Observe that all nodes in W are included in a disk with radius D centered at node u. Thus, according to Lemma 3.1 in [150], it follows that $|W| \leq 5$. This leads to the result that each node in OPT is associated with at most 5 nodes in F. □

Chapter 3

MULTICAST DYNAMIC RATE ADAPTATION

3.1 Introduction

As described in Chapter 2, improving WiFi multicast performance requires a scheme that *dynamically adapts the transmission rate* [152]. Yet, designing such a scheme poses several challenges, as outlined below.

Multicast Rate Adaptation (RA) - Challenges: A key challenge in designing multicast RA schemes for large groups is to obtain accurate quality reports with low overhead. Some systems [67, 187, 221] experimentally demonstrated impressive ability to deliver video to a few dozen nodes by utilizing Forward Error Correction (FEC) codes and retransmissions. However, most approaches do not scale to very large groups with hundreds of nodes, due to the following:

(i) Most schemes tune the rate to satisfy the receiver with the worst channel condition. As shown in [50, 161] in crowded venues, a few unpredictable outliers, referred to as *abnormal nodes*, may suffer from low SNR and Packet Delivery Ratio (PDR) even at the lowest rate and without interference. This results from effects such as multipath and fast fading [172]. Therefore, *a multicast scheme cannot provide high rate while ensuring reliable delivery to*

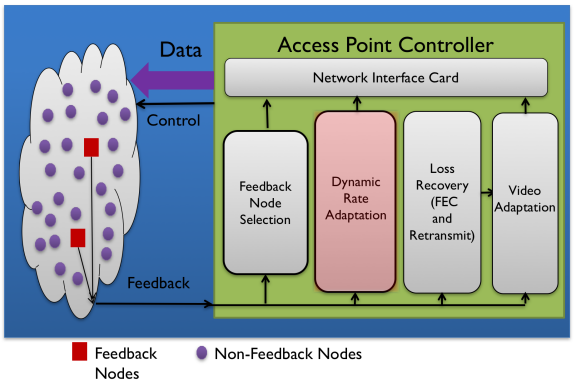


Figure 3.1: The Adaptive Multicast Services (*AMuSe*) system consisting of the Multicast Dynamic Rate Adaptation (MuDRA) algorithm and a multicast feedback mechanism.

all users.

(ii) It is impractical to continuously collect status reports from all or most users without hindering performance. Even if feedback is not collected continuously, a swarm of retransmission requests may be sent following an interference event, (wireless interference is bursty [32]) thereby causing additional interruptions.

To overcome these challenges, a multicast system should *conduct efficient RA based on only limited reports from the nodes.* In the previous chapter we focused on efficient feedback collection mechanisms for WiFi multicast as part of the *AMuSe* system. In this chapter, we present the Multicast Dynamic Rate Adaptation (*MuDRA*) algorithm. *MuDRA* leverages the efficient multicast feedback collection of *AMuSe* and *dynamically adapts the multicast transmission rate to maximize channel utilization while meeting performance requirements.* Fig. 3.1 shows the overall *AMuSe* system composed of the *MuDRA* algorithm and the *AMuSe* feedback mechanism where the focus of this chapter is *MuDRA* .

3.1.1 Our Contributions

We present a multicast rate adaptation algorithm *MuDRA* which is designed to support WiFi multicast to hundreds of users in crowded venues. *MuDRA* can provide high throughput while ensuring high *Quality of Experience* (QoE). *MuDRA* benefits from a large user population, which allows selecting a small yet sufficient number of Feedback (FB) nodes

with marginal channel conditions for monitoring the quality. We address several design challenges related to appropriate configuration of the feedback level.

We note that using *MuDRA* does not require any modifications to the IEEE 802.11 standard or the mobile devices. *MuDRA* requires application layer measurements from mobile devices for multicast rate adaptation decisions. The multicast rate changes can be supported by most Access Points through changes in the driver-level code or through API calls.

We implemented *MuDRA* with the *AMuSe* system on the ORBIT testbed [14], evaluated its performance with all the operational IEEE 802.11 nodes (between 150 and 200), and compared it to other multicast schemes. We use 802.11a to maximize the number of WiFi devices available[1]. To the best of our knowledge, this is the largest set of wireless devices available to the research community. Our key contributions are:

(i) The need for RA: We empirically demonstrate the importance of RA. Our experiments on ORBIT show that when the multicast rate exceeds an optimal rate, termed as *target-rate*, numerous receivers suffer from low PDR and their losses cannot be recovered. We also observed that even a controlled environment, such as ORBIT, can suffer from significant interference. These observations constitute the need for a stable and interference agnostic RA algorithm that *does not exceed the target-rate*.

(ii) Practical method to detect the target-rate: Pseudo-multicast schemes that rely on unicast RA [67] may occasionally sample higher rates and retreat to a lower rate after a few failures. Based on the observation above about the target rate, schemes with such sampling mechanisms will provide low QoE to many users. To overcome this, we developed a method to detect when the system operates at the target-rate, termed the **target condition**. Although the target condition is sufficient but not necessary, our experiments show that it is almost always satisfied when transmitting at the target-rate. *MuDRA* makes RA decisions based on the target condition and employs a dynamic window based mechanism to avoid rate changes due to small interference bursts.

[1]The ORBIT testbed supports only about 30 802.11n enabled devices

(iii) Extensive experiments with hundreds of receivers: Our experiments demonstrate that *MuDRA* swiftly converges to the target-rate, while meeting the Service Level Agreement (SLA) requirements (e.g., ensuring PDR above 85% to at least 95% of the nodes). Losses can be recovered by using appropriate application-level FEC methods [31, 71, 162, 189, 210].

MuDRA is experimentally compared to (i) pseudo-multicast with a unicast RA [10], (ii) fixed rate, and (iii) a rate adaptation mechanism proposed in [50] which we refer to as the Simple Rate Adaptation (SRA) algorithm. *MuDRA* achieves 2x higher throughput than pseudo-multicast while sacrificing PDR only at a few poorly performing nodes. While the fixed rate and SRA schemes can obtain similar throughput as *MuDRA*, they do not meet the SLA requirements. Unlike other schemes, *MuDRA* preserves high throughput even in the presence of interference. Additionally, *MuDRA* can handle significant node churn. Finally, we devise a live multicast video delivery approach for *MuDRA*. We show that in our experimental settings with target rate of $24-36$Mbps, *MuDRA* can deliver 3 or 4 high definition H.264 videos (each one of 4Mbps) where over 90% of the nodes receive video quality that is classified as excellent or good based on user perception.

To summarize, to the best of our knowledge, *MuDRA* is the first multicast RA algorithm designed to satisfy the specific needs of multimedia/video distribution in crowded venues. Moreover, *AMuSe* in conjunction with *MuDRA* is the first multicast content delivery system that has been evaluated *at scale*. The rest of the chapter is organized as follows. Section 3.3 describes the ORBIT testbed and important observations. Section 3.4 presents the model and objectives. *MuDRA*'s design is described in Sections 3.5 and 3.6. The experimental evaluation is presented in Section 3.7.

The design and experimental evaluation of *MuDRA* appeared in the proceedings of IEEE INFOCOM'16 [99]. A technical report can be found in [100] and an extended version was submitted to a journal. Yigal Bejerano and Craig Gutterman contributed to the design behind *MuDRA*. A demo of the rate adaptation process was presented at and appeared in the proceedings of IEEE INFOCOM'16 [103] with significant contributions from Raphael Norwitz and Savvas Petridis.

3.2 Related Work

Multicast rate adaptation approaches are in general closely linked to multicast feedback. In Chapter 2, we described existing approaches for multicast feedback mechanisms in detail. In this chapter, we focus on unicast and multicast te adaptation.

Unicast RA: We discuss unicast RA schemes, since they can provide insight into the design of multicast RA. In *Sampling-based algorithms*, both ACKs after successful transmissions and the relation between the rate and the success probability are used for RA after several consecutive successful or failed transmissions [59, 122, 132]. The schemes in [128, 160, 219] distinguish between losses due to poor channel conditions and collisions, and update the rate based on former. Recently, [76, 169] propose multi-arm bandit based RA schemes with a statistical bound on the regret. However, such schemes cannot support multicast, since multicast packets are not acknowledged. In *Measurement-based schemes* the receiver reports the channel quality to the sender which determines the rate [79,110,119,170,173,211]. Most measurement-based schemes modify the wireless driver on the receiver end and some require changes to the standard, which we avoid.

Multicast RA: In [48, 67, 145, 188, 208] the sender uses feedback from leaders (nodes with worst channel conditions) for RA. In [144] when the channel conditions are stable, RA is conducted based on reports of a single leader. When the channel conditions are dynamic, feedback is collected from all nodes. Medusa [187] combines Pseudo-Multicast with infrequent application layer feedback reports from all nodes. The MAC layer feedback sets backoff parameters while application layer feedback is used for RA and retransmissions of video packets. Recently, in [50] we considered multicast to a large set of nodes and provided a rudimentary RA scheme which is not designed to achieve optimal rate, maintain stability, or respond to interference.

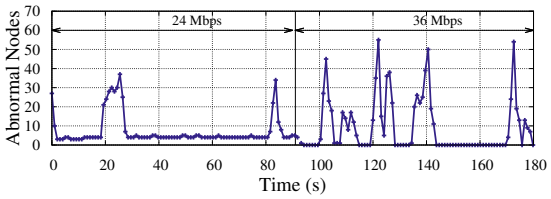


Figure 3.2: Experimental measurement of the number of abnormal nodes in time, for fixed rates of 24 and 36Mbps.

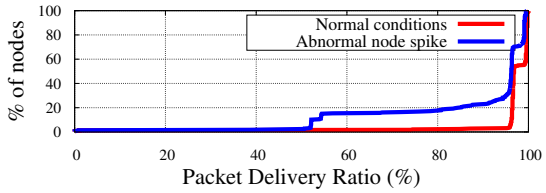


Figure 3.3: The CDF of the PDR values of 170 nodes during normal operation and during a spike at rate of 36Mbps.

3.3 Testbed and Key Observations

We evaluate *MuDRA* on the ORBIT testbed [14], which is a dynamically configurable grid of 20×20 (400) 802.11 nodes where the separation between nodes is 1m. It is a good environment to evaluate *MuDRA*, since it provides a very large and dense population of wireless nodes, similar to the anticipated crowded venues.

Experiments: To avoid performance variability due to a mismatch of WiFi hardware and software, only nodes equipped with *Atheros 5212/5213* cards with *ath5k* driver were selected. For each experiment *we activated all the operational nodes* that meet these specifications (between 150 and 250 nodes). In all the experiments, one corner node served as a single multicast AP. The other nodes were multicast receivers. The AP used 802.11a to send a multicast UDP flow, where each packet was 1400 bytes. Most practical applications such as video streaming include a sequence number to keep track of packet delivery at the clients. We embed an artificial sequence number for each packet in the UDP payload for measurement purposes. *The AP used the lowest supported transmission power of* $1mW = 0dBm$ *to ensure that the channel conditions of some nodes are marginal.*

Technical challenges: While analyzing the performance, we noticed that clients disconnect from the AP at high bit-rates, thereby causing performance degradation. We noticed that in several WiFi driver implementations, the beacon rate is set as the multicast rate. Increasing the bit-rate also increases the WiFi beacon bit-rate which may not be decoded at some nodes. A sustained loss of beacons leads to node disconnection. To counter this, we modified the ath5k driver to send beacons at a fixed minimum bit-rate.

Interference and Stability: We study the time variability of the channel conditions on the ORBIT testbed by measuring the number of nodes with low PDR (below a threshold of 85%). We call these nodes *abnormal nodes* (the term will be formally defined in Section 3.4). The number of abnormal nodes out of 170 nodes for rates of 24 and 36Mbps is shown in Fig. 3.2. We repeated these experiments several times and observed that even at a low rate, the channel may suffer from sporadic interference events, which cause a sharp increase in the number of abnormal nodes. These interference spikes caused by non-WiFi devices are beyond our control and their duration varies in time.

Fig. 3.3 provides the Cumulative Distribution Function (CDF) of the PDR values with and without sporadic interference. The figure shows that during a spike, over 15% of the nodes suffer from PDR around 50%. Further, the location of the nodes affected by the spikes varies with time and does not follow a known pattern. These experiments show that even in a seemingly controlled environment, *nodes may suffer from sporadic continuous interference, which may cause multicast rate fluctuations.* Users are very sensitive to changes in video quality [46, 78], and therefore, to keep a high QoE we would like to avoid rate changes due to sporadic interference.

3.4 Network Model and Objective

We consider a WiFi LAN with multiple APs and frequency planning such that the transmissions of adjacent APs do not interfere with each other. Thus, *for RA we consider a single AP with n associated users.* We assume low mobility (e.g., users watching a sports event). Although we consider a controlled environment, the network may still suffer from

Table 3.1: Notation and parameter values used in experiments.

Symbol	**Semantics**	**Exp. Val.**
n	Number of nodes associated with the AP.	> 150
X	*Population threshold* - Minimal fraction of nodes that should experience high PDR.	95%
A_{max}	The maximal number of allowed abnormal nodes.	8
L	*PDR threshold* - Threshold between acceptable (normal) and low (abnormal) PDR.	85%
H	Threshold between high PDR and mid-PDR.	97%
K	Expected number of FB nodes, $K = \alpha \cdot A_{max}$.	30
R	Reporting PDR threshold.	
A_t	Number of abnormal nodes at time t.	
M_t	Number of mid-PDR FB nodes at time t.	
W_{min}	Minimal RA window size (multiples of reporting intervals).	8
W_{max}	Maximal RA window size.	32

sporadic interference, as shown in Section 3.3. The main notation used in the chapter is summarized in Table 3.1. Specifically, a *PDR-Threshold* L, is defined such that a node has high QoE if its PDR is above L. Such a node is called a *normal node*. Otherwise, it is considered an *abnormal node*.

Our *objective is to develop a practical and efficient rate control system* which satisfies the following requirements:

(R1) High throughput – Operate at the highest possible rate, i.e., the *target rate*, while preserving SLAs.

(R2) Service Level Agreements (SLAs) – Given L (e.g., $L = 85\%$), and a *Population-Threshold* X (e.g., $X = 95\%$), the selected rate should guarantee that at least $X\%$ of the nodes experience PDR above L (i.e., are normal nodes). Except for short transition periods, this provides an upper bound of $A_{max} = \lceil n \cdot (1 - X) \rceil$ on the number of permitted abnormal nodes.

(R3) Scalability – Support hundreds of nodes.

(R4) Stability – Avoid rate changes due to sporadic channel condition changes.

(R5) Fast Convergence – Converge fast to the target rate after long-lasting changes (e.g., user mobility or network changes).

(R6) Standard and Technology Compliance – No change to the IEEE 802.11 standard or operating system of the nodes.

3.5 Multicast Rate Adaptation

The overall multicast rate adaptation process of *MuDRA* as a part of the *AMuSe* system relies on three main components, as illustrated in Fig. 3.1 and discussed below. We first provide a high level description of each component and then discuss the details in the following subsections.

(i) Feedback (FB) Node Selection: Selects a small set of *FB nodes* that provide reports for making RA decisions. We describe the FB node selection process in Section 3.5.1 and

Algorithm 1 *MuDRA* Algorithm

1: $rate \leftarrow lowestRate$, $window \leftarrow W_{min}$, $changeTime \leftarrow t$, $refTime \leftarrow t, t :=$ current time
2: **while** $(true)$ **do**
3: Get PDR reports from all FB nodes
4: Get Status of each FB node i
5: Calc $\hat{A}_t$ and $\hat{M}_t$
6: $rate, action, changeTime \leftarrow GetRate(...)$
7: $window, refTime \leftarrow GetWinSize(...)$
8: set multicast rate to $rate$
9: sleep one reporting interval

calculate the reporting interval duration in Section 3.6.[2]

The following two components compose the *MuDRA* Algorithm (Algorithm 1). It collects the PDR values from the FB nodes, updates their status (normal or abnormal), invokes the GetRate procedure, which calculates the desired rate, and invokes the GetWinSize procedure, which determines the window size of rate updates (to maintain stability).

(ii) Rate Decision (Procedure 1): Utilizes the limited and infrequent FB reports to determine the highest possible rate, termed the *target-rate*, while meeting the requirements in Section 3.4. The rate decisions (lines 5–15) rely on rate decision rules that are described in Section 3.5.2. To maintain rate stability, rate change operations are permitted, only if the conditions for rate change are satisfied for time equal to a window size (determined by the *Stability Preserving Method*).

(iii) Stability Preserving Method (Procedure 2): A window based method that maintains rate stability in the event of sporadic interference and after an RA decision. It follows the classical Additive Increase Multiplicative Decrease (AIMD) approach. The duration of the time window varies according to the network and channel characteristics (e.g., the typical duration of interference). More details appear in Section 3.5.3.

[2]Unlike in unicast where each packet is acknowledged, *MuDRA*'s reporting intervals are long (in the experiments we consider 2 reports per second).

Procedure 1 Rate Decision

```
1:  procedure GetRate(rate, window, changeTime, t)
2:      action ← Hold
3:      if (t − changeTime) > window then
4:          canDecrease ← true, canIncrease ← true
5:          for τ ← 0 to window do
6:              if Â_{t−τ} < A_max then
7:                  canDecrease ← false
8:              else if Â_{t−τ} + M̂_{t−τ} > A_max − ϵ then
9:                  canIncrease ← false
10:         if canDecrease and rate > rate_min then
11:             rate ← NextLowerRate
12:             action ← Decrease, changeTime ← t
13:         if canIncrease and rate < rate_max then
14:             rate ← NextHigherRate
15:             action ← Increase, changeTime ← t
16: return rate, action, changeTime
```

3.5.1 Feedback Node Selection

MuDRA uses a simple and efficient mechanism based on a quasi-distributed FB node selection process, termed *K-Worst* [50], where the AP sets the number of FB nodes and their reporting rates. K nodes with the worst channel conditions are selected as FB nodes (the node's channel condition is determined by its PDR). Hence, the selection process ensures an upper bound on the number of FB messages, regardless of the multicast group size. This upper bound is required for limiting the interference from FB reports, as explained in Section 3.6. The process works as follows: At the beginning of each reporting interval the AP sends a message with a list of K or less FB nodes as well as a *reporting PDR threshold* R. R is used for adjusting the set of FB nodes to changes due to mobility or variation of the channel condition, i.e., interference[3]. Upon receiving this message, each FB node waits a short random time for avoiding collisions and then reports its measured PDR to the AP.

[3] when the system is activated the FB list is empty and $R = L$.

Procedure 2 Window Size Determination

1: **procedure** GETWINSIZE($Action, window, refTime, t$)
2: **if** $Action = Decrease$ **then**
3: $window \leftarrow \min(W_{max}, 2 \cdot window)$, $refTime \leftarrow t$
4: **else if** $Action = Increase$ **then**
5: $refTime \leftarrow t$
6: **else if** $(t - refTime) > thresholdTime$
7: and $Action = Hold$ **then**
8: $window \leftarrow \max(W_{min}, window - 1)$
9: $refTime \leftarrow t$
10: **return** $window, refTime$

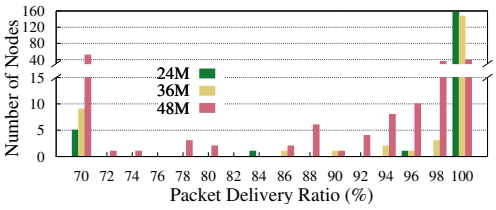


Figure 3.4: The PDR distribution of one set of experiments with TX_{AP} rates of 24, 36, and 48Mbps.

Every other node checks if its PDR value is below R and in such situation it volunteers to serve as an FB node. To avoid a swarm of volunteering messages in the case of sporadic interference, a non FB node verifies that its PDR values are below R for three consecutive reporting intervals before volunteering. At the end of a reporting interval, the AP checks the PDR values of all the FB and volunteering nodes, it selects the K with lowest PDR values as FB nodes and updates R. If the number of selected FB nodes is K then for keeping the stability of the FB list, R is set slightly below the highest PDR value of the FB nodes (e.g., 1% point below) . Otherwise, R is set slightly above the highest PDR value of the FB nodes (e.g., 0.5% point above). The AP sends a new message and the process repeats. We note that in a quasi static scenario, the values of R do not have a significant impact on the feedback or the overhead of feedback. Tuning R is a challenge only in the rare scenario when a large number of nodes with significantly different PDR values rapidly enter or leave the multicast system.

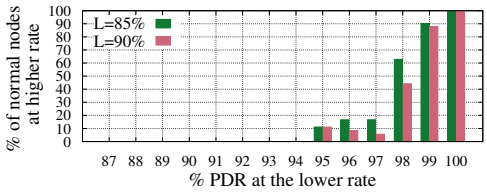


Figure 3.5: The percentage of nodes that remain normal after increasing the TX_{AP} from 36Mbps to 48Mbps vs. their PDR values at the 36Mbps for different PDR-thresholds (L).

3.5.2 Rate Decision Rules and Procedure

In this subsection, we describe the *target condition* which is an essential component of the rate selection rules. Then, we describe the rules and the corresponding Procedure 1.

The Target Condition: At a given time, the FB reports are available only for the current rate. To detect the target-rate, most RA schemes occasionally sample higher rates. However, the following experiment shows that this approach may cause undesired disruption to many receivers. We evaluated the PDR distribution of $160 - 170$ nodes for different multicast transmission rates, denoted as TX_{AP} for 3 different experiment runs on different days. Fig. 3.4 shows the number of nodes in different PDR ranges for TX_{AP} values of 24, 36, and 48Mbps for one experiment with 168 nodes. When TX_{AP} is at most 36Mbps, the number of abnormal nodes is very small (at most 5). However, when TX_{AP} exceeds 36Mbps, the PDR of many nodes drops significantly. In this experiment 47 nodes became abnormal nodes which is more than $A_{max} = 8$ (for $X = 95\%$). We observed similar results in other experiments. Thus, in this case, the *target rate* is 36Mbps which is the highest rate above which the SLA requirements will be violated.

A key challenge is to *determine if the AP operates at the target-rate, without FB reports from higher rates.* We refer to this assessment as the *target condition.* Unfortunately, the target-rate cannot be detected from RF measurements, such as SNR. As shown in [108,174] different nodes may have different receiver sensitivities, which may result in substantial PDR gaps between nodes with similar RF measurements. However, large scale multicast environments enable us to efficiently predict the target condition as described next.

From Fig. 3.4, we obtain the following important observation.

Observation I: When operating below the target-rate, almost all the nodes have PDR close to 100%. However, when operating at the target-rate, noticeable number of receivers experience PDR below 97%. At 36Mbps, 17 nodes had PDR below 97%, which is substantially more than $A_{max} = 8$.

Fig. 3.5 shows the average percentage of nodes that remain normal vs. their initial PDR when increasing TX_{AP} from 36Mbps to 48Mbps averaged for 3 different sets of experiments. The total number of nodes in these experiments was 168. We derive the following observation from Fig. 3.5.

Observation II: There is a PDR threshold, H, such that every node with PDR between L and H becomes abnormal after the rate increase with very high probability. Typically, H can be a value slightly below 100%. In our experiments on ORBIT, we use $H = 97\%$ since 97% is the highest threshold for which this observation holds. We refer to these nodes as *mid-PDR nodes*.

Observation II is not surprising. As reported in [174, 194], each receiver has an SNR band of $2 - 5$dB, in which its PDR drops from almost 100% to almost 0%. The SNR of mid-PDR nodes lies in this band. Increasing the rate requires $2 - 3$dB higher SNR at the nodes. Hence, mid-PDR nodes with SNR in the transition band before the rate increase will be below or at the lower end of the transition band after the increase, and therefore, become abnormal nodes.

In summary, Observations I and II imply that it is possible to assess the target condition by monitoring the nodes close to transitioning from normal to abnormal. Let A_t and M_t denote the number of abnormal and mid-PDR nodes at time t, respectively. We obtain the following empirical property.

Property 1 (Target Condition)**.** *Assume that at a given time t, the following condition holds,*

$$A_t \leq A_{max} \quad \textit{and} \quad A_t + M_t > A_{max} \tag{3.1}$$

then almost surely, the AP transmits on the target-rate at time t. This is sufficient but not

a necessary condition.

It is challenging to analytically predict when the target condition is satisfied with the available FB information and without a model of the receiver sensitivity of all the nodes. However, our experiments show that the target condition is typically valid when operating at the target-rate.

Adjusting the Multicast Rate: The SLA requirement (R2) and target condition (3.1) give us a clear criteria for changing the rate. The FB scheme only gives us estimates of A_t and M_t, denoted by $\hat{A}_t$ and $\hat{M}_t$ respectively. For the K-Worst scheme, if $K > A_{max} + \epsilon$ (ϵ is a small constant), then $\hat{A}_t$ and $\hat{M}_t$ are sufficient to verify if (3.1) is satisfied because of the following property:

Property 2. *If $K \geq A_{max} + \epsilon$, then, $\hat{A}_t = \min(A_t, A_{max} + \epsilon)$ and $\hat{A}_t + \hat{M}_t = min(A_t + M_t, A_{max} + \epsilon)$, where $\hat{A}_t$ and $\hat{M}_t$ are the known number of abnormal and mid-PDR known to the AP, and ϵ is a small constant. In other words, given that K is large enough, the K-worst scheme provides accurate estimates of abnormal and mid-PDR nodes.*

Proof. *First consider the claim $\hat{A}_t = \min(A_t, A_{max} + \epsilon)$. Consider the case $A_t \leq A_{max} + \epsilon$, we know that the number of estimated abnormal nodes $\hat{A}_t = A_t$ since $K \geq A_{max} + \epsilon$ and all abnormal nodes must belong in the K FB nodes set. Next, if $A_t > A_{max} + \epsilon$ then all the FB nodes chosen are abnormal by the definition of the K-worst feedback scheme which implies $\hat{A}_t = A_{max} + \epsilon$.*

A similar argument can be made for the claim $\hat{A}_t + \hat{M}_t = min(A_t + M_t, A_{max} + \epsilon)$. If $A_t + M_t \leq A_{max} + \epsilon$, then $\hat{A}_t + \hat{M}_t = A_t + M_t$ since the K feedback nodes will necessarily include the A_t abnormal and M_t mid-PDR nodes. If $A_t + M_t > A_{max} + \epsilon$, then $\hat{A}_t + \hat{M}_t$ which is upper bounded by $A_{max} + \epsilon$. □

The objective is to choose minimum K (for minimum FB overhead) that is sufficient to verify (3.1). In our experiments, we found that for $A_{max} = 8$, $K > 10$ works well (Section 3.7.1). We now derive the following *rate changing rules*:

Rule I $\hat{A}_t > A_{max}$: The system violates the SLA requirement (R2) and the rate is reduced.

Rule II $\hat{A}_t + \hat{M}_t \geq A_{max} - \epsilon$: The system satisfies the target condition.

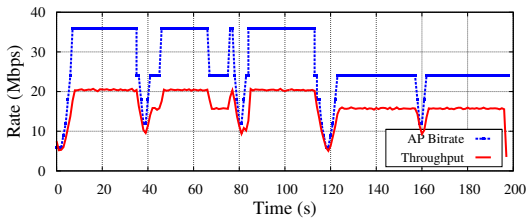


Figure 3.6: Evolution of the multicast rate over time when the delay between rate changes = 1s (2 reporting intervals).

.

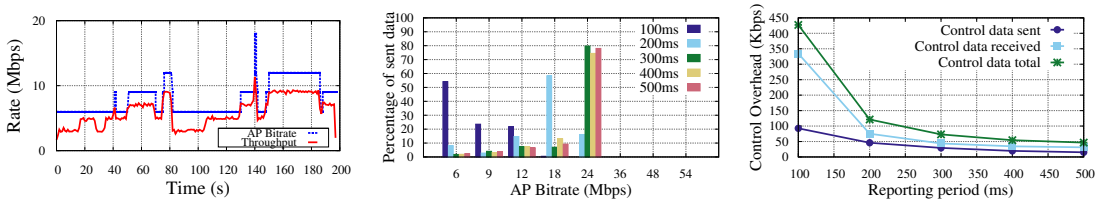
Rule III $\hat{A}_t + \hat{M}_t < A_{max} - \epsilon$: The target condition does not hold and the rate can be increased, under the stability constraints provided in Section 3.5.3.
In our experiments we use $\epsilon = 2$ to prevent rate oscillations.

The rate change actions in Procedure 1 are based on the these rules. The flags *canIncrease* and *canDecrease* indicate whether the multicast rate should be increased or decreased. Rate change operations are permitted, only if the time elapsed since the last rate change is larger than the window size determined by the *Stability Preserving Method* (line 3). The for-loop checks whether the rate should be decreased according to Rule I (line 6) or increased according to Rule III (line 9) for the window duration. Finally, based on the value of the flags and the current rate, the algorithm determines the rate change operation and updates the parameters *rate* and *action*, accordingly (lines 10–15).

3.5.3 The Stability Preserving Method

It is desirable to change the rate as soon as Rules I or III are satisfied to minimize QoE disruption (see (R5) in Section 3.4). However, as we show in Fig. 3.6 such a strategy may cause severe fluctuations of the transmission rate. These result from two main reasons: (i) the reporting mechanism not stabilizing after the last rate change, and (ii) interference causing numerous low PDR reports.

To address this, we introduce in Procedure 2 a *window based RA technique* which considers the two situations and balances fast convergence with stability. In Procedure 1, the rate is changed only if the rate change conditions are satisfied over a given *time window*,



(a) (b) (c)

Figure 3.7: (a) Rate adaptation performance for reporting intervals of 100ms, (b) Fraction of data sent at various rates with *MuDRA* for different reporting intervals, and (c) Control overhead for various reporting intervals.

after the last rate change operation (lines 5-9). To prevent oscillations due to short-term wireless channel degradation, when the rate is reduced, the window is doubled in Procedure 2 (line 3). The window size is decreased by 1 when a duration $thresholdTime$ elapses from the last rate or window size change (line 8). This allows recalibrating the window after an atypical long interference episode. The window duration varies between W_{min} and W_{max} FB reporting periods. In the experiments, $W_{min} = 8$ and $W_{max} = 32$.

3.5.4 Handling Losses

MuDRA can handle mild losses (below 15%) by adding application level FEC [210] to the multicast streams. The PDR-Threshold in our experiments ($L = 85\%$) was selected to allow nodes to handle losses in the event of short simultaneous transmission of another node. In such a situation, the collision probability is below $2/CW_{min}$, where CW_{min} is the minimal 802.11 contention window. For 802.11a/g/n $CW_{min} = 16$, which implies collision probability is below 12.5%. Therefore, nodes with high PDR (near 100%) should be able to compensate for the lost packets. If there is strong interference, other means should be used. For instance, the multicast content can be divided into high and low priority flows, augmenting the high priority flow with stronger FEC during the interference period, while postponing low priority flows.

Table 3.2: The percentage of PDR loss at nodes ($\Delta PDR(T)$) as a function the reporting interval T.

T (ms)	100	200	300	400	500	700	1000
$\Delta PDR\%$	4.69	1.56	0.94	0.67	0.52	0.36	0.25

3.6 Reporting Interval Duration

MuDRA relies on status reports from the FB nodes. For immediate response to changes in service quality, the status reports should be sent as frequently as possible, (i.e., minimal *reporting interval*). However, this significantly impairs the system performance as described below.

Impact of Aggressive Reporting: Figs. 3.7(a)-3.7(c) show the impact of different reporting intervals on *MuDRA*. In these experiments, the number of FB nodes (K) is 50 and the total number of nodes is 158. To focus on RA aspects, we set both W_{min} and W_{max} to 5 reporting intervals. Fig. 3.7(a) shows that when the reporting interval is too short, *MuDRA* does not converge to the target rate of 24Mbps. Fig. 3.7(b) shows that in the case of reporting interval of 100ms, more than 50% of the packets are transmitted at the lowest rate of 6 Mbps. Fig. 3.7(c) shows that the control overhead is significantly larger for short reporting intervals (shorter than 200ms). The control overhead comprises of unicast FB data sent by nodes and multicast data sent by AP to manage K FB nodes.

These phenomena result from collisions between feedback reports and multicast messages. In the event of a collision, FB reports, which are unicast messages, are retransmitted, while multicast messages are lost. Frequent reporting increases the collision probability, resulting in PDR reduction and causes the classification of many nodes as mid-PDR nodes, i.e., $PDR < H_{high} = 97\%$. Thus, due to Rule II from Section 3.5.2, the rate is kept close to the minimal rate.

Appropriate Reporting Interval Duration:

Assume a greedy AP which continuously transmits multicast messages. We now estimate the PDR reduction, denoted as ΔPDR, for a given reporting interval T and upper bound K on the number of FB nodes (both normal and abnormal), when the system operates at

the low rate of 6Mbps.

Packet Transmission Duration: We denote with D and d the transmission duration of multicast and feedback report message at the rate of 6Mbps, respectively. Since the length of each multicast packet is $12Kbits$, its transmission duration is $\frac{12\text{Kbits}}{6\text{Mbps}} = 2.0\text{ms}$. Given WiFi overhead of about 30%, we assume $D = 3\text{ms}$. The feedback messages are much shorter and we assume that their transmission duration is $d = 1\text{ms}$.

Number of feedback reports and multicast messages: Consider a time interval U, say a minute. The number of feedback reports, denoted as F, is

$$F = \frac{U}{T} \cdot K$$

The number of multicast message B is given by,

$$B = \frac{U - d \cdot F}{D} = \frac{U}{D} \cdot \left(1 - \frac{d \cdot K}{T}\right)$$

Collision probably of a multicast packet (ΔPDR): Let us first calculate the number of contention window slots, denoted by S, in which packet may be transmitted from the view point of the AP during the time interval U. Recall that between any two multicast transmissions, the AP waits an average of half of the contention window size $CW_{min}/2 = 8$. This leads to

$$S = \frac{CW_{min}}{2} \cdot B$$

ΔPDR is the fraction of slots in which both the AP and a FB node send a message. To simplify our estimation, we ignore collisions and retransmission of FB messages[4], and assume that in any slots only one FB node may transmit. Therefore,

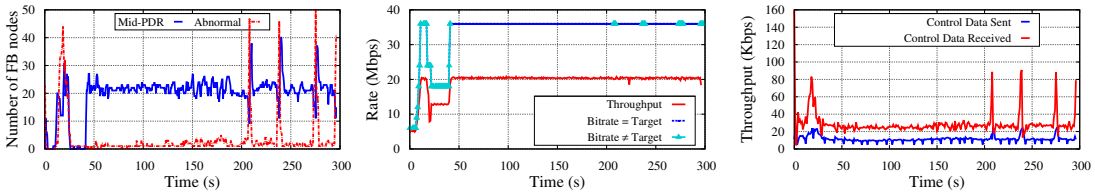
$$\Delta PDR = \frac{F}{S} \cdot \frac{B}{S} = \left[\frac{2}{CW_{min}}\right]^2 \cdot \frac{F}{B}$$

With proper assignment we get,

$$\Delta PDR = \left[\frac{2}{CW_{min}}\right]^2 \cdot \frac{K \cdot D}{T - d \cdot K} \tag{3.2}$$

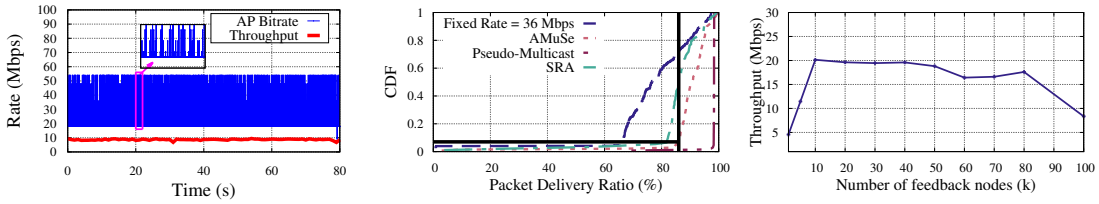
Equation (3.2) confirms that ΔPDR is reduced by increasing the reporting interval or by using a higher bit-rate, which reduces D. Table 3.2 provides the ΔPDR values for $K = 50$

[4]These are second order effects of already low collision probabilities.



(a) (b) (c)

Figure 3.8: A typical sample of *MuDRA*'s operation over 300s with 162 nodes: (a) Mid-PDR and abnormal nodes, (b) Multicast rate and throughput measured at the AP, and (c) Control data sent and received.



(a) (b) (c)

Figure 3.9: (a) Rate and throughput for the pseudo-multicast scheme, (b) CDF of PDR distributions of 162 nodes for fixed rate, *MuDRA*, Pseudo-Multicast, and SRA schemes, and (c) Multicast throughput vs. the number of feedback nodes (K).

when T varies between 0.1 to 1 second. In our experiments we wanted $\Delta PDR \leq 0.5\%$, which implies using reporting interval $T \geq 500$ms.

3.7 Experimental Evaluation

For evaluating the performance of *MuDRA* on the ORBIT testbed, we use the parameter values listed in Table 3.1. In all our evaluations, we consider backlogged multicast traffic. The performance metrics are described below:

(i) *Multicast rate and throughput*: The time instants when the target condition is satisfied are marked separately.

(ii) *PDR at nodes*: Measured at each node.

(iii) *Number of abnormal and mid-PDR nodes*: We monitored *all* the abnormal and mid-PDR nodes (not just the FB nodes).
(iv) *Control traffic*: The feedback overhead (this overhead is very low and is measured in Kbps).

We compared *MuDRA* to the following schemes:
(i) *Fixed rate scheme*: Transmit at a fixed rate of 36Mbps, since it is expected to be the target rate.
(ii) *Pseudo-multicast*: Unicast transmissions to the node with the lowest SNR/RSS. The unicast RA is the driver specific RA algorithm *Minstrel* [10]. The remaining nodes are configured in promiscuous mode.
(iii) *Simple Rate Adaptation (SRA) algorithm [50]*: This scheme also relies on measuring the number of abnormal nodes for making RA decisions. Yet, it is not designed to achieve the target rate, maintain stability, or respond to interference.

3.7.1 Performance Comparison

We evaluated the performance of *MuDRA* in several experiments on different days with $160-170$ nodes. Fig. 3.8 shows one instance of such an experiment over 300s with 162 nodes. Fig. 3.8(a) shows the mid-PDR and abnormal nodes for the duration of one experiment run. Fig. 3.8(b) shows the rate determined by *MuDRA*. The AP converges to the target rate after the initial interference spike in abnormal nodes at 15s. The AP successfully ignored the interference spikes at time instants of 210, 240, and 280s to maintain a stable rate. The target-condition is satisfied except during the spikes. The overall control overhead as seen in Fig. 3.8(c) is approximately 40Kbps. The population of abnormal nodes stays around $2-3$ for most of the time which implies that more than 160 nodes ($> 98\%$) have a PDR $> 85\%$. The actual throughput is stable at around 20Mbps which after accounting for 15% FEC correction implies a goodput of 17Mbps.

Fig. 3.9(a) shows a sample of the throughput and rate performance of the pseudo-multicast scheme. The throughput achieved is close to 9Mbps. We observe that pseudo-multicast frequently samples higher rates (up to 54Mbps) leading to packet losses. The

Table 3.3: Average throughput (Mbps) of pseudo-multicast, *MuDRA*, and SRA schemes with and without background traffic.

	No Background traffic	Background traffic
Fixed rate = 36Mbps	20.42	13.38
Pseudo-Multicast	9.13	5.36
MuDRA	18.75	11.67
SRA	19.30	4.55

average throughput for different schemes over 3 experiments of 300s each (conducted on different days) with 162 nodes is shown in Table 3.3. *MuDRA* achieves 2x throughput than pseudo-multicast scheme. The fixed rate scheme yields approximately 10% higher throughput than *MuDRA*. SRA has similar throughput as *MuDRA*.

Fig. 3.9(b) shows the distribution of average PDR of 162 nodes for the same 3 experiments. In the pseudo-multicast scheme, more than 95% of nodes obtain a PDR close to 100% (we did not consider any retransmissions to nodes listening in promiscuous mode). *MuDRA* meets the QoS requirements of 95% nodes with at least 85% PDR. On the other hand, in SRA and the fixed rate schemes 45% and 70% of the nodes have PDR less than 85%, respectively.

In pseudo-multicast, more reliable transmissions take place at the cost of reduced throughput, since the AP communicates with the node with the poorest channel quality in unicast. The significant difference in QoS performance of the fixed rate and SRA schemes is because the target rate can change due to interference etc. In such a situation, *MuDRA* can achieve the new target rate while the fixed rate and SRA schemes lead to significant losses (we observed that exceeding the target rate even 10% of time may cause up to 20% losses and less than 5% throughput gain).

Changing number of FB nodes: We varied the number of FB nodes (K) between $1-100$ for *MuDRA*. Fig. 3.9(c) shows the throughput as K changes. For $K = 1$, *MuDRA* tunes to the node with the worst channel quality, and consequently, the throughput is very low. On the other hand, increasing K from 30 to 90 adds similar amount of FB overhead as

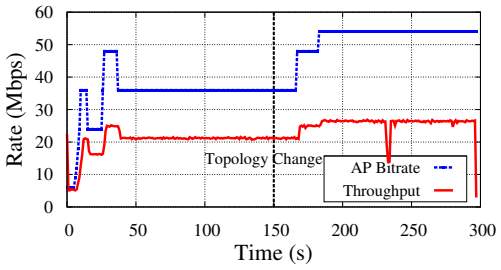


Figure 3.10: Emulating topology change by turning off FB nodes after 150s results in changing optimal rate for *MuDRA*.

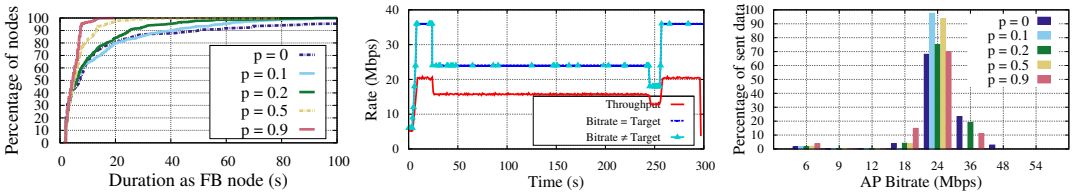
decreasing the report interval from 500ms to 200ms in Section 3.6. Thus, the throughput decreases for a large number of FB nodes. The throughput for K between $10-50$ does not vary significantly which is aligned with our discussion in Section 3.5 that *MuDRA* needs only $K > A_{max} + \epsilon$ for small ϵ to evaluate the target rate conditions.

Impact of topology changes: To demonstrate that changes in the network may lead *MuDRA* to converge to a different rate, we devised a strategy to emulate network topology changes on the grid. During an experiment, a number of FB nodes are turned off at a given time. Since FB nodes have the lowest PDRs, it may lead to changes in the target rate as a large number of nodes with low PDR disappear from the network. Fig. 3.10 shows the scenario when 30 FB nodes are turned off after 150s during the experiment. The rate converges quickly and without oscillations to a new target rate of 54Mbps.

3.7.2 Impact of High Node Churn

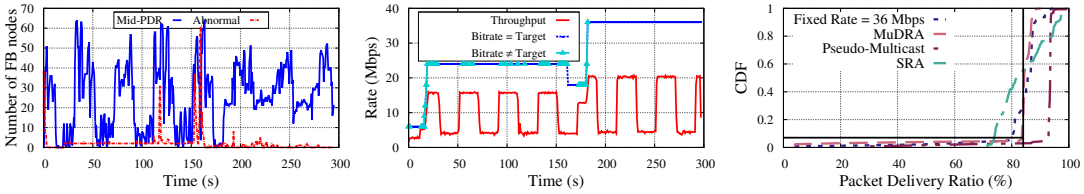
We evaluate the performance of *MuDRA* when emulating severe network changing conditions. In the experiments, each node leaves or joins the network with probability p every 6s. Thus, $p = 0.1$ implies that a node changes its state with probability of approximately 50% at least once in a minute. Initially, 50% of the nodes are randomly selected to be in the network.

We conducted 3 experiments consisting of 155 nodes (initially, 77 nodes are in on state). Fig. 3.11(a) shows the impact of p on the distribution of time duration that the nodes



(a) (b) (c)

Figure 3.11: Performance of *MuDRA* with high node churn: (a) Distribution of time durations for which a node is a FB node for different values of probability p of node switching its state on/off every 6s, (b) Multicast rate and throughput measured at the AP with $p = 0.2$, (c) Percentage of data sent at various rates for different values of p.



(a) (b) (c)

Figure 3.12: Performance of *MuDRA* with 155 nodes where an interfering AP transmits on/off traffic: (a) Mid-PDR and abnormal FB nodes, (b) Multicast rate and throughput, (c) CDF for PDR distribution with interference for fixed rate, *MuDRA*, pseudo-multicast, SRA.

remain as FB nodes. Higher values of p imply higher churn and lead to shorter periods for which nodes serve as FB nodes. The average number of changes in FB nodes per second is 2, 5, and 10 for p equal to 0, 0.2, and 0.9, respectively. Even with these changes, the average control overhead is very low (35Kbps) and is not affected by the degree of churn. Fig. 3.11(b) shows one instance of the RA process with $p = 0.2$. We see that *MuDRA* can adapt to the changing target rate at times 10, 30, and 255s. Fig. 3.11(c) shows the percentage of data sent at different rates for several values of p averaged over 3 different experiment runs. *MuDRA* achieves a similar rate distribution for all values of p. *Our experiments show that MuDRA can achieve the target rate, maintain stability, and adds low overhead, even under severe network changing conditions.*

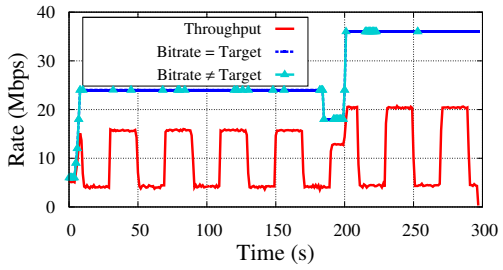


Figure 3.13: Multicast throughput with node 1-8 transmitting interfering on/off packet stream with node churn.

3.7.3 Impact of External Interference

We envision that *MuDRA* will be deployed in environments where the wireless infrastructure is centrally controlled. However, in-channel interference can arise from mobile nodes and other wireless transmissions. In addition to the uncontrolled interference spikes on ORBIT, we evaluate the impact of interference from a nearby node which transmits at the same channel as the multicast AP. We consider a scenario with two nodes near the center of the grid that exchange unicast traffic at a fixed rate of 6Mbps in a periodic on/off pattern with on and off periods 20s each. The transmission power of the interfering nodes is 0dBm which is equal to the transmission power of the multicast AP. This helps us evaluate the performance in the worst case scenario of continuous interference and study the dynamics of changing interference.

Fig. 3.12(a) shows the mid-PDR and abnormal nodes and Fig. 3.12(b) shows the rate and throughput for one experiment with 155 nodes. The number of mid-PDR nodes increases during the interference periods, due to losses from collisions. *MuDRA* converges to the target rate of 24Mbps. *Notice during interference periods, MuDRA satisfied the target-condition and that using the stability preserving method, MuDRA manages to preserve a stable rate.* The average throughput of different schemes with on/off background traffic for 3 experiments of 300s each is in Table 3.3. Pseudo-multicast achieves half while SRA has a third of the throughput of *MuDRA*. The fixed rate scheme achieves similar throughput as *MuDRA*.

The PDR distribution of nodes is in Fig. 3.12(c). *MuDRA satisfies QoS requirements while maintaining high throughput.* Pseudo-multicast scheme has 90% nodes with PDR more than 90% since it makes backoff decisions from unicast ACKs. SRA yields 55% nodes with PDR less than 85% as it transmits at low rates. The fixed rate scheme yields 30% nodes with PDR less than 85%. The fixed rate scheme performs better than SRA since it maintains a higher rate. We also investigate the combined impact of *both interference and node churn*, where every 6s, the probability of a node switching on/off is $p = 0.2$. Fig. 3.13 shows the rate and throughput for this case. Similar to results in Section 3.7.2, *the performance of the system is not affected by node churn.*

3.7.4 Video multicast

We demonstrate the feasibility of using *MuDRA* for streaming video. The video is segmented with segment durations equal to the period of rate changes (1s) and each segment is encoded at several rates in H.264 format. For each time period, the key (I) frames are transmitted reliably at the lowest rate 6Mbps (note that transmitting the key frames can be achieved with 100% reliability even at 12Mbps on the testbed). The non-key (B and P) frames are transmitted at the rate set by *MuDRA*.

Let the multicast rate for current time period be R,the expected data throughput at this rate be $\hat{D}_R$, and the estimated throughput at the minimum rate be $\hat{D}_{min}$. Let f_k be the fraction of key frame data and f_{nk} be the fraction of non-key frame data. The video server has to determine the video rate V_R at each time t. Let the fraction of transmission time for key frames $T_k = \frac{V_R \cdot f_k}{\hat{D}_{min}}$ and fraction of transmission time for non-key frames $T_{nk} = \frac{V_R \cdot f_{nk}}{\hat{D}_R}$. We know that

$$t_k + t_{nk} = 1$$

The video rate can be calculated by solving linear equations $V_R = \frac{\hat{D}_{min} \cdot \hat{D}_R}{\hat{D}_{min} \cdot f_{nk} + \hat{D}_R \cdot f_k}$. In environments where estimates of throughput are inaccurate due to interference, techniques such as in [202] can be utilized.

Experimental Results: We use raw videos from an online dataset [20] and encode the

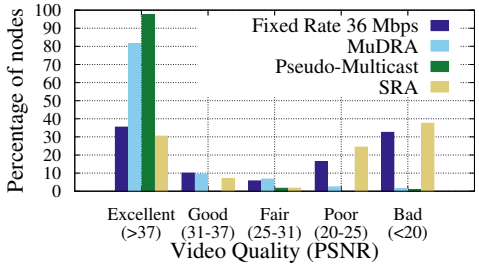


Figure 3.14: Distribution of video quality and PSNR (in brackets) measured at 160 nodes for different multicast schemes.

videos with H.264 standard. In our data sets, f_k is $15-20\%$. For *MuDRA* with throughput 19Mbps and FEC correction of 15%, we can support a video rate of $13-15$ Mbps, which is sufficient for 3 or 4 HD streams (each 4Mbps) on mobile devices. For each node, we generated the video streams offline by mapping the video frames to the detailed packet traces collected on ORBIT from an RA experiment. *For a fair comparison, the I frames were transmitted at* $6Mbps$ *for all schemes even though MuDRA can dynamically adjust the transmission rate to be much higher even for reliable transmissions.* In our experiments, we only considered a single video stream of rate V_R. We measured the PSNR of the video at each node and classified the PSNR in 5 categories based on visual perception[5].

Fig. 3.14 shows the video quality and PSNR ranges at the nodes for 3 experiments each of 300s and with $150-160$ nodes. With *MuDRA*, more than 90% of the nodes achieve excellent or good quality, 5% achieve fair quality, and less than 5% get poor or bad quality. While the pseudo-multicast scheme results in almost all nodes obtaining excellent quality, the video throughput for this scheme is significantly lower (8Mbps). SRA and the fixed rate schemes have more than 50% nodes with poor or bad video quality. The higher thorughput from *MuDRA* can allow streaming of several concurrent video streams or streams encoded at higher rates while ensuring QoS requirements.

[5]PSNR quantifies the distortion of the received as compared to the original transmitted video.

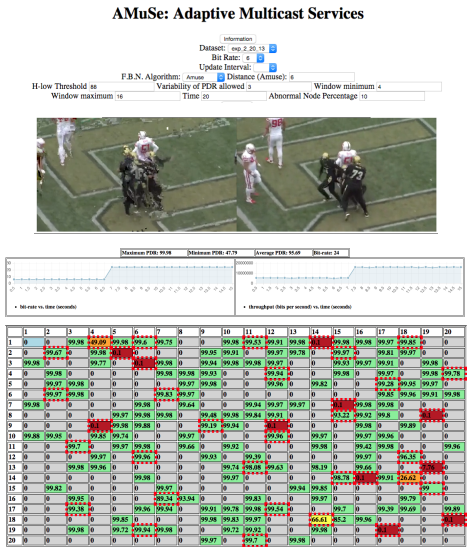


Figure 3.15: A screenshot of the web-based application for evaluating performance of *AMuSe* . The control panel for selecting the feedback and *MuDRA* algorithm parameters is on the top. The video at two selected nodes is shown below. In this example we show one node with poor quality and one with good quality video. The multicast throughput and other metrics are in the graphs. The performance of the client nodes is shown on the grid where numbers in each box indicate the PDR and the color of the box indicates the range of PDR. The nodes highlighted with a red border are FB nodes and nodes in grey are non-functional due to hardware issues.

3.8 Demonstration Application

To visually evaluate the performance of *AMuSe* and video delivery over *AMuSe* , we developed an interactive web-based application that illustrates the performance of the overall *AMuSe* system based on experimental traces collected on the ORBIT testbed. We collected the traces over several days in different experimental settings with 150-200 nodes. Each experimental trace consisted of channel measurements at each node using several metrics such Link Quality, Packet Delivery Ratio (PDR) etc.

The application allows considering different scenarios such as different channel conditions, interfering transmissions etc. For each scenario, the application shows the dynamic conditions over a period of time on the testbed from the appropriate experimental traces. The application can be used to compare the performance of several multicast schemes such

as pseudo-multicast, unicast transmissions in different scenarios that have been measured on the testbed (e.g. interference, other WiFi flows, etc.) as well as syntactic scenarios based on manipulating the measured data. We note that the application is flexible and can be used for testing even more scenarios and algorithms in the future.

The application has three main components: (i) the back-end where the experimental data is stored and managed, (ii) the front-end which provides the user interface, and (iii) a video tool for generating video streams. Both the front-end and the back-end are light weight applications. The front-end is web-based and can operate on any standard browser while the back-end requires installation of easily available open source libraries. For any experimental condition, the video tool generates the video stream received by a selected node. It maps video payload to UDP packets and discarding lost packet, according to the node's traces.

The front end is built using Angular [1] which is a JavaScript framework for rendering dynamic features on web applications. The FB node selection and *MuDRA* algorithms are built in the Django framework. The back-end utilizes a Postgres [18] database and interfaces with Django [6]. The algorithmic parameters can be tuned at any given time on the front-end. The front-end periodically relays the parameters to Django. Django utilizes the user input and system state information derived from the back-end to run the required FB and rate adaptation algorithms. The system state is then relayed to Angular, which renders the information on user's screen. Finally, the experiment can be paused at any time to allow the video tool to generate videos at the nodes for that period of time. The video tool uses ffmpeg to render and generate the videos and an nginx server [12] to transmit to the front-end.

The back-end utilizes a Postgres [18] database and interfaces with Django. The database is populated using the data derived from the experimental traces. The database consists of parameters several experimental parameters at each node at different times for each experimental scenario. This allows us to characterize the performance of the testbed with evolving channel conditions. The statistics about performance at each node are derived from the detailed packet traces. The feedback algorithms are built in the Django framework. The

application is very flexible and allows other feedback algorithms to be incorporated as well. The users can change the feedback algorithms and tune the algorithm specific parameters at any given time on the front-end.

The front end is built using Angular [1] which is a JavaScript framework designed for rendering dynamic features on web applications. The front-end periodically relays the user defined parameters to Django which runs the corresponding FB algorithm and responds with information (including the state of the nodes and the system) to Angular. Angular in turn renders the information on user's screen. The period of rendering at the front-end as well as calculation of system performance parameters can be changed by the user. Typically, we use a period of 500ms.

Fig. 3.15 shows a screenshot of the application. The application allows selecting different experiment settings such as AP bit rate, feedback algorithm, number of feedback nodes on the web interface. This information is used along with data collected from the experiments to show how the performance at all the nodes on the grid. The feedback nodes are highlighted with a red border. The rate adaptation and multicast throughput measured at the AP appears below. The information on the front-end is updated periodically.

Chapter 4

OPTIMIZING VIDEO QoE FOR MULTICAST STREAMING

4.1 Introduction

Mobile video is expected to contribute 78% of all the mobile traffic by 2021 [29]. Applications such as NFL Red Zone, which support live video streaming in crowded venues, are gaining attention but their uptake has been slow due to performance issues. Research has shown that videos which play at lower bitrates or freeze frequently lead to high abandoment which results in revenue loss for video providers and suboptimal use of network resources [85]. Thus, ensuring high *Quality of Experience (QoE)* is essential for video streaming.

Existing *unicast* video streaming techniques rely on *Adaptive Bitrate (ABR)* schemes to adjust the video playback for a diverse set of user devices and network conditions. Each video is partitioned into segments, where each segment includes a few seconds long playback. Each segment is then encoded in a number of different encoding rates (henceforth referred to as the video rates). When the user plays a video, the video player can download each segment at a video rate that is appropriate for its connection, thereby switching rates in response to changes in the available bandwidth.

WiFi multicast could potentially support live (or almost live) streaming in crowded

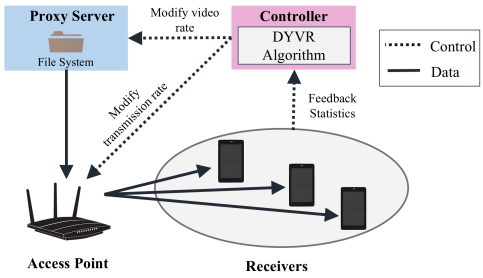


Figure 4.1: The multicast video streaming system where the *DYVR* Algorithm controls both the video rate and the multicast transmission rate.

venues more efficiently than unicast, since it leverages the shared nature of the wireless medium.[1] However, directly applying techniques similar to unicast ABR streaming for WiFi multicast is challenging due to the following reasons:

(i) Diverse channel conditions: Due to the diverse channel states at different receivers, selecting an appropriate video rate is non-trivial.

(ii) Difficulty in bandwidth estimation: A key assumption in several ABR streaming algorithms such as [28,30,113] is that future bandwidth estimates are available. Bandwidth estimation is challenging even in unicast [234] and is more challenging in multicast.

(iii) Lack of feedback: WiFi multicast does not have a reliable feedback mechanism. While the 802.11aa standard [179] attempts to resolve this, it is still not widely adopted and obtaining per-packet feedback (as available in unicast) will remain infeasible in the foreseeable future. Hence, only periodic, low granularity application layer feedback can be practically obtained.

(iv) Lack of transmission rate adaptation mechanism: While considerable research effort has been dedicated to multicast rate adaptation (see Section 7.2), the practicality of rate adaptation techniques has been limited due to performance issues.

To overcome these challenges, *we focus on leveraging WiFi multicast for video stream-*

[1]To enable multicast services, some resources can be provisioned either in the time or frequency domains.

Table 4.1: Effects of transmission rate (r) and video rate (v) on video QoE.

	Low v	Optimal v	High v
Low r	Under utilization + Poor video rate	Underflows	Underflows
Optimal r	Poor video rate	High quality	Underflows
High r	Packet losses + Poor video rate	Packet losses	Packet losses+ Underflows

ing.[2] In particular, we design and evaluate a system and algorithms for joint adaptation of the multicast transmission rate and video rate with the objective of optimizing the video QoE (see Fig. 4.1).

Video QoE is complex and depends on several parameters [70]. Network operators, video providers, or receivers may be interested in different QoE metrics such as the video rate, rebuffering events, number of video frames lost or corrupted, and video rate switches. Ensuring good performance across all QoE metrics is challenging. As an example for the need for joint adaptation to provide QoE, assume that the receivers start with a small amount of buffered video. If the transmission and video rate controller detects that the receivers' channel conditions are poor, the controller can reduce the transmission rate, video rate, or both. While reducing the transmission rate will lead to reliable transmissions, it may also cause buffer underflows. Reducing the video rate may minimize the buffer underflows but it is undesirable. These tradeoffs are summarized in Table 4.1.

Hence, our objective is to design a system in which the underlying streaming algorithm adjusts the transmission and video rates based on the desired QoE metrics (segments loss, buffer underflows, and video rate switches) specified by network operators or receivers. First, we formulate the QoE optimiztion probelm for wireless video multicast as a utility maximization problem. We present two variants of the *DYVR* (DYnamic Video and transmission Rate adaptation) online algorithm which dynamically tune the transmission and video rates to achieve different performance guarantees for video QoE constraints. We derive performance guarantees for the algorithms using the Lyapunov optimization frame-

[2]The concepts and techniques are more generally applicable to other wireless technologies such as LTE.

work [158]. We show that they can achieve time average utility which is within an additive term $O(\frac{1}{W})$ of the optimal value and satisfy the QoE constraints with an $O(W)$ factor, for a control parameter W. Our simulations in realistic settings derived from experimental traces show that the *DYVR* algorithms provide close to optimal performance while satisfying the QoE constraints.

Then, we present the system architecture for video streaming over WiFi multicast (shown in Fig. 4.1) that can be integrated with existing ABR services. The architecture is comprised of 4 main components: a proxy server, WiFi Access Point (AP), an AP controller, and receiver-side software. The proxy server interfaces with existing ABR services and locally caches the video content that is later tranmitted via mutlicast from a commercial off-the-shelf WiFi AP. The video rate at the proxy server and the multicast transmission rate at the AP are controlled by the *DYVR* Algorithm. The receiver-side software is a light-weight application that does not require any modifications to the hardware or operating system.

We implemented the architecture in a testbed composed of Android tablets and a WiFi AP. We evaluated the performance of the *DYVR* Algorithm through experiments and compared it to other schemes. *Our experiments in different channel conditions and receiver mobility settings show that DYVR can stream high definition video and satisfy the desired QoE constraints on segment loss, buffer underflows, and video rate switches. DYVR* can provide almost 2x higher video rate while ensuring 4x fewer segments losses and video rate switches in poor channel conditions when compared to state of the art unicast video streaming algorithms which have been tuned to optimal settings for a multicast environment. Even in challenging cases of receiver mobility, when compared to unicast streaming, *DYVR* can provide higher video rate while ensuring 2x fewer segment losses.

To summarize, the main contributions of this chapter are: (i) to the best of our knowledge, it presents the first online algorithm (*DYVR*) for QoE optimization of ABR video streaming over wireless multicast, and (ii) it presents an architecture and testbed implementation that allow evaluating the algorithm in realistic environments. The system can be integrated with existing ABR schemes.

The research presented in this chapter involved significant contributions from students in the Wimnet lab. Hannaneh Pasandi, Andy Xu, Bohan Wu, and Rodda John helped in implementing the system and algorithms presented in this chapter. A preliminary description of the system design and experimental results was presented in a demonstration in IEEE INFOCOM'17 [106]. The rest of the chapter is organized as follows. We review the related work in Section 4.2 and present the model in Section 4.3. We present the *DYVR* algorithm and its analysis in Section 4.4. The simulation and experimental results are presented in Sections 4.5 and 4.6, respectively.

4.2 Related Work

ABR-based video streaming, video streaming over wireless (for both unicast and multicast), and Lyapunov optimization for wireless networks received considerable attention. Below we review the relevant literature in these areas.

Video Adaptation: There is extensive literature on video rate adaptation techniques for ABR-based streaming. Most current commercial streaming applications rely on heursitics [28,30] that may work well in practice but do not provide any performance guarantees. The video rate adaptation algorithms can be classified into rate-based [143, 157, 203] and buffer-based [113, 195]. The rate-based algorithms usually rely on bandwidth prediction derived from historical performance or probing. Bandwidth predictions are hard and especially challenging for wireless multicast enviornments. Buffer-based algorithms instead use the amount of video in the player buffer to make rate decisions. Recent hybrid approaches [117, 225] use both the buffer and bandwidth prediction to optimize video QoE. All these techniques are not directly applicable to multicast, since they do not consider multiple clients and address the adaptation of both transmission and video rates.

Wireless Video: Addressing the challenges associated with the wireless and mobile video streaming has been gaining increasing attention. While bandwidth unpredictability over internet is widely known, this problem is even more severe in wireless networks. It was shown in [234] that reliable bandwidth predictions of even a few seconds could yield a significant

improvement in video performance. Related approaches rely on predicting underlying physical layer resource allocation for cellular networks [223, 227]. However, such methods are designed for a single receiver and may not be applicable to other physical-layer techniques. Another class of algorithms assumes that the system state, e.g., bandwidth, evolves as a Markov process. They leverage Markov Decision Process (MDP) [69, 222] to probabilistically derive the future bandwidth predictions and compute the expected utilities for all possible adaptation choices at every segment offline. Obtaining a general Markovian model for wireless networks is difficult. The Markovian approaches can also lead to explosion in state space due to a large number of receivers and transition probabilities.

Multicast Streaming: MuVi [226] and Medusa [187] make per-packet transmission rate decisions for maximizing the overall utility derived from the delivery of each packet. However, the aforementioned schemes do not consider ABR video. DirCast [67] and [45] focus on adjusting FEC to ensure reliable multicast. While multicast rate adaptation for maximizing overall throughput has attracted considerable attention, e.g. [99, 101], these techniques do not address the problem of improving video QoE. There has been considerable effort dedicated to Scalable Video Coding (SVC) techniques for multicast [82, 112, 126, 139]. Although we do not focus on SVC, we believe that our approach can be extended to QoE optimization for SVC-based streaming.

Lyapunov Techniques: Utilizing Lyapunov optimization techniques within the context of wireless networks for rate, congestion, and power control has been extensively studied (see [91, 93] and references therein). For example, EZ-Flow [44] is a buffer based flow control mechanism for 802.11 wireless mesh networks and BOLA [195] is an online algorithm for unicast ABR video streaming.

4.3 Model and Problem Formulation

We consider a single-hop wireless multicast network with N multicast receivers. Time is divided into slots of fixed length with a total of T time slots. For simplicity of presentation, we assume that a slot length is 1s. The AP can only set the multicast transmission and the

Table 4.2: Nomenclature

N	Number of receivers
T	Number of time slots
v_t	Video rate at time t
$\mathcal{V}$	Set of possible video rates
r_t	Multicast rate at time t
$\mathcal{R}$	Set of possible multicast rates
$q(v_t)$	User perceived quality of v_t
$\chi_t^i(r)$	Indicator of success at r at slot t
$p_t^i(r)$	Probability of success at r at slot t
r_t^i	Maximum r for which $\chi_t^i(r) = 1$
B_t^i	Buffer at receiver i at slot t
α	Desired fraction of lost segments
β	Desired fraction of buffer underflows
γ	Desired fraction of quality changes

video rates at the beginning of each time slot. At slot t, the video rate and transmission rate, denoted by v_t and r_t, can be chosen from discrete sets $\mathcal{V}$ and $\mathcal{R}$, respectively. Since the typical video segments are a few megabits in size and the multicast transmission rates can be on the order of tens of Mbps, multiple video segments can be transmitted in one slot. The nomenclature used throughout this section is summarized in table 4.2.

We do not consider the case where each packet can be multicast over several transmission rates. This is because unlike unicast rate, multicast rate can only be changed slowly with feedback collected over the time scale of several hundred milliseconds to avoid high feedback overhead [101].

The number of segments that can be transmitted in a slot t is $\frac{r_t}{v_t}$. The amount of video in the buffer at receiver i at time t is denoted by B_t^i. The units of the buffer are in seconds.

Channel State: Previous work [109] as well as our experimental observations have shown that the SNR value at a receiver must be greater than a threshold to decode packets at a particular rate. As long as the SNR is greater than the threshold, the receiver can decode almost 100% of the packets. Further, the channel correlation time is generally of the order of a few seconds. Since packet level retransmissions are infeasible for multicast[3], a small

[3]Per packet loss information could lead to high feedback overhead and may require changes to hardware

amount of losses $5-10\%$ can be recovered by application layer Forward Error Correction (FEC) over the duration of a slot.

Based on the previous observations, we assume a binary channel model for the duration of a slot. Namely, at any given rate and slot, a receiver receives either successuly receives the video segments or not. Further, with some abuse of notation we define r_t^i as the maximum rate for receiver i above which no video segments are received. At transmission rates below r_t^i, a receiver will successfully receive all video segments. Thus, r_t^i fully describes the state of a receiver at any time. We assume r_t^i is a stationary random variable.

In case that a segment is lost at a receiver due to transmission rate being higher than r_t^i, it can be recovered by a segment level retransmission mechanism. Further, for live video, few segments lost over a long period of time maybe tolerable. For simplification, we assume that a small number of segments lost are tolerable.

Channel Information: The channel state for the duration of a slot is denoted by an indicator variable $\chi_t^i(r)$ which signifies if the transmission at rate r is successful or not. The probability of a successful transmission at rate r is given by $p_t^i(r)$. We assume that the AP does not have accurate information about the channel state but can infer $p_t^i(r)$. $p_t^i(r)$ can be estimated from historical channel performance or other channel metrics such as RSSI and CSI [109]. In this chapter, we do not focus on the methods to derive $p_t^i(r)$ itself. Instead, we discuss how even coarse estimates of channel state can improve video QoE.

Buffer Dynamics: The video buffer at each receiver will drain at a constant rate. Thus, during a single slot, the buffer can drain by at most 1s. The number of segments are added to the buffer in the slot is $\chi_t(r_t)\frac{r_t}{v_t}$. Setting a value of r_t higher than r_t^i or video rate $v_t > r_t$ may lead to buffer underflows at receiver i.

Objective: Our objective is to maximize the overall QoE at the receivers. The QoE of the video depends on a large number of factors with the most important one being the video rate. We assume that the utility of a video segment at a receiver is given by a concave function q of video rate v_t. Besides the video rate, we consider three other key factors that

or wireless standards.

affect the QoE:

(i) Lost Segments: As described before, a small amount of segments lost are tolerable and can be recovered by a segment level retransmission mechanism. However, a large number of lost segments degrades the video watching experiience of the end-user.

(ii) Buffer Underflows: [85] has shown that freezes caused by buffer underflows, i.e., when the amount of video in a buffer falls to 0, has a large negative impact on user-engagement.

(iii) Video Rate Switches: While not as disrputive as buffer underflows, frequent and abrupt video rate switches are undesriable [157].

Thus, for maximzing QoE, the average video rate should be maximized while meeting some constraints for the 3 QoE factors (number of lost segments, buffer underflows, and video rate switches). The optimization problem, when the channel indicator variables, $\chi_t^i(r)$, are known in advance for all i and t can be formulated as follows:

Problem 1: *QoE Optimization with Per-Receiver Constraints (QPRC)*

$$\max \quad \frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T} q(v_t)\chi_t^i(r_t)$$

$$\text{subject to} \quad B_t^i = \Big[B_{t-1}^i - 1\Big]_+ + \frac{r_t}{v_t}\chi_t^i(r_t) \tag{4.1a}$$

$$\frac{1}{T}\sum_{t=1}^{T}(1-\chi_t^i(r_t)) \leq \alpha \quad \forall i = \{1,...,N\} \tag{4.1b}$$

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{1}_{B_t^i \leq 0} \leq \beta \quad \forall i = \{1,...,N\} \tag{4.1c}$$

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{1}_{v_t \neq v_{t-1}} \leq \gamma \tag{4.1d}$$

$$v_t \in \mathcal{V} \quad \forall t = \{1,...,T\} \tag{4.1e}$$

$$r_t \in \mathcal{R} \quad \forall t = \{1,...,T\} \tag{4.1f}$$

Constraint (4.1a) indicates the time evolution of the buffer at receiver i. Each successful segment reception adds $\frac{r_t}{v_t}$ seconds of video to the buffer and the amount of video consumed in each slot is fixed and equal to the duration of the timeslot (1s as based on our assumption). Constraints (4.1b) and (4.1c) are QoE constraints for the number of segments received at

each receiver and number of rebuffering events, respectively. Constraint (4.1d) specifies a limit on the number of video rate switches.

The *QPRC* Problem considers a scenario with segments lost constraints on each receiver. However, in practice, satisfying strict constraints on each receiver individually may lead to poor performance. As an example, consider a case with a large number of receivers where one or two receivers have poor channel quality. In such a case, guaranteeing high quality while meeting the constraints might unfairly penalize the receivers with good channel quality.

Hence, we consider an alternative formulation as follows:

Problem 2: *QoE Optimization with Aggregate-Receiver Constraints (QARC)*

$$\begin{aligned}
\max \quad & \frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T} q(v_t)\chi_t^i(r_t) \\
\text{subject to} \quad & \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(1-\chi_t^i(r_t)) \leq \alpha && \text{(4.2a)} \\
& B_t^i = \left[B_{t-1}^i - 1\right]_+ + \frac{r_t}{v_t}\chi_t^i(r_t) && \text{(4.2b)} \\
& \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{1}_{B_t^i \leq 0} \leq \beta \quad \forall i = \{1,...,N\} && \text{(4.2c)} \\
& \frac{1}{T}\sum_{t=1}^{T}\mathbb{1}_{v_t \neq v_{t-1}} \leq \gamma && \text{(4.2d)} \\
& v_t \in \mathcal{V} \quad \forall t = \{1,...,T\} \\
& r_t \in \mathcal{R} \quad \forall t = \{1,...,T\}
\end{aligned}$$

The main difference between the *QPRC* and *QARC* problems is that the latter considers a constraint on the average number of segments lost over all receivers. This provides looser QoE constraints but does not unfairly penalize the performance due to the presence of a few receivers with poor channel quality.

Finally, we assume that there is a feasible solution for both the *QPRC* and *QARC* problems. This is a practical assumption since for most wireless technologies, setting $r_t = r_{min} = min(r \in \mathcal{R})$ implies $\chi_t^i(r_t) = 1$, and thus, a solution for both problems can be found by setting $r_t = r_{\min}$ for all slots. Since, $\chi_t^i(r)$ values are not known in advance, our goal is to develop optimal online policies for the *QPRC* and *QARC* problems.

4.4 Online Transmission and Video Rate Adaptation

We utilize the Lyapunov framework [158] to develop online algorithms for the *QPRC* and *QARC* problems. The drift-plus-penalty method is the key technique in Lyapunov optimization [158] which stabilizes a queueing network while also optimizing the time-average of an objective (e.g., utility derived from video segments). To use this framework, a solution to the *QPRC* Problem must address the following:

QoE Constraints: To handle the QoE constraints of the *QPRC* Problem, we represent the QoE constraints for each receiver as virtual queues. Constraint (4.1b) can be represented by a virtual queue as follows:

$$X^i_{t+1} = \left[X^i_t - \alpha + (1 - \chi^i_t(r_t))\right]_+ .$$

Similarly, constraints (4.1c) and (4.1d) can be represented by virtual queues Y^i_t and Z_t such that:

$$Y^i_{t+1} = \left[Y^i_t - \beta + \mathbb{1}_{B^i_t \leq 0}\right]_+$$

$$Z_{t+1} = \left[Z_t - \gamma + \mathbb{1}_{v_t \neq v_{t-1}}\right]_+$$

The constraints for the *QARC* Problem can be transformed in a similar way. For details, see the technical report [105].

For the virtual queues to be stable, we have:

$$\lim_{t\to\infty} \sum_{\tau=1}^{t} (1 - \chi^i_\tau(r_\tau)) \leq \alpha$$

$$\lim_{t\to\infty} \sum_{\tau=1}^{t} \mathbb{1}_{B^i_t \leq 0} \leq \beta$$

$$\lim_{t\to\infty} \sum_{\tau=1}^{t} \mathbb{1}_{v_t \neq v_{t-1}} \leq \gamma$$

Thus, if the virtual queues are stable, the QoE constraints are also satisfied, since the input of the virtual queues is smaller than the output.

Next, we define conditions on the virtual queues Let Q_t be a vector process of queue lengths for a discrete stochastic queueing network with an arbitrary number of queues.

Let $L(Q)$ be a non-negative scalar function of the queue lengths, termed as the Lyapunov function. Define the Lyapunov drift as follows:

$$\Delta_t\left(Q_t\right) = \mathbb{E}[L(Q_{t+1}) - L(Q_t)]$$

We assume that a reward is accumulated every slot where a reward corresponds to the utility derived from an action at a slot. Let f_t denote the bounded function of the reward at slot t and f^* be the target reward. The following lemma specifies conditions for which the time average of the reward process is close to the target reward.

Lemma 1. *Suppose there exist finite constants $W > 0$, $C > 0$, and a non-negative function $L\left(Q_t\right)$ such that $\mathbb{E}[L\left(Q_t\right)] < \infty$ for every t. If the Lyapunov drift satisfies:*

$$\Delta\left(Q_t\right) - W\,\mathbb{E}[f_t] \leq C - W f^*,$$

for every t, then we have:

$$\liminf_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}[f(\tau)] \geq f* - \frac{C}{W}.$$

Proof. The proof can be obtained by a telescoping sums argument. □

The following lemma shows the existence of a randomized stationary policy that meets the target reward.

Lemma 2. *For large T, there exists a stationary policy, referred to as STAT, that is independent of the states of the virtual queues which makes i.i.d. control decisions in every slot and satisfies the virtual queues stability constraints while achieving time-average utility no smaller than f^*.*

Proof. The proof follows from Theorem 4.5 in [158] and is omitted for brevity. □

Note that calculating a policy described in Lemma 2 explicitly would require the knowledge of channel performance for each receiver or the $\chi_t^i(r_t)$ indicator variables. However, instead of calculating this policy explicitly, we will use its existence and characterization per Lemma 2 to design an online control algorithm using Lyapunov optimization.

4.4.1 The DYVR Algorithms

We now describe two variants of the *DYVR* (DYnamic Video and transmission Rate adaptation) algorithm for the *QPRC* and *QARC* problems. *DYVR-M* (*DYVR*-Maximum) for the *QPRC* Problem sets the transmission and video rate based on the performance of the receiver with the largest value of virtual queues. On the other hand, *DYVR-A* (*DYVR*-Average) for the *QARC* Problem makes decisions based on the average value of virtual queues at the receivers.

DYVR-M Algorithm: Algorithm 3 shows the outline of the *DYVR-M* Algorithm. It maintains per-receiver virtual queues X_t^i and Y_t^i and a global virtual queue Z_t. For every slot, the x and y indexes are calculated in lines 2–3. The values of $v_t \in \mathcal{V}$ and $r_t \in \mathcal{R}$ are chosen such that they maximize the max-weight equation in line 4, where W is an algorithmic parameter. After the end of the slot, it collects the feedback $\chi_t^i(r_t)$ as shown in line 8. The feedback is used to refine the estimates of channel for the next slot in line 9. Finally, the virtual queues are updated as shown in lines 10–12.

DYVR-A Algorithm: Algorithm 4 shows the outline of the *DYVR-A* Algorithm. The operation of *DYVR-A* is similar to *DYVR-M* . It maintains global virtual queues X_t, Y_t, and Z_t. The values of $v_t \in \mathcal{V}$ and $r_t \in \mathcal{R}$ are chosen such that they maximize the max-weight equation in line 2. The feedback collection process in line 6 is same as *DYVR-A* and updating of virtual queues is done in lines 8–10.

4.4.2 Performance Analysis

Here we analyze the performance of the *DYVR-M* Algorithm. The analysis of *DYVR-A* is similar and can be found in the technical report [105]. The following lemma shows that the utility achieved by *DYVR-M* is within an additive factor $O(\frac{1}{W}$ within the optimal, for a parameter W.

Theorem 1. *The overall utility achieved by the DYVR-M Algorithm satisfies:*

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=1}^{t} q(v_t)\chi_t^i(r_t) \geq f^* - \frac{C}{W},$$

Algorithm 3 *DYVR-M* Algorithm

for $t \leftarrow 1$ **to** T **do**
  $x \leftarrow \arg\max_i X_t^i$
  $y \leftarrow \arg\max_i Y_t^i$
  $(v_t, r_t) \leftarrow \arg\max_{v,r} \Big[W \sum_{i=1}^{N} q(v_t) p_t^i(r_t) -$
    $X_t^x \left(1 - p_t^x(r_t) - \alpha\right) - Y_t^y \left(\mathbb{1}_{B_t^y \leq 0} - \beta\right) - Z_t \left(\mathbb{1}_{v_t \neq v_{t-1}} - \gamma\right) \Big]$
  s.t. $v \in \mathcal{V}, r \in \mathcal{R}$
  Set transmission rate r_t and video rate v_t
  for $i \leftarrow 1$ **to** N **do**
    Collect $\chi_t^i(r_t)$ feedback from receiver i
    Estimate p_{t+1}^i
    $X_{t+1}^i \leftarrow \left[X_t^i - \alpha + (1 - \chi_t^i(r_t))\right]_+$
    $Y_{t+1}^i \leftarrow \left[Y_t^i - \beta + \mathbb{1}_{B_t^i \leq 0}\right]_+$
  $Z_{t+1} \leftarrow \left[Z_t - \gamma + \mathbb{1}_{v_t \neq v_{t-1}}\right]_+$

where f^* is the optimal utility achieved by any policy that meets the QoE constraints, W is an algorithmic parameter, and C is a constant.

Proof. Let $Q(t) = (X^1(t), \cdots, X^N(t), Y^1(t), \cdots, Y^N(t), Z(t))$ be the collection of virtual queues at all receivers. We consider the following Lyapunov function:

$$L(Q_t) = \frac{1}{2}\left[\sum_{i=1}^{N} {X_t^i}^2 + \sum_{i=1}^{N} {Y_t^i}^2 + {Z_t}^2\right],$$

The Lyapunov drift plus penalty for *DYVR-M* is given as:

$$\Delta(Q_t) - W \sum_{i=1}^{N} \mathbb{E}[q(v_t)\chi_t^i(r_t) \mid Q(t)] \leq C - \sum_{i=1}^{N} \mathbb{E}\Big[W q(v_t)\chi_t^i(r_t) - X_t^i \left(1 - \chi_t^i(r_t) - \alpha\right) - Y_t^i \left(\mathbb{1}_{B_t^i \leq 0} - \beta\right) - Z_t \left(\mathbb{1}_{v_t \neq v_{t-1}} - \gamma\right) | Q(t) \Big], \quad (4.4)$$

where,

$$C = \frac{N(1-\alpha)^2 + N(1-\beta)^2 + (1-\gamma)^2}{2}.$$

Let and x be the index at which X_t is maximum, thus $x = \arg\max(X_t^i : i \in (1, \cdots, N))$. Similarly, define y. Then, (4.4) can be written as:

$$\Delta(Q_t) - W \sum_{i=1}^{N} \mathbb{E}[q(v_t)\chi_t^i(r_t) \mid Q(t)] \leq C - \mathbb{E}[\sum_{i=1}^{N} W q(v_t)\chi_t^i(r_t) - N X_t^x \left(1 - \chi_t^x(r_t) - \alpha\right) - N Y_t^y \left(\mathbb{1}_{B_t^y \leq 0} - \beta\right) - Z_t \left(\mathbb{1}_{v_t \neq v_{t-1}} - \gamma\right) | Q(t)]. \quad (4.5)$$

Algorithm 4 *DYVR-A* Algorithm

for $t \leftarrow 1$ **to** T **do**

$\{v_t, r_t\} \leftarrow \arg\max_{v,r} \Big[W \sum_{i=1}^{N} q(v) p_t^i(r) -$
$X_t \left(\frac{1}{N}\sum_{i=1}^{N}\left(1 - p_t^i(r)\right) - \alpha\right) - Y_t \left(\sum_{i=1}^{N} \mathbb{1}_{B_t^i \leq 0} - \beta\right) - Z_t \left(\mathbb{1}_{v \neq v_{t-1}} - \gamma\right)\Big]$

s.t. $v \in \mathcal{V}, r \in \mathcal{R}$

Set transmission rate r_t and video rate v_t

for $i \leftarrow 1$ **to** N **do**

Collect $\chi_t^i(r_t)$ feedback from receiver i

Estimate p_{t+1}^i

$X_{t+1} \leftarrow \left[X_t - \alpha + \frac{1}{N}\sum_{i=1}^{N}(1 - \chi_t^i(r_t))\right]_+$

$Y_{t+1} \leftarrow \left[Y_t - \beta + \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{B_t^i \leq 0}\right]_+$

$Z_{t+1} \leftarrow \left[Z_t - \gamma + \mathbb{1}_{v_t \neq v_{t-1}}\right]_+$

If the *DYVR-M* Algorithm is implemented at all slots upto t, the virtual queue backlogs $(X^i(t), Y^i(t), Z(t))$ are determined by the history before t only. Thus, given the current virtual queue backlogs, *DYVR-M* maximizes the expectation on the right hand side of (4.5) over all alternative policies, including *STAT*. Since *STAT* satisfies time average constraints, $\mathbb{E}[X_t^i\left(1 - \chi_t^x(r_t) - \alpha\right)|Q(t)] = 0$. Similarly for other terms corresponding to the constraints in (4.5). Thus we can rewrite (4.5) as:

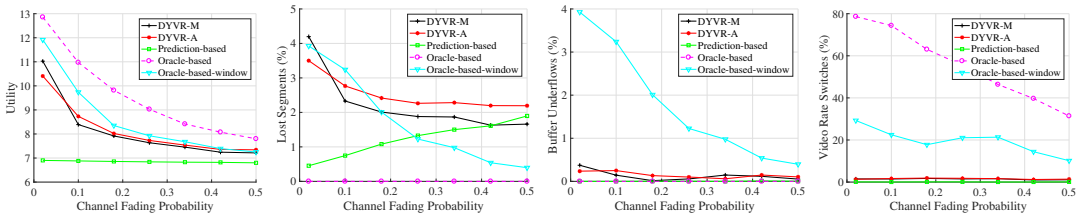
$$\Delta\left(Q_t\right) - W\sum_{i=1}^{N}\mathbb{E}[q(v_t)\chi_t^i(r_t) \mid Q(t)] \leq C - \mathbb{E}[\sum_{i=1}^{N} W q(v_t^{STAT})\chi_t^i(r_t^{STAT})|Q(t)].$$

The above equation is of the form required in Lemma 1. Then, from Lemma 2, we prove the desired result. □

We can further show that the maximum value virtual queues X_t^i, Y_t^i, Z_t grows with a multiplicative factor $O(W)$, thus yielding a tradeoff between achieving the optimal utility and satisfying QoE constraints (see [105] for more details).

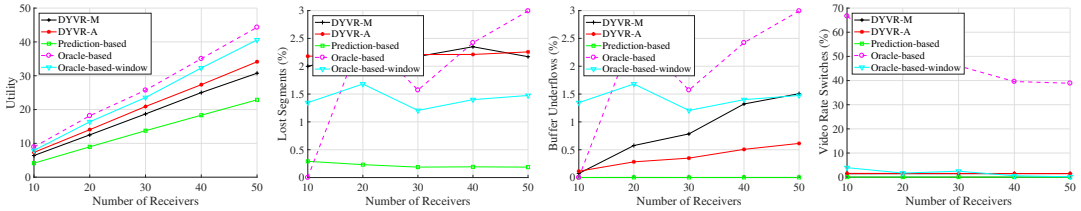
4.5 Numerical Evaluations

To evaluate the performance of both the *DYVR-M* and *DYVR-A* algorithms at a large scale which is not feasible experimentally, we performed extensive simulations. The simulation environment mimics the characteristics of real networks that we observed through



(a) (b) (c) (d)

Figure 4.2: Simulation results for uniform channel conditions with varying channel fading probabilities (a) average utility achieved, (b) percentage of lost segments, (c) percentage of buffer underflows, and (d) percentage of video rate switches.



(a) (b) (c) (d)

Figure 4.3: Simulation results for fast changing channel conditions with varying number of receivers: (a) average utility achieved, (b) percentage of lost segments, (c) percentage of buffer underflows, and (d) percentage of video rate switches.

experimental measurements and of those reported in [99, 109, 174].

We assume that *DYVR-M* and *DYVR-A* can estimate the channel conditions at the receivers at the beginning of each slot. More specifically, we assume that the probability distribution of the state of the system is known at the beginning of each slot. *DYVR-M* and *DYVR-A* do not have any knowledge about the long-term channel state.

We compare the performance of *DYVR-M* and *DYVR-A* to the following buffer/virtual queue independent algorithms:

(i) Oracle: The *Oracle* Algorithm has exact knowledge of the channel conditions for each time slot up to a maximum window of wnd slots. At the beginning of each slot, it sets the transmission rate r_t to be the maximum rate at which a fixed fraction f of the receivers can successfully receive segments. The value of f is a fixed parameter for a simulation and is

generally very high. For a fair comparison, we tune f such that the *Oracle* Algorithm leads to similar number of segments lost as *DYVR* . Further, *Oracle* sets the video rate as the average of transmission rates over wnd, $v_t = \frac{1}{wnd}\sum_{\tau=t}^{t+wnd} r_\tau$ to avoid frequent video rate switches.

We consider two variants of *Oracle* with $wnd = 1$ called *Oracle-Based* and a large wnd called *Oracle-Based-Window*. *Oracle-Based* provides a loose upper bound on the achievable utility with more video rate switches and underflows. *Oracle-Based-Window* provides a more realistic comparison to *DYVR* where the number of video rate switches and underflows are close to *DYVR*. We choose wnd values for *Oracle-Based-Window* such that the number of video rate switches and buffer underflows are close to that of *DYVR*.

(ii) Prediction-Based: The *Prediction-Based* Algorithm has the same knowledge of the channel conditions as the *DYVR* algorithms. More specifically, it knows $p_t^i(r)$, the probability of successful reception at each receiver i and at each rate r. At the beginning of each time slot, it selects r_t such that an expected fraction f of receivers will successfully receive the video segments. Similar to *Oracle*, we choose f such that the number of segments lost is close to *DYVR* and a window-based mechanism to set the video rate $v_t = \frac{1}{wnd}\sum_{\tau=t}^{t+wnd} r_\tau$.

We simulate a variety of environments with different channel state distributions, receiver mobility patterns, and varying number of receivers. We assume that the transmission and video rate values can be chosen from sets of 8 different values each. The channel state characteristics of the simulated environments mimic those of real networks obtained through experimental measurements and existing literature. For our measurements, we conducted experiments with Nexus 7 tablets and an ASUS WiFi AP in indoor settings over a 5GHz channel for 802.11a transmissions. We measured the probability of successful packet reception at the receiver at different locations for different values of transmission rates for 5 experimental runs of 500s each. We observed that packet losses are bursty, the amount of losses is stable for the duration of a few seconds for stationary receivers, and there are atypical events that can lead to high losses for short durations of time. These observations agree with measurements in existing literature [109, 174]. Accordingly, the simulation scenarios and the assumptions are as follows:

(i) Uniform: The maximum transmission rate for which a receiver i can successfully receive a segment, r_t^i, is chosen uniformly at random at beginning and remains same at all times. However, for certain randomly chosen slots, the maximum r_t^i drops to a lower value. This assumption models the channel fading effect we experimentally observed. We assume that the *DYVR* and *Prediction-Based* algorithms only know the probability of these fading events, while the *Oracle* Algorithm can predict a fading event in advance.

(ii) Mobility: The state of the system changes slowly over a period of time. We simulate a condition when the r_t^i value for each receiver change according to a discrete Markov Chain with 8 states corresponding to 8 channel states . Markov channel models have been extensively studied before [216, 231][4]. We consider a variety of transition probabilities to simulate the effect of higher mobility.

We developed a custom simulation tool based on the above observations. For various simulation scenarios, we ran 5 instances each 2,500 slots long.

Fig. 4.2 shows the performance of different algorithms in the Uniform scenario. Each simulation consisted of 10 receivers. We set the QoE parameters $\alpha = \beta = \gamma = 0.02$. Fig. 4.2(a) shows the average utility achieved for different algorithms. The utility achieved by *DYVR-A* is marginally better than by *DYVR-M* and the average utility reduces as the fading probability increases. The utility achieved by both *DYVR* algorithms is close to that of of the *Oracle-Based-Window* Algorithm but higher than the utility of the *Prediction-Based* Algorithm.

Figs. 4.2(b), 4.2(c), and 4.2(d) show the average percentage of segments lost, percentage of slots with buffer underflows, and percentage of slots with video rate switches, respectively. We observe that both *DYVR* algorithms achieve performance as dictated by QoE requirements. While *DYVR-A* achieved marginally higher utility than *DYVR-M*, it also leads to marginally higher number of segments lost due to looser constraints on the number of segments lost.

[4]While significant effort has been dedicated to modeling mobility (e.g., [33, 58] and subsequent literature consider Markovian mobility models), we use a simplistic mobility model since our focus is on the algorithmic performance evaluation rather than on mobility patterns.

Since the parameters of *Oracle-Based-Window* and *Prediction-Based* algorithms were tuned to yield performance close to *DYVR*, they satisfy the QoE requirements. It should be noted that the parameters of *Oracle-Based-Window* and *Prediction-Based* algorithms were obtained by rigorous trial and error. In practice, these parameters will change in different environments and it is infeasible to tune these parameters in realistic environments. The *Oracle-Based* Algorithm does not result in any segments lost and buffer underflows, due to knowledge of channel states. However, it results in large number of video rate switches.

Fig. 4.3 shows the performance of different algorithms in the Mobility scenario as a function of the number of receivers in the system. As expected, the average utility as shown in Fig. 4.3(a) grows for each algorithm with the number of receivers. Both *DYVR-A* and *DYVR-M* achieve higher utility than *Prediction-Based* but lower than the *Oracle-Based* and *Oracle-Based-Window* algorithms. Further, the gap between the performance of *Oracle*, *DYVR*, and *Prediction-Based* algorithms grows larger with increasing number of receivers.

Figs. 4.3(b), 4.3(c), and 4.3(d) show the average number of segments lost, number of buffer of underflows, and video rate switches, respectively. Even in the challenging mobility scenario, the *DYVR-M* and *DYVR-A* algorithms satisfy the required QoE constrains. Moreoever, the number of lost segments, buffer underflows, and video rate switches are less than 2%. The *Prediction-Based* and *Oracle-Based-Window* algorithms satisfy the QoE requirements by design but *Oracle-Based* results in high number of video rate switches.

In summary, the simulations demonstrate that both the *DYVR-A* and *DYVR-M* algorithms can provide close to optimal utility while satisfying the QoE requirements.

4.6 Implementation and Experimental Evaluation

In this section, we describe our wireless multicast video delivery system and the experimental testbed. We also present the results of experimental evaluations of *DYVR* algorithms.

4.6.1 System Architecture

The system consists of 4 main components as shown in Fig. 4.4:

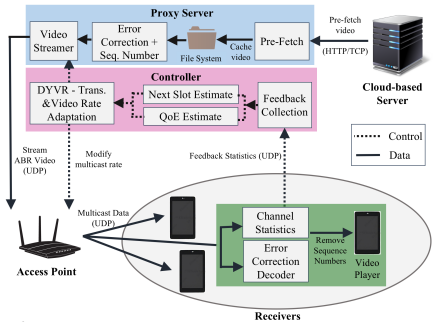


Figure 4.4: The wireless multicast video delivery system (with the *DYVR* algorithms at its core) consisting of: (i) Proxy Server (ii) Controller, (iii) WiFi Access Point, and (iv) Receivers.

(i) Proxy Server: The proxy server pre-fetches video segments at all available video rates from a cloud-based server for streaming to multicast receivers. The pre-fetch module performs the function of a standard DASH player. The pre-fetching can be based based on current or predicted demand for a particular video (e.g., for a key highlight in a stadium). We assume that the bandwidth between the Proxy and the cloud server is unlimited so the delay in pre-fetching operation is minimal. However, buffering each segment adds delay which is directly proportional to the segment length. We use a small segment length of 2s to avoid large delays.

The proxy server also adds application-layer error correction to the video segments and embeds sequence numbers in each packet. Finally, the video data is transmitted to the WiFi AP using UDP packets through a video streaming module which adjusts the video rate based on input from *DYVR* . We selected UDP due to its compatibility with the existing media player libraries.

(ii) Controller: The Controller maintains detailed statistics of QoE performance at each receiver. The Controller collects periodic feedback reports from receivers about the number of segment losses, video buffer levels, RSSI, Packet Delivery Ratio (PDR). It also estimates the channel state at the receivers in the next time slot. Both the QoE performance and the next slot estimate are derived from the feedback reports. The controller is responsible for adjusting two key parameters determined by the *DYVR* Algorithm:

- Video Rate: The controller configures the video rate for each segment at the Proxy

Table 4.3: Commercial OpenWRT or DD-Wrt compatible WiFi APs

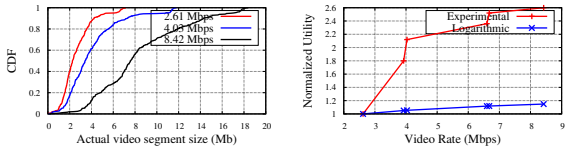
Manufacturer/Model	Linksys WRT1200AC	TP-Link Archer C7 v2	ASUS RT-AC56U
CPU Architecture	ARMv7	MIPS-32	ARMv7
WLAN Chipset	Marvell 88W8864	Atheros QCA9880	Broadcom BCM4708A0
Driver Support (Changing multicast rate)	No	Yes (non-official)	Yes (official)
Customize Firmware	OpenWrt	DDWrt	OpenWrt

Server.

- Transmission Rate: The controller communicates with the AP and sets the transmssion rate.

(iii) WiFi AP: The WiFi AP adapts the multicast transmission rate based on instructions from the controller. We experimented with several popular OpenWRT- and DD-WRT-compatible off-the-shelf APs to determine the feasibility of rate adaptations. Our observations are summarized in Table 4.3. We noticed that most popular drivers support multicast rate change from application layer calls. When the driver does not support the change, (e.g., Marvel), the driver can be modified to build a pipeline to support the rate changes. Since the wireless infrastructure is usually under the control of wireless operators, performing AP side changes is feasible option for a wide deployment of such a system architecture.

(iv) Receivers: The receivers in a multicast group listen to the UDP video stream packets, collect and calculate the performance statistics, play the video, and send the statistics back to the controller. The receiver first strips the sequence numbers from the packets at the application layer to compute packet delivery statistics. Each packet is passed to the error-correction decoder. Once the decoder determines that all the sequence numbers within an error correction block have been received, it decodes the block and forwards it to the media player at the receiver. The performance statistics are sent to the Controller periodically.



(a) (b)

Figure 4.5: (a) The distribution of video segment sizes at 3 different video rates, and (b) The concave utility function used in experiments and a standard logarithmic utility function shown for comparison.

4.6.2 Practical Issues

Feedback: *DYVR* needs estimates of channel state p_t^i values from all receivers. It also needs to keep track of the virtual queues X_t^i, Y_t^i, and Z_t^i. The feedback messages from each receiver are a few Kilobits in size. Since these values need to be estimated at the order of a video segment duration (seconds), the feedback overhead is not high.

Video Rate Variation: The *DYVR* algorithms described in Section 4.4 assume that the size of the segment is equal to the video rate. In practice, the video rate only specifies the average segment size over a period of time. Fig. 4.5(a) shows the distribution of segment sizes of a particular video file encoded in several video rates. It is clear that the actual segment size can be more than 2x higher than the video rate. Since the sizes of individual segments are known in advance at the Proxy Server, the actual video rates can be incorporated in *DYVR* at the beginning of each time slot.

Flow Rate: The multicast flow rate is based on several factors such as the available buffer at the client and the AP, the transmission rate, and time duration available for multicast in a slot. Typically, the downlink buffer available for multicast at APs is only a few hundred kilobytes. Thus, the flow rate should not significantly exceed the multicast throughput. In our implementation, the Controller sets the flow rate to the transmission rate determined by the *DYVR* algorithm. Further, while we assume that receivers have a large amount of buffer, the Controller pauses the video tranmission if the buffer levels grow beyond 40 segments.

Utility Function: The increase in the video utility with video rates diminishes as the

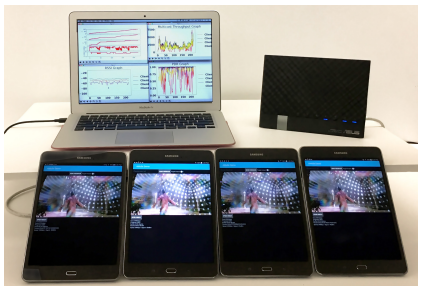


Figure 4.6: The implementation of the architecture shown in Fig. 4.4 which was used for experimental performance evaluation: (i) a laptop acting as the Proxy Server and Controller, (ii) the WiFi Access Point, and (iii) receivers.

video rates increase. Thus, it is natural to consider a concave utility function for *DYVR*. For the limited set of video rates available, we computed a utility function that works well in practice. Fig. 4.5(b) shows the utility function for different video rates and the logarithmic utility function for comparison.

Channel State Estimation: The *DYVR* algorithms rely on evaluating $p_t(r)$ values at the beginning of each time slot. The next slot estimate module in the system calculates these values using a combination of current RSSI and past history of PDR values. We collected offline measurement data with the AP transmitting multicast data at a particular transmission rate for 200s. We used 10 sets of experiments for each transmssion rate to estimate mappings between RSSI values to PDR values at different transmission rates.

With these mappings, we achieve 90% accuracy in predicting PDR within 5% range for another set of experimental data. To account for prediction errors due to environmental changes and noise, we consider an additional online technique that refines PDR estimate at the current RSSI value. If the PDR values are lower than 80% for 3 subsequent slots, we lower the estimated PDR for the current RSSI values.

4.6.3 Implementation

We implemented the architecture described in the previous section on a testbed as shown in Fig. 6.4(b). A preliminary version of this testbed was recently demonstrated in [106].

(i) Proxy-Server and Controller: The Proxy Server and Controller were implemented

on a standard laptop. For the experimental evaluation described in the following section, we ignored the live pre-fetching and video encoding mechanisms to avoid complexity. Instead, the laptop hosted video segments at pre-encoded at different video rates. The laptop was located in close proximity to the AP to minimize delay in control functions.

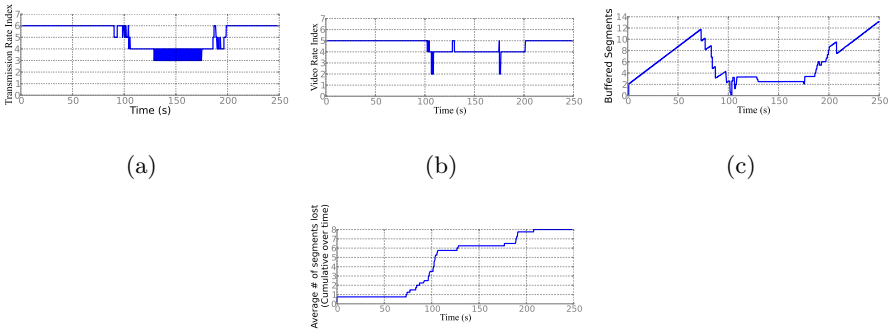
(ii) WiFi AP: We used ASUS RT-AC56U as an AP. We installed the open source DD-WRT firmware on the AP to provide multicast rate changes from the application layer. In all experiments described in the following section, we utilized broadcast transmissions to avoid the complexity of multicast group management at the AP.

(iii) Receivers: We used 4 Samsung Galaxy tablets as receivers. We used the Vitamio media player library as the video player, since it is open source and supports UDP streaming and native MPEG decoding. This allowed us to include the modules for channel statistics evaluation within the video player. We observed that implementing complex application-layer error correction schemes leads to video packets missing their decoding deadlines, which in turn leads to poor video quality. This issue can be resolved by a native and optimized implementation of an error correction scheme. However, our focus is on evaluating video segment level performance. Thus, we considered a segment transmission successful, if 85% of its packets are received and unsuccessful otherwise. All receivers have the kernel socket buffers set to large values to avoid packet drops at high transmission rates.

4.6.4 Experimental Evaluation

We evaluated the performance of our system architecture and the *DYVR* algorithms on the testbed described in the pervious section. In all our experiments, we used the 5GHz channel and the 802.11a standard. Our experiments consisted of indoor lab environments and we focus on the following settings: (a) *Near* - all receivers are near the AP and randomly placed in the lab, (b) *Far* - all receivers are far from the AP and randomly placed, and (c) *Mobility* - all receivers are mobile. In all the experiments, we set the constraints on segments lost, buffer underflows, and video rate switches to 2%.

Fig. 4.7 shows one instance of a trace obtained when the system operates with the *DYVR-A* Algorithm with 4 receivers initially near the AP, then slowly move away from the



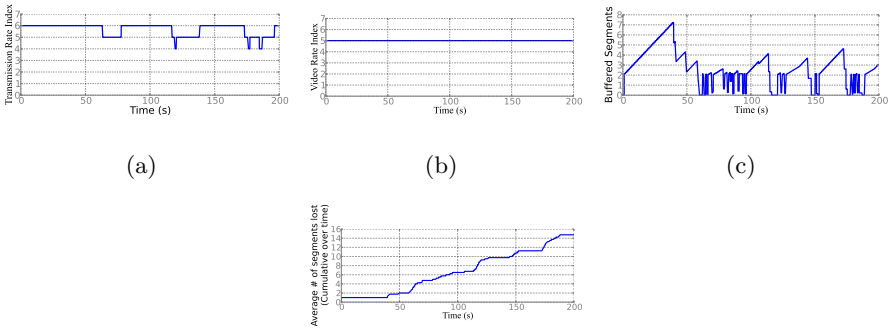
(a) (b) (c)

(d)

Figure 4.7: Experimental trace of an instance of the *DYVR-M* adaptation process during 250s consisting of 4 receivers with all 4 receivers moving away from the AP and then back: (a) transmission rate index, (b) video rate index, (c) buffer at one of the receivers, and (d) average number of segments lost (cumulative over time).

AP, and finally move back close to the AP. A demo video of the experiment is available at [104]. Figs. 4.7(a) and 4.7(b) show the evolution of the transmission rate and video rate over time. The index corresponding to the transmission rate (values between 6Mbps – 54Mbps) reduces for the period of 120s–160s when the receivers move away. While the video rate index (values between 2.5Mbps – 8.5 Mbps as shown in Fig. 4.5(b)) also reduces for this period, the algorithm is able to avoid the rapid oscillations that occur in the transmission rate by utilizing the buffer. *DYVR-A* is able to maintain non-zero buffer occupancy as shown in Fig. 4.7(c) with only one underflow even when the receivers start moving away from the AP. The average number of segments lost at the receivers over time is shown in Fig. 4.7(d). The average segments lost at the end of experiment is 3% which is close to the QoE requirements.

Fig. 4.8 shows one instance of a trace obtained when the system operates with the *DYVR-A* Algorithm with 4 receivers. Only one receiver is mobile and moves in random mobility patterns. This case is particularly challenging for multicast due to the varying mismatch between channel state of receivers. Figs. 4.8(a) and 4.8(b) show the evolution of the transmission rate and video rate over time. While the transmission rate changes in



(a) (b) (c)

(d)

Figure 4.8: Experimental trace of an instance of the *DYVR-M* adaptation process during 200s consisting of 4 receivers with a single receiver moving in a random mobility pattern: (a) transmission rate index, (b) video rate index, (c) buffer at one of the receivers, and (d) average number of segments lost (cumulative over time).

response to channel conditions, the video rate constantly stays at a high value. The drops in transmission rate occur in response to buffer underflows and segments lost. For instance, at 60s and 120s, the number of buffer underflows in Fig. 4.8(c) and segments lost in FIg. 4.8(d) increases which leads to corresponding drops in transmission rates.

We compare the performance of *DYVR-M* and *DYVR-A* to the following algorithms:

(i) BBA (Buffer Based Adaptation) [113]: BBA is a solely buffer-based video rate adaptation algorithm. BBA has been commercially deployed on Netflix and A/B testing has shown better performance as compared to other approaches. We adapt the BBA Algorithm for multicast. In our implementation, the video rate switching decisions at each slot are dictated by the smallest buffer across all clients.

(ii) PBA (Prediction Based Adaptation) [234]: The PBA Algorithm relies on both bandwidth predictions and buffer conditions to tune the video rate. The bandwidth estimates used in PBA were shown to improve video QoE. We also adapted PBA for wireless multicast. The video rate switching decisions at each slot are dictated by the smallest buffer across all clients and the predicted channel state at the receiver with weakest channel quality.

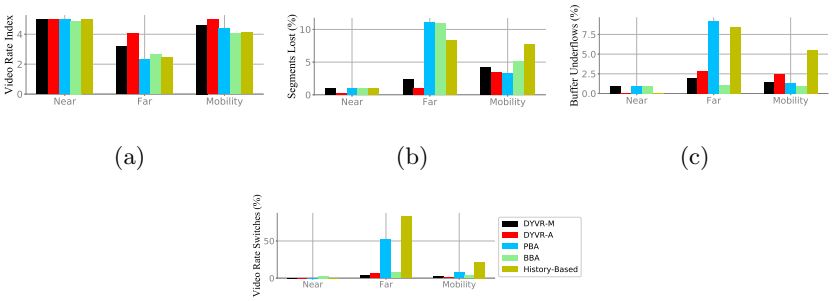


Figure 4.9: Experimental results for different algorithms under various experimental scenarios with 4 tablets: (a) average video rate, (b) percentage of segments lost, (c) percentage of buffer underflows, and (d) percentage of video rate switches.

(iii) History-Based: We use a simple heuristic for video adaptation that sets the video rate to the maximum rate at which all receivers are expected to successfully receive the video segment in the next slot. The number of buffer underflows and video rate switches are expected to be significantly higher with this scheme.

The BBA, PBA, and History-Based algorithms provide a mechanism for tuning only the video rate. These algorithms provided significantly lower video rate when the transmission rate is tuned such that the receivery with the weakest channel quality always receives video segments. To ensure a fair comparison with *DYVR-M* and *DYVR-A*, we used the same channel state estimates for the *DYVR* algorithms and tuned the transmission rate such that an expected fraction f of the receivers will successully receive a segment in each slot. We tuned f such that the average transmission rate achieved for all 3 algorithms is close to the one achieved by *DYVR-M* or *DYVR-A* in various scenarios. Such tuning is hard to achieve in practice and provides a best-case comparison.

Fig. 4.9 shows the comparison performance of *DYVR* and other algorithms for different experimental scenarios. For each approach, we conducted 5 experiments on different days with each experiment's duration about 250s. The average video rate index is shown in Fig. 4.9(a). The average number of segments lost, buffer underflows, and video rate switches are illustrated in Figs. 4.9(b), 4.9(c), and 4.9(d), respectively.

For the near scenario, all algorithms perform comparably and yield the number of segments lost, buffer underflows, and video rate switches within the 2% constraint required. This is because all the schemes are able to transmit at a high transmission rate without incurring any losses. However, for the far case, both *DYVR-M* and *DYVR-A* yield higher video rate than other schemes. The number of segments lost for both *DYVR-M* and *DYVR-A* is less than 1% which is 5x less than other schemes. The number of losses for PBA, BBA, and History-based algorithms can be reduced by setting the transmission rate more conservatively. However, this will further reduce the video rate for these schemes.

The *DYVR-M* Algorithm yields lower video rate than *DYVR-A* on an average but also leads to lower buffer underflows and video rate swtiches than *DYVR-A*. This is due to strict per-receiver constraints in *DYVR-M*. The BBA Algorithm satisfies the constraints on buffer underflows and video rate switches. On the other hand, the PBA Algorithm yields more than 8% buffer underflows and 50% video rate switches. This is because PBA is more aggressive in using more optimistic channel estimates to set higher video rates.

For the mobility scenario, both *DYVR-M* and *DYVR-A* achieve higher video rate than other schemes. *DYVR-M*, *DYVR-A*, and PBA algorithms lead to almost half the number of segment losses compared to BBA. The percentages of buffer underflows and video rate switches for *DYVR-M* and *DYVR-A* are comparable to PBA. In this case, PBA better leverages the channel estimates and leads to higher video rate than BBA while ensuring lower segments lost, buffer underflows, and video rate switches. Similar to the previous case, the *DYVR-M* Algorithm yields slightly lower video rate than *DYVR-A* on an average but also leads to lower buffer underflows and video rate swtiches than *DYVR-A*.

In summary, *DYVR-M* and *DYVR-A* are able to stream high quality video while meeting the QoE constraints in diverse conditions. *DYVR-M* and *DYVR-A* can provide higher video rate than other video rate adaptation approaches even when they have been tuned offline for optimal performance.

Chapter 5

DYNAMIC MONITORING OF LARGE SCALE LTE-eMBMS

5.1 Introduction

So far we focused on WiFi multicast in Chapter 2 and 3. In this chapter, we consider LTE cellular networks. Analogous to WiFi, unicast video streaming over LTE to a large user population in crowded venues requires a dense deployment of Base Stations (BSs) [87, 124, 201]. Such deployments require high capital and operational expenditure and may suffer from extensive interference between adjacent BSs.

LTE-eMBMS (evolved Multimedia Broadcast/Multicast Service) [27, 135] provides an alternative method for content delivery in crowded venues which is based on broadcasting to a large population of User Equipment (UEs) (a.k.a. eMBMS receivers). As illustrated in Fig. 5.2, in order to improve the Signal-to-Noise Ratio (SNR) at the receivers, eMBMS utilizes soft signal combining techniques.[1] *Thus, a large scale Modulation and Coding Scheme (MCS) adaptation should be conducted simultaneously for all the BSs based on the Quality of Service (QoS) at the UEs.*

Unfortunately, the eMBMS standard [27] only provides a mechanism for UE QoS report-

[1] All the BSs in a particular venue transmit identical multicast signals in a time synchronized manner.

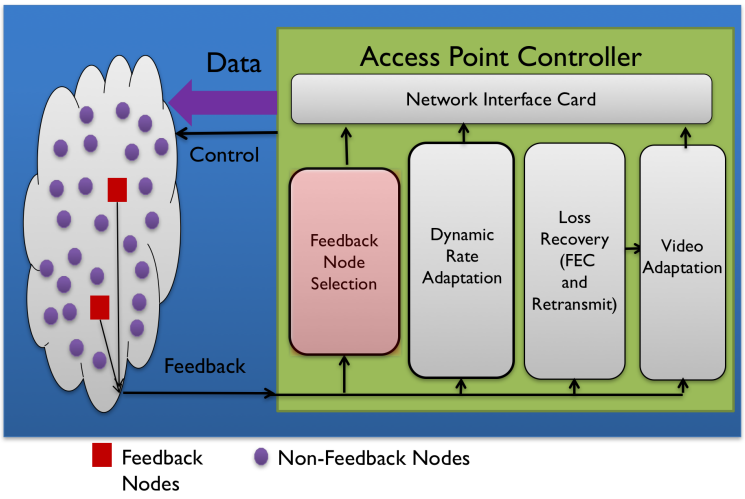


Figure 5.1: An overview of the multicast feedback for LTE-eMBMS.

ing once the communication terminates, thereby making it unsuitable for real-time traffic. Recently, the Minimization of Drive Tests (MDT) protocol [26] was extended to provide eMBMS QoS reports periodically from all the UEs or when a UE joins/leaves a BS. However, in crowded venues with tens of thousands of UEs (e.g., [87]), even infrequent QoS reports by each UE may result in high signaling overhead and blocking of unicast traffic.[2] Due to the limited ability to collect feedback, a deployment of an eMBMS system is very challenging. In particular, it is hindered by the following limitations:

(i) *Extensive and time consuming radio frequency surveys*: Such surveys are conducted before each new eMBMS deployment. Yet, they provide only limited information from a few monitoring nodes.

(ii) *Conservative resource allocation*: The eMBMS MCS and Forward Error Correction (FEC) codes are set conservatively to increase the decoding probability.

(iii) *Oblivious to environmental changes*: It is impossible to infer QoS degradation due to environmental changes, such as new obstacles or component failures.

[2]A BS can only support a limited number of connections while the minimal duration for an LTE connection is in the order of hundreds of milliseconds.

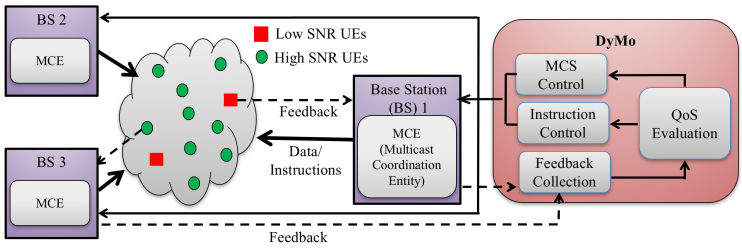


Figure 5.2: The *DyMo* system architecture: It exchanges control information with the Multicast Coordination Entity (MCE) of BSs which use soft signal combining for eMBMS. The Instruction Control module uses broadcast to dynamically partition the UEs into groups, each sending QoS reports at a different rate. The reports are sent to the Feedback Collection module and allow the QoS Evaluation module to identify an SNR Threshold. It is used by the MCS Control module to specify the optimal MCS to the MCEs.

Clearly, there is a need to dynamically tune the eMBMS parameters according to the feedback from UEs. However, a key challenge for eMBMS parameter tuning for large scale groups is *obtaining accurate QoS reports with low overhead*. Schemes for efficient feedback collection in wireless multicast networks have recently received considerable attention, particulalty in the context of WiFi networks (e.g., [89, 101, 207, 214]). Yet, WiFi feedback schemes cannot be easily adapted to eMBMS since unlike WiFi, where a single Access Point transmits to a node, transmissions from multiple BSs are combined in eMBMS. Efforts for optimizing eMBMS performance focus on periodically collecting QoS reports from all UEs (e.g., [64]) but such approaches rely on extensive knowledge of the user population (for more details, see Section 5.2.2).

In this chapter, we present the *Dynamic Monitoring* (*DyMo*) system designed to support efficient LTE-eMBMS deployments in crowded and dynamic environments by providing accurate QoS reports with low overhead. *DyMo* identifies the maximal eMBMS *SNR Threshold* such that only a small number of UEs with SNR below the SNR Threshold may suffer from poor service[3]. To identify the SNR Threshold accurately, *DyMo* leverages

[3]While various metrics can be used for QoS evaluation, we consider the commonly used eMBMS SNR, referred to as SNR, as a primary metric.

the broadcast capabilities of eMBMS for fast dissemination of instructions to a large UE population.

Each instruction is targeted at a sub-group of UEs that satisfies a given condition. It instructs the UEs in the group to send a QoS report with some probability during a reporting interval.[4] We refer to these instructions as *Stochastic Group Instructions*. For instance, as shown in Fig. 5.3, *DyMo* divides UEs into two groups. UEs with poor or moderate eMBMS SNR are requested to send a report with a higher rate during the next reporting interval. In order to improve the accuracy of the SNR Threshold, the QoS reports are analyzed and the group partitions and instructions are dynamically adapted such that the UEs whose SNR is around the SNR Threshold report more frequently. The SNR Threshold is then used for setting the eMBMS parameters, such as the MCS and FEC codes.

From a statistics perspective, *DyMo* can be viewed as a practical method for realizing *importance sampling* [159] in wireless networks. Importance sampling improves the expectation approximation of a rare event by sampling from a distribution that overweighs the important region. With limited knowledge of the SNR distribution, *DyMo* leverages Stochastic Group Instructions to narrow down the SNR sampling to UEs that suffer from poor service and consequently obtains accurate estimation of the SNR Threshold. To the best of our knowledge, this is the first realization of using broadcast instructions for importance sampling in wireless networks.

The *DyMo* system architecture is illustrated in Fig. 5.2. It operates on an independent server and exchanges control information with several BSs supporting eMBMS. The Instruction Control module instructs the different groups to send reports at different rates. The reports are sent via unicast to the Feedback Collection module and allow the QoS Evaluation module to identify an accurate SNR Threshold. The SNR Threshold is determined such that only a predefined number of UEs with SNR below the threshold, termed as *outliers*, may suffer from poor service. The MCS Control module utilizes the SNR Threshold to configure the eMBMS parameters (e.g., MCS) accordingly. Finally, the QoS Evaluation

[4] A higher probability results in a higher reporting rate, and therefore, we will use rate and probability interchangeably.

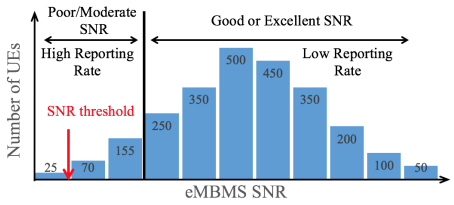


Figure 5.3: Operation of *DyMo* for a sample UE QoS distribution: UEs are partitioned into two groups based on their SNR and each group is instructed to send QoS reports at a different rate. The partitioning is dynamically adjusted based on the reports to yield more reports from UEs whose SNR is around the estimated SNR Threshold.

module continually refines group partitions based on the reports.

We focus on the QoS Evaluation module and develop a *Two-step estimation* algorithm which can efficiently identify the SNR Threshold as a one time estimation. We also develop an *Iterative estimation* algorithm for estimating the SNR Threshold iteratively, when the distribution changes due to UE mobility or environmental changes, such as network component failures. Our analysis shows that the *Two-step estimation* and *Iterative estimation* algorithms can infer the SNR Threshold with a small error and limited number of QoS reports. It is also shown that they outperform the *Order-Statistics estimation* method, a well-known statistical method, which relies on sampling UEs with a fixed probability. For instance, the *Two-step estimation* requires only 400 reports when estimating the $1th$ percentile to limit the error to 0.3% for each re-estimation. The *Iterative estimation* algorithm performs even better than the *Two-step estimation* and the maximum estimation error can be bounded according to the maximum change of SNR Threshold.

We conduct extensive at-scale simulations, based on real eMBMS radio survey measurements from a stadium and an urban area. It is shown that *DyMo* accurately infers the SNR Threshold and optimizes the eMBMS parameters with low overhead under different mobility patterns and even in the event of component failures. *DyMo* significantly outperforms alternative schemes based on the *Order-Statistics estimation* method which rely on random or periodic sampling.

Our simulations show that both in a stadium-like and urban area, *DyMo* detects the

eMBMS SNR value of the 0.1% *percentile with Root Mean Square Error (RMSE) of* 0.05% with only 5 messages per second in total across the whole network. This is at least 8 times better than *Order-Statistics estimation* based methods. *DyMo* also infers the optimal SNR Threshold with RMSE of 0.3 dB regardless of the UE population size, while the error of the best *Order-Statistics estimation* method is above 1 dB. *DyMo* violates the outlier bound (of 0.1%) with RMSE of at most 0.35 while the best *Order-Statistics estimation* method incurs RMSE of over 4 times as compared to *DyMo*. The simulations also show that after a failure, *DyMo* converges instantly (i.e., in a single reporting interval) to the optimal SNR Threshold. Thus, *DyMo* is able to infer the maximum MCS while preserving QoS constraints.

To summarize, the main contributions of this chapter are three-fold:

(i) We present the concept of Stochastic Group Instructions for efficient realization of importance sampling in wireless networks.

(ii) We present the system architecture of *DyMo* and efficient algorithms for SNR Threshold estimation.

(iii) We show via extensive simulations that *DyMo* performs well in diverse scenarios.

The principal benefit of *DyMo* is its ability to infer the system performance based on a low number of QoS reports. It converges very fast to the optimal eMBMS configuration and it reacts very fast to changes in the environment. Hence, it eliminates the need for service planning and extensive field trials. Further, *DyMo* is compatible with existing LTE-eMBMS deployments and does not need any knowledge of the UE population.

The description and evaluation of the *DyMo* system appeared in IEEE INFOCOM'17 [54]. An extended version with additional results and proofs that could not be included in the conference version was fast-tracked to IEEE/ACM Transactions on Networking and the technical report can be found in [55]. The design and evaluation of *DyMo* was based on significant contributions from co-authors at Bell Labs especially, Yigal Bejerano, Chandru Raman, and Chun-Nam Yu.

The rest of the chapter is organized as follows. We provide background information about eMBMS and a brief review of related work in Section 5.2. We introduce the model and objective in Section 5.3. We present the *DyMo* system in Section 5.4. The algorithms

for SNR threshold estimation with their analysis are given in Section 5.5. The numerical evaluation results appear in Section 5.6 while some details of our analysis are given in the Appendix.

5.2 Related Work

5.2.1 eMBMS Background

LTE-Advanced networks provide broadcast services by using *evolved Multimedia Broadcast/Multicast Service* (eMBMS) [135]. eMBMS is best suited to simultaneously deliver common content like video distribution to a large number of users within a contiguous region of cells. eMBMS video distribution is offered as an unidirectional service without feedback from the UE nor retransmissions of lost packets. This is enabled by all cells acting in a coordinated Single Frequency Network (SFN) arrangement, i.e., transmitting identical signals in a time synchronized manner, called *Multicast Broadcast Single Frequency Network* (MBSFN). The identical signals combine over the air in a non-coherent manner at each of the user locations, resulting in an improved Signal-Noise Ratio (SINR). Thus, what is normally out-of-cell interference in unicast becomes useful signal in eMBMS. For avoiding further interference from cells not transmitting the same MBSFN signal, the BSs near the boundary of the MBSFN area are used as a *protection tier* and they should not include eMBMS receivers in their coverage areas.

5.2.2 Related Work

Most previous work on eMBMS (e.g., [68, 154, 193, 226]) assumes individual feedback from all the UEs and proposes various MCS selection or resource allocation techniques. Yet, extensive QoS reports impose significant overhead on LTE networks, which are already highly congested in crowded venues [87]. An efficient feedback scheme was proposed in [64] but it relies on knowledge of path loss (or block error) of the entire UE population to form the set of feedback nodes.

Table 5.1: Notation for *DyMo* model.

Symbol	**Semantics**
m	The number of UEs in the venue, also the number of active eMBMS receiver in static settings.
$m(t)$	The number of active eMBMS receivers at time t.
$h_v(t)$	The individual SNR value of UE v at time interval t.
$s(t)$	The SNR Threshold at time t.
p	*QoS Threshold* - The maximal portion of UEs with individual SNR value $h_v(t) < s(t)$.
r	*Overhead Threshold* - An upper bound on the average number of reports in a reporting interval.

Recently, [220] proposed a multicast-based anonymous query scheme for inferring the maximum MCS that satisfies *all UEs* without sending individual queries. However, the scheme cannot be implemented in current LTE networks, since it will require UEs to transmit simultaneous beacon messages in response to broadcast queries.

Most of the wireless multicast schemes are designed for WiFi networks and a comprehensive survey of WiFi multicast feedback approaches was described in Chapter 2.2. However, WiFi multicast solutions cannot easily be applied to LTE-eMBMS systems. First, in WiFi, each node is associated with an Access Point, and therefore, the Access Point is aware of every node and can specify the feedback nodes. In LTE, eMBMS UEs could be in the idle state and *the network may not be aware of the number of active UEs*. Second, eMBMS is based on simultaneous transmission from various BSs. Thus, unlike in WiFi where MCS adaptation is done at each Access Point independently, a common MCS adaptation should be done at all BSs.

5.3 Model and Objective

5.3.1 Network Model

We consider an LTE-Advanced network with multiple base stations (BSs) providing eMBMS service to a very large group of m UEs in a given large venue (e.g., sports arena, transportation hub).[5] Such venues can accommodate tens of thousands of users. The eMBMS service is managed by a single *DyMo* server as shown in Fig. 5.2 and all the BSs transmit identical multicast signals in a time synchronized manner. The multicast flows contain FEC code that allows the UEs to tolerate some level of losses ℓ (e.g., up to 5% packet losses).

All UEs can detect and report the eMBMS QoS they experience. More specifically, time is divided into short *reporting intervals*, a few seconds each. We assume that the eMBMS SNR distribution of the UEs does not change during each reporting interval.[6] We define the *individual SNR value* $h_v(t)$, such that at least a given percentage $1-\ell$ (e.g., 95%) of the eMBMS packets received by an UE v during a reporting interval t have an SNR above $h_v(t)$. For a given SNR value, $h_v(t)$, there is a one-to-one mapping to an *eMBMS MCS* such that a UE can decode all the packets whose SNR is above $h_v(t)$ [68,154]. The remaining packets ℓ can be recovered by appropriate level of FEC assuming ℓ is not too large. A summary of the main notations used throughout the chapter are given in Table 5.1.

5.3.2 Objective

We aim to design a scalable efficient eMBMS monitoring and control system for which the objective is outlined below and that satisfies the following constraints:

(i) *QoS Constraint* – Given a *QoS Threshold* $p \ll 1$, at most a fraction p of the UEs may suffer from packet loss of more than ℓ. This implies that, *with FEC, a fraction* $1-p$

[5] In this chapter, we consider only the UEs subscribing to eMBMS services.

[6] The SNR of each individual eMBMS packet is a random variable selected from the UE SNR distribution. We assume that this distribution does not change significantly during the reporting interval.

of the UEs should receive all of the transmitted data. We refer to the set UEs that suffer from packet loss after FEC as *outliers* and the rest are termed *normal* UEs.

(ii) *Overhead Constraint* – The average number of UE reports during a reporting interval should be below a given *Overhead Threshold* r.

Objective: *Accurately identify at any given time t the maximum SNR Threshold, $s(t)$ that satisfies the QoS and Overhead Constraints.*
Namely, the calculated $s(t)$ needs to ensure that a fraction $1-p$ of the UEs have individual SNR values $h_v(t) \geq s(t)$.

The network performance can be maximized by using $s(t)$ to calculate the maximum eMBMS MCS that meets the QoS constraint [68, 154]. This allows reducing the resource blocks allocated to eMBMS. Alternatively for a service such as video, the video quality can be enhanced without increasing the bandwidth allocated to the video flow.

5.4 The *DyMo* System

This section introduces the *DyMo* system. It first presents the *DyMo* system architecture, which is based on the *Stochastic Group Instructions* concept. Then, it provides an illustrative example of *DyMo* operations along with some technical aspects of eMBMS parameter tuning.

5.4.1 System Overview

We now present the *DyMo* system architecture, shown in Fig. 5.2.
Feedback Collection: This module operates in the *DyMo* server and in a DyMo *Mobile-Application* on each UE. At the beginning of each reporting interval, the Feedback Collection module broadcasts *Stochastic Group Instructions* to all the UEs. These instructions specify the QoS report probability as a function of the observed QoS (i.e., eMBMS SNR). In response, each UE independently determines whether it should send a QoS report at the current reporting interval.

Table 5.2: Example of the *DyMo* feedback report overhead.

Group	No. of UEs	Report Prob.	Avg. reports per interval	Avg. per sec	Rate per min
H	250	20%	50	5	$\approx 100\%$
L	2250	2%	45	≈ 5	$\approx 12\%$

QoS Evaluation: The UE feedback is used to estimate the eMBMS SNR distribution, as shown in Fig. 5.3. Since the system needs to determine the SNR Threshold, $s(t)$, the estimation of the low SNR range of the distribution has to be more accurate. To achieve this goal, the QoS Evaluation module partitions the UEs into two or more groups, according to their QoS values. This allows *DyMo* to accurately infer the optimal value of $s(t)$, by obtaining more reports from UEs with low SNR. We elaborate on the algorithms for $s(t)$ estimation in Section 5.5.

MCS Control: Since the eMBMS signal is a combination of synchronized multicast transmissions from several BSs, the unicast SNR can be used as a lower bound on the eMBMS SNR. Therefore, the initial eMBMS MCS and FEC are determined from unicast SNR values reported by the UEs during unicast connections. Then, after each reporting interval, the QoS Evaluation module infers the SNR Threshold, $s(t)$, and the MCS Control module determines the desired eMBMS settings, mainly the eMBMS MCS and FEC, according to commonly used one-to-one mappings [68, 154].

5.4.2 Illustrative Example

DyMo operations and the Stochastic Group Instructions concept are demonstrated in the following example. Consider an eMBMS system that serves $2,500$ UEs with the QoS Constraint that at most $p = 1\% = 25$ UEs may suffer from poor service. Assume a reporting interval of 10 seconds. To infer the SNR Threshold, $s(t)$, that satisfies the constraint, the UEs are divided into two groups:

- *High-Reporting-Rate* (H): 10% (250) of UEs that experience poor or moderate service quality report with probability of 20%, i.e., an expected number of 50 reports per interval.

• *Low-Reporting-Rate* (L): 90% (2250) of the UEs that experience good or excellent service quality report with probability of 2%, implying about 45 reports per interval.

Table 5.2 presents the reporting probability of each UE and the number of QoS reports per reporting interval by each group. It also shows the number of QoS reports per second and the reporting rate per minute (i.e., the expected fraction of UEs that send QoS reports in a minute). Since the QoS Constraint implies that only 25 UEs may suffer from poor service, these UEs must belong to group H. Although only 10 QoS reports are received at each second, all the UEs in group H send QoS reports at least once a minute. Thus, the SNR Threshold can be accurately detected within one minute.

5.4.3 Dynamic eMBMS Parameter Tuning

Besides the MCS, *DyMo* can leverage the UE feedback and the calculated SNR Threshold, $s(t)$, for optimizing other eMBMS parameters including FEC, video coding and protection tier. While this aspect is not the focus of this study, we briefly discuss the challenges and the solutions for dynamic tuning of the eMBMS parameters.

Once the SNR Threshold $s(t)$ is selected, *DyMo* tunes the eMBMS parameters accordingly. Every time *DyMo* changes the eMBMS parameters, the consumption of wireless resources for the service is affected as well. For instance, when the eMBMS MCS index is increased, some of the wireless resources allocated for eMBMS are not needed and can be released. Alternatively, the service provider may prefer to improve the video quality by instructing the content server to increase the video resolution. Similarly, before the eMBMS MCS index is lowered, the wireless resources should be increased or the video resolution should be reduced to match the content bandwidth requirements to the available wireless resources.

Since the eMBMS signal is a soft combination of the signals from all BSs in the venue, any change of eMBMS parameters must be synchronized at all the BSs to avoid interruption of service. The fact that all the clocks of the BSs are synchronized can be used and a scheme similar to the *two phase commit* protocol (which is commonly used in distributed databases [191]) can be used.

5.5 Algorithms for SNR Threshold Estimation

This section describes the algorithms utilized by *DyMo* for estimating the SNR Threshold, $s(t)$, for a given QoS Constraint, p and Overhead Constraint r. In particular, it addresses the challenges of partitioning the UEs into groups according to their SNR distribution as well as determining the group boundaries and the reporting rate from the UEs in each group, such that the overall estimation error of $s(t)$ is minimized. We first consider a static setting with fixed number of eMBMS receivers, m, where the SNR values of UEs are fixed. Then, we extend our solution to the case of dynamic environments and UE mobility.

5.5.1 Order Statistics

We first briefly review a known statistical method in quantile estimation, referred to as *Order-Statistics estimation*. It provides a baseline for estimating $s(t)$ and is also used by *DyMo* for determining the initial SNR distribution in its first iteration assuming a single group. Let $F(x)$ be a Cumulative Distribution Function (CDF) for a random variable X, the quantile function $F^{-1}(p)$ is given by, $\inf\{x \mid F(x) \geq p\}$.

Let $X_1, X_2, \ldots, X_r$ be a sample from the distribution F, and F_r its empirical distribution function. It is well known that the empirical quantile $F_r^{-1}(p)$ converges to the population quantile $F^{-1}(p)$ at all points p where F^{-1} is continuous [205]. Moreover, the true quantile, $\hat{p} = F(F_r^{-1}(p))$, of the empirical quantile estimate $F_r^{-1}(p)$ is asymptotically normal [205] with mean p and variance

$$\mathbb{V}\text{ar}[\hat{p}] = \frac{p(1-p)}{r} \tag{5.1}$$

For SNR Threshold estimation, F is the SNR distribution of all UEs. A direct way to estimate the SNR Threshold $s(t)$ is to collect QoS reports from r randomly selected UEs, and calculate the empirical quantile $F_r^{-1}(p)$ as an estimate.[7]

[7]Note that F can have at most m points of discontinuity. Therefore, we assume p is a point of continuity for F^{-1} to enable normal approximation. If the assumption does not hold, we can always perturb p by an

5.5.2 The Two-Step Estimation Algorithm

We now present the *Two-step estimation* algorithm that uses two groups for estimating the SNR Threshold, $s(t)$, in a static setting. We assume a fixed number of UEs, m, and a bound r on the number of expected reports. By leveraging *Stochastic Group Instructions*, *DyMo* is not restricted to collecting reports uniformly from all UEs and can use these instructions to improve the accuracy of $s(t)$. One way to realize this idea is to perform a two-step estimation that approximates the shape of the SNR distribution before focusing on the low quantile tail. The *Two-step estimation* algorithm works as follows:

Algorithm 1: Two-Step Estimation for the Static Case

1. Select p_1 and p_2 such that $p_1 p_2 = p$. Use p_1 as the percentile boundary for defining the two groups.
2. Select number of reports r_1 and r_2 for each step such that $r_1 + r_2 = r$.
3. Instruct all UEs to send QoS reports with probability r_1/m and use these reports to estimate the p_1 quantile $\hat{x}_1 = F^{-1}_{r_1}(p_1)$.
4. Instruct UEs with SNR value below $\hat{x}_1$ to send reports with probability $r_2/(p_1 \cdot m)$ and calculate the p_2 quantile $\hat{x}_2 = G^{-1}_{r_2}(p_2)$ as an estimation for $s(t)$ (G is the CDF of the subpopulation with SNR below $\hat{x}_1$). (G_{r_2} is the empirical CDF of the subpopulation with SNR below $\hat{x}_1$).

Upper Bound Analysis of the Two-Step Algorithm: To simplify the notation, we use r_1 and r_2 to denote the expected number of reports at each step. From (5.1) we know that

$$\hat{p}_1 = F(\hat{x}_1) \quad and \quad \hat{p}_2 = G(\hat{x}_2)$$

are unbiased estimators of p_1 and p_2 with variance

$$\mathbb{V}\mathrm{ar}[\hat{p}_1] = \frac{p_1(1-p_1)}{r_1} \quad and \quad \mathbb{V}\mathrm{ar}[\hat{p}_2] = \frac{p_2(1-p_2)}{r_2} \tag{5.2}$$

infinitesimal amount to make it a point of continuity for F^{-1}.

Our estimate $\hat{x}_2$ has true quantile $\hat{p}_1\hat{p}_2$. Assume $\hat{p}_1$ is less than $p_1+\epsilon_1$ and $\hat{p}_2$ is less than $p_2+\epsilon_2$ with high probability (for example, we can take ϵ_1 and ϵ_2 to be 3 times the standard deviation for $> 99.8\%$ probability). Then, the over-estimation error is bounded by

$$\begin{aligned}\epsilon &= (p_1+\epsilon_1)(p_2+\epsilon_2) \ - \ p \\ &= p_1p_2+\epsilon_1p_2+\epsilon_2p_1+\epsilon_1\epsilon_2-p \\ &\approx \epsilon_1p_2+\epsilon_2p_1\end{aligned} \tag{5.3}$$

after ignoring the small higher order term $\epsilon_1\epsilon_2$. The case for under-estimation is similar. As shown in the Appendix, the error is minimized by taking,

$$p_1=p_2=\sqrt{p} \quad \textit{and} \quad r_1=r_2=r/2$$

so that

$$\epsilon_1=\epsilon_2=3\sqrt{\sqrt{p}(1-\sqrt{p})/(r/2)}$$

This leads to proposition 2.

Proposition 2. *The distance between p and the quantile of the Two-Step estimator $\hat{x}_2$, $\hat{p}=F^{-1}(x_2)$, is bounded by*

$$6\sqrt{2}\sqrt{\frac{p\sqrt{p}(1-\sqrt{p})}{r}}$$

with probability at least $1-2(1-\Phi(3))>99.6\%$, where Φ is the normal CDF.

We now compare this result against the bound of 3 standard deviations in the Order Statistics case, which is $3\sqrt{p(1-p)/r}$. With some simple calculations, it can be easily shown that if $p \leq 1/49 \approx 2\%$, the *Two-step estimation* has smaller error than the *Order-Statistics estimation* method. Essentially the *Order-Statistics estimation* method has an error of order $\sqrt{p}/\sqrt{r}$, while the *Two-step estimation* has an error of order $p^{3/4}/\sqrt{r}$. Since $p \ll 1$, the difference can be significant.

Example: We validated the error estimation of the *Two-step estimation* algorithm and the *Order-Statistics estimation* method by numerical analysis. We considered the cases of $p=1\%$ and $p=0.1\%$ of uniform distribution on $[0,1]$ using $r=400$ samples over population

size of 10^6. The *Two-step estimation* algorithm has smaller standard error compared to the *Order-Statistics estimation*, as shown in Fig. 5.4. Its accuracy is significantly better for very small p.

The *Two-step estimation* algorithm can be generalized to 3 or more telescoping group sizes, but p will need to be much smaller for these sampling schemes in order to reduce the number of samples.

5.5.3 The Iterative Estimation Algorithm

We now turn to the dynamic case in which *DyMo* uses the SNR Threshold estimation $s(t-1)$ from the previous reporting interval to estimate $s(t)$ at the end of reporting interval t. Assume that the total number of eMBMS receivers, m, is fixed and it is known initially.

Suppose that *DyMo* has a current estimate $\hat{x}$ of the SNR threshold, $s(t)$, and $s(t)$ changes over time. We assume that the change in SNR of each UE is bounded over a time period. Formally,

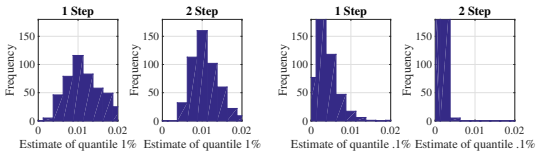
$$|h_v(t_1) - h_v(t_2)| \leq L|t_1 - t_2|$$

where L is a Lipschitz constant for SNR changes. For example, we can assume that the UEs' SNR cannot change by more than 5dB during a reporting interval. [8] This implies that within the interval, only UEs with SNR below $\hat{x} + 5$dB affect the estimation of the p quantile (subject to small estimation error in $\hat{x}$).

DyMo only needs to monitor UEs with SNR below $x_L = \hat{x} + L$. Denote the true quantile of x_L, defined by $F^{-1}(x_L)$, as p_L. To apply a process similar to the second step of the *Two-step estimation* algorithm by focusing on UEs with SNR below x_L, first an estimate of p_L is required. *DyMo* uses the previous SNR distribution to estimate p_L and instructs the UEs to send reports at a rate $q = r/(p_L \cdot m)$. Let Y be the number of reports received during the last reporting interval, then $Y/m \cdot q$ can be used as an updated estimator, $\hat{p_L}$, for p_L. This estimator is unbiased and has variance

$$\mathbb{V}\text{ar}[\hat{p_L}] = \mathbb{V}\text{ar}[\frac{Y}{m \cdot q}] = \frac{p_L}{m}\frac{1-q}{q} \tag{5.4}$$

[8] In our simulations, each reporting interval has a duration of 12s.



(a) (b)

Figure 5.4: Estimates of (a) $p = 1\%$ and (b) $p = 0.1\%$ quantiles for 500 runs for the *Order-Statistics estimation* (1-step) method and the *Two-step estimation* algorithm.

As a result, the Iterative Estimation algorithm works as follows:

Algorithm 2: Iterative Estimation for the Dynamic Case

1. Instruct UEs with SNR below $\hat{x}+L$ to send reports at a rate q. Construct an estimator $\hat{p}_L$ of p_L from the number of received reports Y.
2. Set $p' = p/\hat{p}_L$. Find the p' quantile $x' = G_Y^{-1}(p')$ and report it as the p quantile of the whole population (G is the CDF of the subpopulation with SNR below $\hat{x} + L$).

Upper Bound Analysis of the Iterative Algorithm: Suppose the estimation error of p_L is bounded by ϵ_1, and the estimation error of $p' = p/\hat{p}_L$ is bounded by ϵ_2 with high probability. Then, the estimation error is

$$\epsilon = (\frac{p}{\hat{p}_L} \pm \epsilon_2)p_L - p = (\frac{p}{p_L \pm \epsilon_1} \pm \epsilon_2)p_L - p.$$

The over-estimation error is bounded by

$$\frac{p}{p_L - \epsilon_1}\epsilon_1 + p_L\epsilon_2. \tag{5.5}$$

If we assume $p_L - \epsilon_1 \geq p$ (we know $p_L \geq p$ by the Lipschitz assumption), then the bound can be simplified to $\epsilon_1 + p_L\epsilon_2$. The same bound also works for the under-estimation error. If r denotes also the expected number of samples collected, $r = p_L \cdot m \cdot q$. The standard deviation of $\hat{p}_L$ can be written as:

$$\sqrt{\frac{p_L}{m}\frac{1-q}{q}} = \sqrt{\frac{p_L^2}{r}(1 - \frac{r}{p_L m})} \leq \frac{p_L}{\sqrt{r}}.$$

If we assume $\epsilon_1 = 3p_L/\sqrt{r}$, the error of $\hat{p}_L$ is less than ϵ_1 with probability at least $\Phi(3)$. Since we assume $p_L - \epsilon_1 \geq p$ above, this implies $(1-3/\sqrt{r})p_L \geq p$. If $r \geq 100$, then $p < 0.7p_L$ will satisfy our requirement.

The standard deviation of estimating the $p' = p/\hat{p}_L$ quantile is

$$\sqrt{\frac{1}{Y}\frac{p}{\hat{p}_L}(1-\frac{p}{\hat{p}_L})} \leq \frac{1}{2\sqrt{Y}}, \tag{5.6}$$

by using the fact that $x(1-x) \leq 1/4$ for $x \in [0,1]$ and Y is the number of reports received (a random variable). If the expected number of reports r is reasonably large (≥ 100, say), then Y can be well approximated by a normal and $Y \geq 0.7r$ with high probability $\Phi(3) = 99.8\%$. Then, (5.6) is bounded by $2/(3\sqrt{r}) \geq 1/(2\sqrt{0.7r})$ with high probability ($\Phi(3) = 99.8\%$), and we can set $\epsilon_2 = 2/\sqrt{r}$. Substituting these back into (5.5), gives us the following proposition.

Proposition 3. *The distance between p and the quantile of the estimator x, $\hat{p} = F^{-1}(x)$, is approximately bounded by*

$$5\frac{p_L}{\sqrt{r}}$$

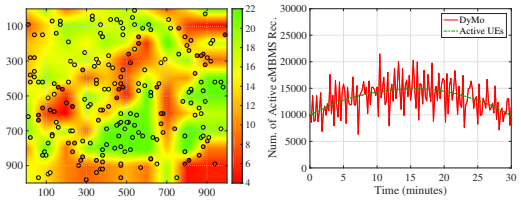
with probability at least $1 - 2(1 - \Phi(3)) > 99.6\%$, if the expected sample size $r \geq 100$ and $p \leq 0.7p_L$.

This shows that the error is of order $p_L/\sqrt{r}$. We can see that the estimation error can be smaller compared to the error of order $p^{3/4}/\sqrt{r}$ in the static *Two-step estimation* if p_L is small (i.e., the SNR of individual users does not change much during a reporting interval).

Exponential Smoothing: *DyMo* applies exponential smoothing by weighing past and current reports to smooth the estimates of the SNR Threshold, $s(t)$, and take older reports into account. It estimates the SNR Threshold as

$$s(t) = \alpha\hat{x}(t) + (1-\alpha)s(t-1)$$

where $\hat{x}(t)$ is the new raw SNR Threshold estimate using the *Iterative estimation* above and $s(t-1)$ is the SNR Threshold from the previous reporting interval. We set $\alpha = 0.5$ to allow some re-use of past reports without letting them have too strong an effect on the estimates (e.g., samples older than 7 reporting intervals have less than 1% weight). *DyMo* also uses



(a) (b)

Figure 5.5: (a) The heatmap of SNR distribution of UEs (b) the evolution of the number of active UEs over time compared to the number estimated by *DyMo* for a homogeneous environment.

the exponential smoothing method for estimating the SNR distribution while taking into account QoS reports from previous reporting intervals.

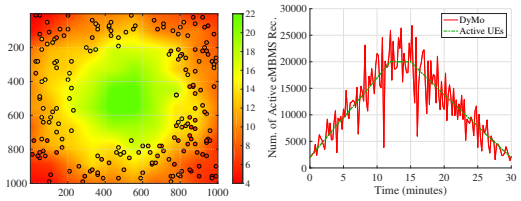
Dynamic and Unknown Number of eMBMS Receivers: If the total number of UEs, $m(t)$, is unknown or changes dynamically, *DyMo* can estimate $m(t)$ by requiring UEs above the threshold $\hat{x} + L$ to send reports. These UEs can send reports at a lower rate, since $m(t)$ is not expected to change rapidly. Similar to the *Two-step estimation* algorithm, *DyMo* allocates $r_1 = r_2 = r/2$ reports to each group. The errors in estimating the total number of UEs $m(t)$ will contribute to the error ϵ_1 in the estimation of p_L in (5.5). The error analysis in this case is largely similar.

5.6 Performance Evaluation

5.6.1 Methodology

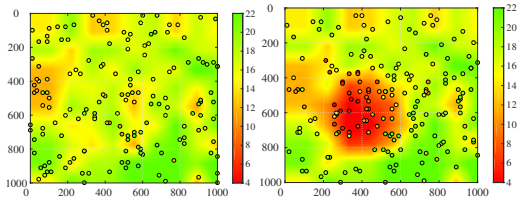
We perform extensive simulations to evaluate the performance of *DyMo* with various values of QoS Constraint, p, Overhead Constraint, r, and number of UEs, m. Our evaluation considers dynamic environments with UE mobility and a changing number of *active eMBMS receivers* denoted by $m(t)$, dynamically selected from the given set of m UEs in the considered venue. In this chapter, we present a few sets of simulation results, which capture various levels of variability of the SNR threshold, $s(t)$, over time.

We consider a variant of *DyMo* where the number of active UEs is unknown and is



(a) (b)

Figure 5.6: (a) The heatmap of UE SNR distribution in a stadium area of $1000 \times 1000m^2$ and (b) the evolution of the number of active UEs over time compared to the number estimated by *DyMo* for a stadium environment.



(a) (b)

Figure 5.7: The heatmap of the SNR distribution of UEs (a) before a failure and (b) after a failure.

estimated from its measurements. We compare the performance of *DyMo* to four other schemes. To demonstrate the advantages of *DyMo*, we augment each scheme with additional information, which is hard to obtain in practice. The evaluated benchmarks are the following:

- ***Optimal*** – Full knowledge of SNR values of the UEs at any time and consequently accurate information of the SNR distribution. *This is the best possible benchmark although impractical, due to its high overhead.*
- ***Uniform*** – Full knowledge of the SNR characteristics at any location while assuming uniform UE distribution and static eMBMS settings. In practice, *this knowledge cannot be obtained even with rigorous field trial measurements.*
- ***Order-Statistics*** – It is based estimation of the SNR Threshold using random sampling. The active UEs send reports with a fixed probability of $r/\mathbb{E}[m(t)]$ per second, assuming that the expected number of active UEs, $\mathbb{E}[m(t)]$, is known. We assume that the UEs are

configured with this reporting rate during initialization. In practice, $\mathbb{E}[m(t)]$ *is not available. We also ignore initial configuration overhead in our evaluation. Order-Statistics* is the best possible approach when not using broadcast messages for UE configuration. We consider two variants of *Order-Statistics*. The first is ***Order-Statistics w.o. History*** which ignores SNR measurements from earlier reporting intervals. The second variant ***Order-Statistics w. History*** considers the history of reports.

Both *DyMo* and *Order-Statistics w. History* perform the same exponential smoothing process for assigning weights to the measurements from previous reporting intervals with a smoothing factor of $\alpha = 0.5$. We use the following metrics to evaluate the performance of the schemes:

(i) *Accuracy* – The accuracy of the *SNR Threshold* estimation, $s(t)$. After calculating $s(t)$ at each reporting interval, we check the actual *SNR Threshold Percentile* in the accurate SNR distribution of the considered scheme. This metric provides the percentile of active UEs with individual SNR values below $s(t)$.

(ii) *QoS Constraint violation* – The number of outliers above the QoS Constraint p. The number of outliers of a scheme in a given reporting interval t is defined as the actual SNR Threshold Percentile of the scheme times the number of active eMBMS receivers, $m(t)$, at time t.

(iii) *Overhead Constraint violation* – The number of reports above the Overhead Threshold, r, at each reporting interval.

The total simulation time for each instance is 30mins with 5 reporting intervals per minute (each is 12s). During each reporting interval, an active UE may send its SNR value at most once. The accuracy of each SNR report is 0.1dB.

5.6.2 Simulated Environments

We simulated a variety of environments with different SNR distributions and UE mobility patterns. Although the simulated environments are artificial, their SNR distributions mimic

those of real eMBMS networks obtained through field trial measurements. To capture the SNR characteristics of an environment, we divide its geographical area into rectangles of $10m \times 10m$. For each reporting interval, each UE draws its individual SNR value, $h_v(t)$, from a Gaussian-like distribution which is a characteristic of the rectangle in which its located. The rectangles have different mean SNR, but the same standard deviation of roughly 5dB (as observed in real measurements). Thus, the SNR characteristics of each environment are determined by the mean SNR values of the rectangles at any reporting interval. To demonstrate the performance of the different schemes, we discuss three types of environments.

• **Homogeneous:** In the homogeneous[9] setting the mean SNR value of each rectangle is fixed and it is uniformly selected in the range of $5 - 25$dB. Fig. 5.5(a) provides an example of the mean SNR values of such a venue as well as typical UE location distribution. In such instances, we assume random mobility pattern, in which each UE moves back and forth between two uniformly selected points. During the simulation, 50% of the UEs are always active, while the other 50% join and leave at some random time, as illustrated by Fig. 5.5(b). As we show later in such setting $s(t)$ barely change over time.

• **Stadiums:** In a stadium, the eMBMS service quality is typically significantly better inside the stadium than in the surrounding vicinity (e.g., the parking lots). To capture this, we simulate several stadium-like environments, in which the stadium, in the center of the venue, has high eMBMS SNR with mean values in the range of $15 - 25$dB. On the other hand, the vicinity has significantly lower SNR with means values of $5 - 10$dB. An example of a stadium is shown in Fig. 5.6(a).

We assume a mobility pattern in which, the UEs move from the edges to the inside of the stadium in 12mins, stay there for 3mins, and then go back to the edges.[10] As shown in Fig. 5.6(b), as the UEs move toward the center, the number of active UEs gradually

[9]We use the term homogeneous since the term uniform is already used to denote the *Uniform* scheme.

[10]While significant effort has been dedicated to modeling mobility (e.g., [175, 186] and references therein), we use a *simplistic mobility model* since our focus is on the multicast aspects rather than the specific mobility patterns.

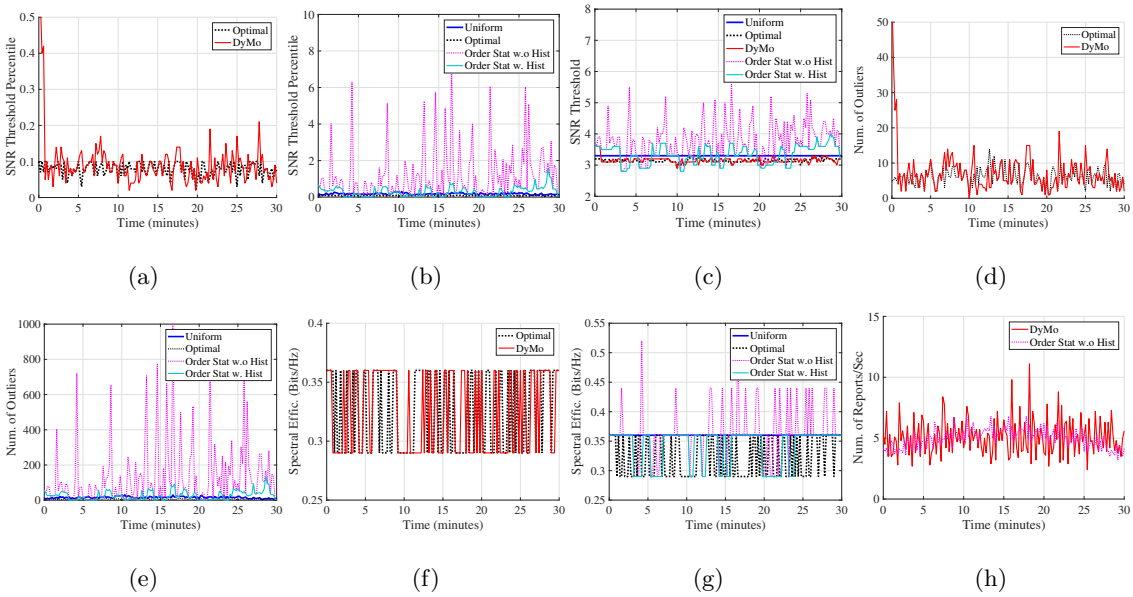


Figure 5.8: Simulation results from a single simulation instance lasting for 30mins in a component homogeneous environment with $20,000$ UEs moving side to side between two random points, with $p = 0.1$ and $r = 5$ messages/sec. (a) The actual percentile of the SNR Threshold estimated by *DyMo*, (b) the actual percentile of the SNR Threshold estimated by *Order-Statistics*, (c) the SNR Threshold estimation, (d) spectral Efficiency of *Optimal* vs. *DyMo*, (e) spectral Efficiency of *Optimal* vs. *Order-Statistics*, (f) the number of Outliers by using *DyMo*, (g) the number of outliers by using *Uniform* and *Order-Statistics*, and (h) the QoS report overhead.

increases from 10% of the UEs to 100%, and then declines again as they move away.

- **Failures:** Such an environment is similar to the homogeneous setting with a sudden event of a component failure. In the case of a malfunctioning component, the QoS in some parts of a venue can degrade significantly. To simulate failures, we consider cases in which the eMBMS SNR is high with a mean between $15 - 25$dB. During the simulation, (around the 10^{th} minute), we mimic a failure by reducing the mean SNR values of some of the rectangles by over 10dB to the range of $5 - 10$dB. The mean SNR values are restored to their original values after a few minutes. Figs. 5.7(a) and 5.7(b) provide an example of the mean SNR values of such a venue before and after a failure, respectively. We assume the same mobility pattern like the homogeneous setting, as shown by Fig. 5.5(b).

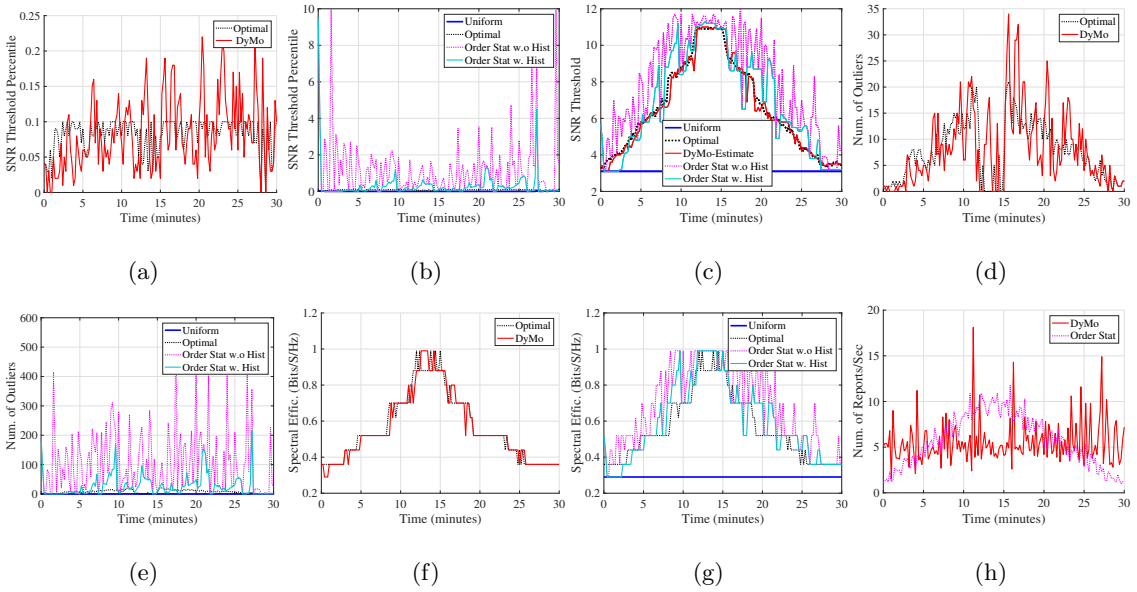


Figure 5.9: Simulation results from a single simulation instance lasting for 30mins in a stadium environment with $20,000$ UEs moving from the edges to the center and back, with $p = 0.1$ and $r = 5$ messages/sec. (a) The actual percentile of the SNR Threshold estimated by *DyMo*, (b) the actual percentile of the SNR Threshold estimated by *Order-Statistics*, (c) the SNR Threshold estimation, (d) spectral efficiency of *Optimal* vs. *DyMo*, (e) spectral efficiency of *Optimal* vs. *Order-Statistics*, (f) the number of Outliers by using *DyMo*, (g) the number of Outliers by using *Uniform* and *Order-Statistics*, and (h) the QoS report overhead.

5.6.3 Performance over time

We first illustrate the performance of the different schemes over time for three given instances, a homogeneous, a stadium and a failure scenarios, with $m = 20,000$ UEs, QoS Constraint $p = 0.1\%$, and Overhead constraint $r = 5$ reports/sec, i.e., 60 messages per reporting interval. The *number of permitted outliers depends on the number of active UEs at the current reporting interval. In the three considered scenarios, it can be at most* 20 *at any given time.* The key difference between the different instances is the rate at which the SNR Threshold changes. In the homogeneous environment the SNR Threshold is almost fixed with very limited variability. In the case of the stadium, the SNR Threshold gradually

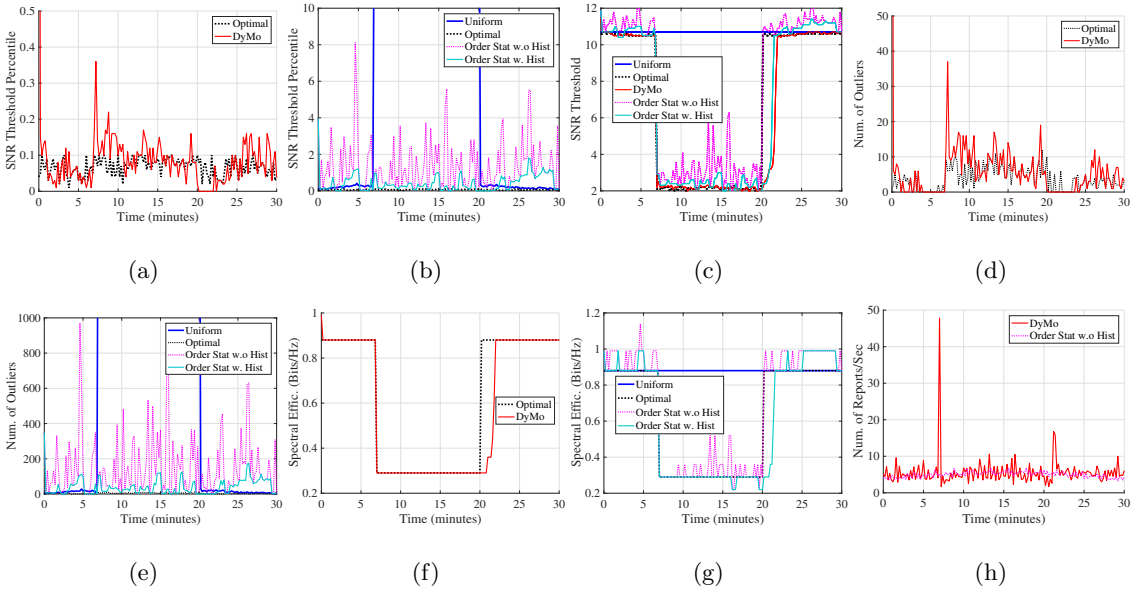


Figure 5.10: Simulation results from a single simulation instance lasting for 30mins in a component failure environment with $20,000$ UEs moving side to side between two random points, with $p = 0.1$ and $r = 5$ messages/sec. (a) The actual percentile of the SNR Threshold estimated by *DyMo*, (b) the actual percentile of the SNR Threshold estimated by *Order-Statistics*, (c) the SNR Threshold estimation, (d) spectral Efficiency of *Optimal* vs. *DyMo*, (e) spectral Efficiency of *Optimal* vs. *Order-Statistics*, (f) the number of Outliers by using *DyMo*, (g) the number of outliers by using *Uniform* and *Order-Statistics*, and (h) the QoS report overhead.

changes as the UEs change their locations. In the failure scenario, the SNR Threshold is roughly fixed but it drops instantly by 10dBs for the duration of the failure.

The results of the homogeneous, stadium and failure cases are shown in Figs. 5.8, 5.9 and 5.10, respectively. Figs. 5.8(a), 5.8(b), 5.9(a), 5.9(b), 5.10(a), and 5.10(b) show the actual SNR Threshold percentile over time. From Figs. 5.8(a), 5.9(a) and 5.10(a), we observe that *DyMo* can accurately infer the SNR Threshold with an estimation error of at most 0.1%. Fig. 5.10(a) shows slightly higher error of 0.25% at the time of the failure (at the 7^{th} minute). The *Order-Statistics* variants suffer from much higher estimation error to the order of a few percentage points, as shown by Figs. 5.9(b), 5.9(b) and 5.9(b)[11]. This

[11]Notice that the pairs (i) Figs. 5.8(a) and 5.8(b), (ii) Figs. 5.9(a) and 5.9(b) as well as (iii)

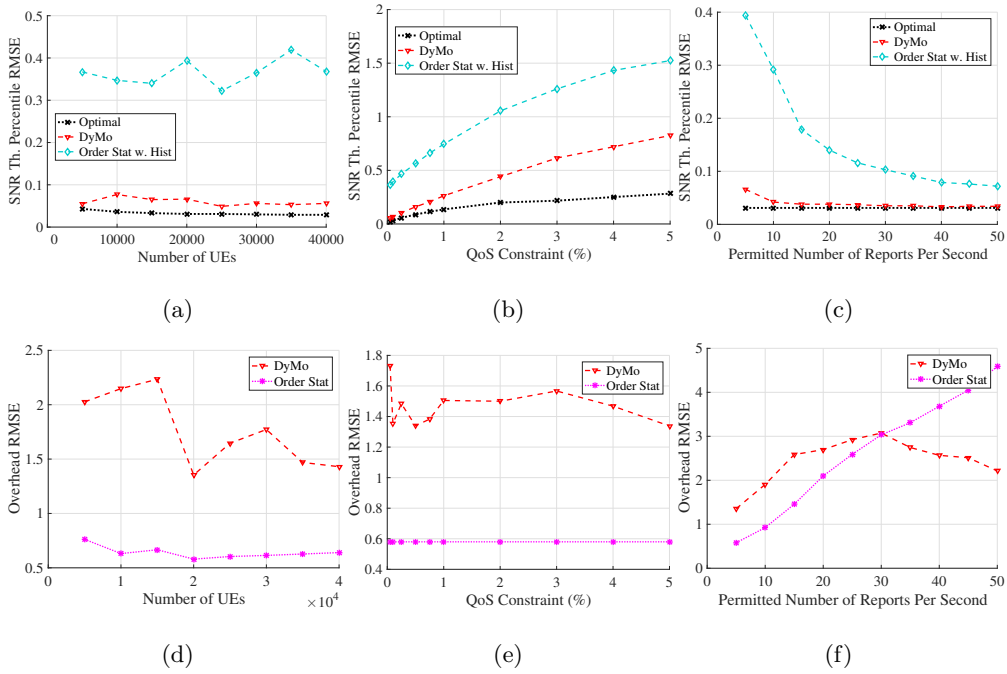


Figure 5.11: The Root Mean Square Error (RMSE) of different parameters averaged over 5 different simulation instances lasting for 30mins each in homogeneous scenario with different SNR characteristics and UE mobility patterns. (a) SNR Threshold percentile RMSE vs. the total number of UEs in the system, (b) SNR Threshold percentile RMSE vs. the QoS Constraint p, (c) SNR Threshold percentile RMSE vs. the number of permitted reports , (d) Overhead RMSE vs. the number of UEs, (e) Overhead RMSE vs. the QoS constraint p, and (f) Overhead RMSE vs. the number of permitted reports.

performance gap results in different estimation accuracy of the SNR Threshold for *DyMo* and *Order-Statistics* schemes as illustrated in Figs. 5.8(c), 5.9(c) and 5.10(c), respectively. These figures show that the performance of *DyMo* and *Optimal* is almost identical. Even in the event of a failure, *DyMo* reacts immediately and detects the SNR Threshold accurately. The *Order-Statistics* variants react quickly to a failure but not as accurately as *DyMo*. After the recovery, both *DyMo* and *Order-Statistics w. History* gradually increase their SNR Threshold estimates, due to the exponential smoothing process.

The SNR Threshold estimation gap directly impacts the number of outliers as well as

Figs. 5.10(a) and 5.10(b) use different scales for the Y axes.

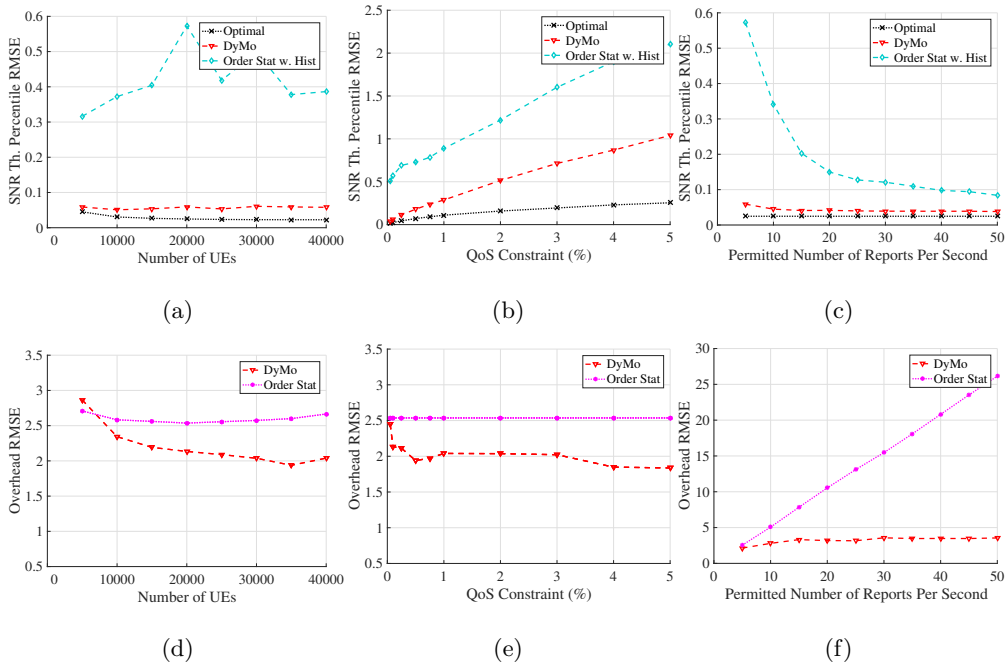


Figure 5.12: The Root Mean Square Error (RMSE) of different parameters averaged over 5 different simulation instances lasting for 30mins each in a stadium environment with different SNR characteristics and UE mobility patterns. (a) SNR Threshold percentile RMSE vs. the total number of UEs in the system, (b) SNR Threshold percentile RMSE vs. the QoS Constraint p, (c) SNR Threshold percentile RMSE vs. the number of permitted reports, (d) Overhead RMSE vs. the number of UEs, (e) Overhead RMSE vs. the QoS constraint p, and (f) Overhead RMSE vs. the number of permitted reports.

the network utilization, i.e., the spectral efficiency. Figs. 5.8(d) and 5.8(e) show the number of outliers of *DyMo* and *Order-Statistics* variants for the homogeneous environment, respectively[12], while Figs. 5.8(f) and 5.8(g) show the spectral efficiency of the schemes. Figs. 5.8(d) and 5.8(f) reveal that after a short adaptation phase *DyMo* converges to the optimal performance, i.e., spectral efficiency, while preserving the QoS constraint. Fig. 5.8(f) show that both *Optimal* and *DyMo* fluctuate between two spectral efficiency levels, 0.29 and 0.36 bit/sec/Hz, which results from oscillatation between two MCS levels 3 and 4. Such oscilla-

[12]Notice that the figure pairs, (i) Figs. 5.8(d) and 5.8(e), (ii) Figs. 5.9(d) and 5.9(e) as well as (iii) Figs. 5.10(d) and 5.10(e), use different scales for the Y axes.

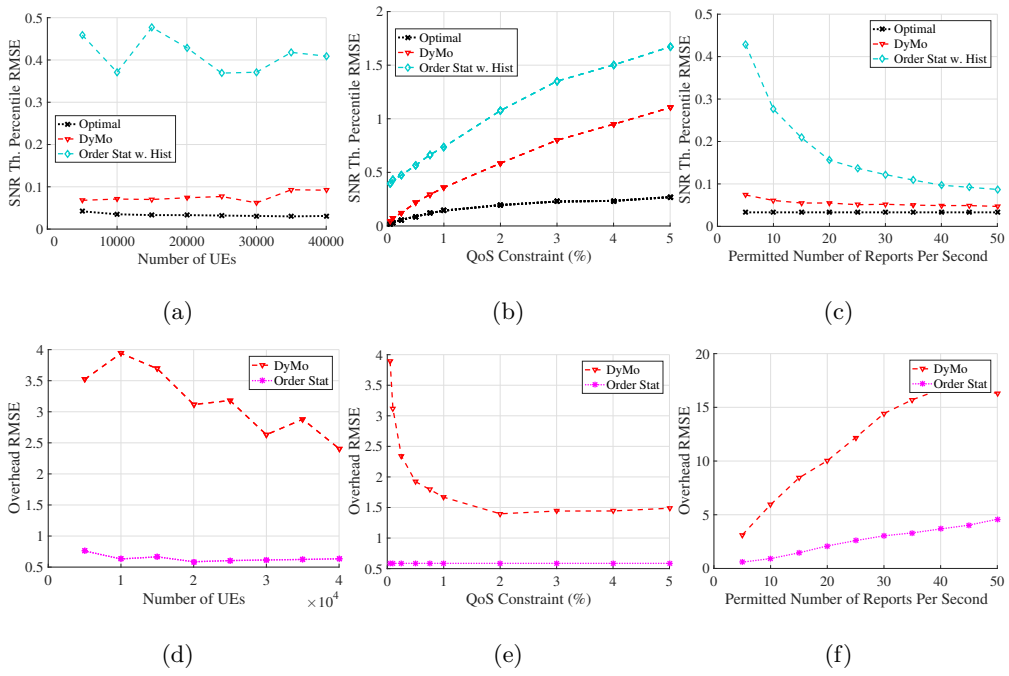


Figure 5.13: The Root Mean Square Error (RMSE) of different parameters averaged over 5 different simulation instances lasting for 30mins each in failure scenario with different SNR characteristics and UE mobility patterns. (a) SNR Threshold percentile RMSE vs. the total number of UEs in the system, (b) SNR Threshold percentile RMSE vs. the QoS Constraint p, (c) SNR Threshold percentile RMSE vs. the number of permitted reports , (d) Overhead RMSE vs. the number of UEs, (e) Overhead RMSE vs. the QoS constraint p, and (f) Overhead RMSE vs. the number of permitted reports.

tions can be easily suppressed by enforcing some delay between MCS increase operations. The *Order-Statistics* variants over estimate the SNR threshold and suffer from higher number of outliers, as shown by Fig. 5.8(e). The homogeneous setting represents quasi-static environments with minor variation of the SNR threshold, $s(t)$. In such settings, the *Uniform* scheme provides a good estimation[13] of $s(t)$ and its number of outliers as well as the obtained spectral efficiency are comparable to *DyMo*. However, this is not the situation when $s(t)$ is time varying.

The number of outliers of *DyMo* and *Order-Statistics* variants for the stadium envi-

[13]Assuming rigorous field trial measurements.

ronment is shown in Figs. 5.9(d) and 5.9(e), respectively, while Figs. 5.10(d) and 5.10(e) illustrate the number of outliers of *DyMo* and *Order-Statistics* variants for the failure scenario, in this order. These figures show that the number of outliers that results from the *Order-Statistics w. History* and *Order-Statistics w.o. History* variants are occasionally over 200 and 800, respectively. Whereas, *DyMo* ensures that the number of outliers at any time is comparable to *Optimal* and in the worst case it exceeds the permitted number by less than a factor of 2.

Figs. 5.9(f) and 5.9(g) show the spectral efficiency for the stadium environment, whereas Figs. 5.10(f) and 5.10(g) show the spectral efficiency for the component failure case. The spectral efficiency for each case is correlated to the SNR Threshold. For the stadium environment, *DyMo* has spectral efficiency close to *Optimal* while *Uniform* has the lowest spectral efficiency. In the event of a failure, the spectral efficiency of *DyMo* follows the *Optimal* as expected from the SNR Threshold estimations. Since *Order-Statistics* variants typically over estimate the SNR Threshold, they frequently determine MCS and consequently spectral efficiency that exceed the optimal settings. Such inaccuracy leads to a high number of outliers.

Figs. 5.8(h), 5.9(h) and 5.10(h) indicate only mild violation of the Overhead Constraint by both the *DyMo* and *Order-Statistics* variants. We observe that accurate SNR Threshold estimation allows *DyMo* to achieve near optimal spectral efficiency with negligible violation of the QoS Constraint. The other schemes suffer from sub-optimal spectral efficiency, excessive number of outliers, or both. Given that the permitted number of outliers is at most 20, the *Order-Statistics w. History* and *Order-Statistics w.o. History* schemes exceed this value sometimes by a factor of 10 and 40, respectively. Among these two alternatives, *Order-Statistics w. History* leads to lower number of outliers. While *Uniform* provides accurate estimation of $s(t)$ for the homogeneous environment, we observe that it yields a very conservative eMBMS MCS setting in the stadium example, which causes low network utilization. In the failure scenario, the conservative eMBMS MCS of *Uniform* is not sufficient to cope with the low SNR Threshold and it leads to excessive number of outliers.

5.6.4 Impact of Various Parameters

We now turn to evaluate the quality of the SNR Threshold estimation and the schemes ability to preserve the QoS and Overhead Constraints under various settings. We use the same configuration of $m = 20,000$ UEs, $p = 0.1\%$ and $r = 5$ reports/sec and we evaluate the impact of changing the values of one of the parameters. The results for the homogeneous, stadium and failure scenarios are shown in Figs. 5.11, 5.12 and 5.13, respectively. Each point in the figures is the average of 5 different simulation instances of 30mins each with different SNR characteristics and UE mobility patterns. The error bars are small and not shown. In these examples, we compare *DyMo* only with *Optimal* and *Order-Statistics w. History* which is the best performing alternative. We omit the *Uniform* scheme since it does not adapt to variation of $s(t)$.

First, we consider the impact of changing these parameters on the accuracy of the SNR Threshold estimation. Figs. 5.11(a), 5.12(a), and 5.13(a) show the Root Mean Square Error (RMSE) in SNR Threshold percentile estimation vs. m, for homogeneous, stadium and failure scenarios, respectively. The non-zero values of RMSE in *Optimal* are due to quantization of SNR reports. The RMSE in the SNR Threshold estimation of *DyMo* is close to that of *Optimal* regardless of the number of UEs, while *Order-Statistics w. History* suffers from order of magnitude higher RMSE.

Figs. 5.11(b), 5.12(b), and 5.13(b) show the RMSE in SNR Threshold estimation as the QoS Constraint p changes, for homogeneous, stadium and failure scenarios. *DyMo* outperforms the alternative schemes as p increases. As p increases, we observe an increasing quantization error, which impacts the RMSE of all the schemes including the *Optimal*. Recall that the SNR distribution is represented by a histogram where each bar has a width of $0.1dB$. As p increases, the number UEs in the bar that contains the p percentile UE increases as well. Since $s(t)$ should be below the SNR value of this bar, we notice a higher quantization error.

Figs. 5.11(c), 5.12(c), and 5.13(c) illustrate the SNR Threshold percentile RMSE as the Overhead Constraint is relaxed, for homogeneous, stadium and failure cases, respectively.

The SNR Threshold percentile RMSE of *DyMo* is 0.05% even with Overhead Constraint of 5 reports/sec, while *Optimal* RMSE due to quantization is 0.025%. *DyMo* error slightly reduces by relaxing the Overhead Constraint (*Optimal* error stays 0.25%). Even with 10 times higher reporting rate, *DyMo* significantly outperforms the *Order-Statistics* alternatives. The RMSE in SNR Threshold percentile for *Order-Statistics* is in the order of the required average value of 0.1 even with a permitted overhead of 50 reports/sec, i.e,. 3000 reports per reporting interval. This is a very high overhead on the unicast traffic, since in LTE networks the number of simultaneously open unicast connections is limited, i.e., several hundreds per base station and each connection lasts several hundred msecs even for sending a short update. Unlike the downlink, uplink resources are not reserved for eMBMS systems and utilize the unicast resources. The RMSE of number of outliers is qualitatively similar to the SNR Threshold percentile results.

We also compute the overhead RMSE for different UE population sizes, m, QoS Constraint p, and Overhead Constraints r. The results are shown is sub-figures (d), (e) and (f) of Figs. 5.11, 5.12 and 5.13, respectively. In most cases, the overhead RMSE of *DyMo* is between $1-4$ reports even when the system parameters change. We observe an increase in the overhead RMSE only in failure scenarios when the permitted overhead is relaxed, as shown in Fig. 5.13(f). This is expected immediately after a failure because many more UEs suffer from poor service than *DyMo* estimated. Thus, as the permitted overhead increases also the spike in the number of reports during the first reporting interval after the failure also increases, which results in a gradual increase of the Overhead RMSE[14].

Figs. 5.11 and 5.13 show that the *Order-Statistics* variants experience very low violation of the Overhead Constraint in the homogeneous and Failure scenarios. This is not surprising, since in these scenarios the variation in the number of active eMBMS receivers is very small and this number is roughly $\mathbb{E}[m(t)]$ (the expected number of active eMBMS receivers). As mentioned in Section 5.6.1, this observation is misleading, since we assume that $\mathbb{E}[m(t)]$ is known and we ignore the overhead of configuring the UEs with the proper reporting

[14]Notice that the RMSE metric is sensitive to sporadic but very high error.

rate. Obviously, *the exact number of active receivers,* $\mathbb{E}[m(t)]$*, is unknown in practice.* Furthermore, Fig. 5.12(f) shows that in scenarios with high variation in the number of active receivers, $m(t)$, (like the case in the stadium simulations) the violation of the Overhead Constraint is high and it is amplified as the permitted number of reports, r, increases. This is due to the static reporting rate of *Order-Statistics* despite dynamic changes of the number of active eMBMS receivers. Fig. 5.12(f) confirms that the overhead violation of *Order-Statistics* is very sensitive to the estimation of $\mathbb{E}[m(t)]$ and its variance.

Given that the number of active eMBMS receivers, $m(t)$*, is unknown and may change significantly over time,* Order-Statistics *cannot practically preserve the Overhead Constraint without keeping track of the active UEs and sending individual messages to a subset of the active UEs.* However, keeping track of $m(t)$ requires each UE to report when it starts and stops receiving eMBMS services, which may incur much higher overhead than permitted. For instance, in our simulations with $m = 20,000$ UEs, even if such switching occurs at most once (start and stop) by each UE, the total number of reports is $40,000$. When dividing this number by the simulation duration of 30 minutes ($1,800$ sec) we get 22 messages/second, which is much higher than the permitted overhead.

Summary: Our simulations show that *DyMo* achieves accurate, close to optimal, estimation of the SNR Threshold even when the number of active eMBMS receivers is unknown. It can improve the spectral efficiency for eMBMS operation, while adding a very low reporting overhead. *DyMo* can predict the SNR Threshold with lower errors than other alternatives under a wide range of the SNR Threshold requirement p and reporting Overhead Constraint r. These observations show that *DyMo* exceeds the expectations of our analysis in Section 5.5.

5.7 Conclusion

This chapter presents a *Dynamic Monitoring* (*DyMo*) system for large scale monitoring of eMBMS services, based on the concept of *Stochastic Group Instructions*. Our extensive simulations show that *DyMo* achieves accurate, close to optimal, estimation of the SNR

Threshold even when the number of active UEs is unknown. It can improve the spectral efficiency for eMBMS operation while adding a low reporting overhead.

5..1 Analysis of the Two-Step Estimation Algorithm

We now extend the analysis of the *Two-step estimation* algorithm given in Section 5.5.2. We show that the optimal settings for minimizing the error ϵ of Equation (5.3) is obtained by taking

$$p_1 = p_2 = \sqrt{p} \quad and \quad r_1 = r_2 = r/2$$

Notice that the settings should satisfy the following two constraints:

$$p = p_1 \cdot p_2 \tag{5.7}$$

and

$$r = r_1 + r_2 \tag{5.8}$$

From Equation (5.2) and by taking 3 times the standard deviation, we get that the errors ϵ_1 and ϵ_2 are

$$\epsilon_1 = 3\sqrt{\frac{p_1(1-p_1)}{r_1}} \quad and \quad \epsilon_2 = 3\sqrt{\frac{p_2(1-p_2)}{r_2}}$$

By combining with Equation (5.3), we get

$$\epsilon = 3\ \sqrt{\frac{p_1(1-p_1)}{r_1}}\ p_2 + 3\ \sqrt{\frac{p_2(1-p_2)}{r_2}}\ p_1 \tag{5.9}$$

By using the two constraints (5.7) and (5.8), we assign $p_2 = p/p_1$ and $r_2 = r - r_1$. Consequently,

$$\begin{aligned} \epsilon &= 3\ \sqrt{\frac{p_1(1-p_1)}{r_1}}\ \frac{p}{p_1} + 3\ \sqrt{\frac{(p/p_1)(1-p/p_1)}{r-r_1}}\ p_1 \\ &= 3\ p\left[\sqrt{\left(\frac{1}{p_1}-1\right)\ \frac{1}{r_1}}\ + \sqrt{\left(\frac{p_1}{p}-1\right)\ \frac{1}{r-r_1}}\right] \end{aligned} \tag{5.10}$$

By taking the partial derivative $\frac{\partial \epsilon}{\partial p_1}$ we get,

$$\frac{\partial \epsilon}{\partial p_1} = 3\ p \left[-\left(2\ p_1^2 \sqrt{\left(\frac{1}{p_1} - 1\right)\ \frac{1}{r_1}} \right)^{-1} + \left(2\ p \sqrt{\left(\frac{p_1}{p} - 1\right)\ \frac{1}{r - r_1}} \right)^{-1} \right] \tag{5.11}$$

For minimizing the error we calculate $\frac{\partial \epsilon}{\partial p_1} = 0$ and get that

$$p_1^2 \sqrt{\left(\frac{1}{p_1} - 1\right) \frac{1}{r_1}} = p \sqrt{\left(\frac{p_1}{p} - 1\right)\ \frac{1}{r - r_1}} \tag{5.12}$$

By simple mathematical manipulations we get

$$p_1^4\ \left(\frac{1}{p_1} - 1\right)(r - r_1) = p^2\ \left(\frac{p_1}{p} - 1\right)\ r_1 \tag{5.13}$$

Similarly, from the partial derivative $\frac{\partial \epsilon}{\partial r_1}$ we get

$$\frac{\partial \epsilon}{\partial r_1} = 3\ p \left[\sqrt{\left(\frac{1}{p_1} - 1\right)}\ \frac{-1}{2\ r_1^{3/2}} + \sqrt{\left(\frac{p_1}{p} - 1\right)}\ \frac{1}{2\ (r - r_1)^{3/2}} \right] \tag{5.14}$$

For minimizing the error we calculate $\frac{\partial \epsilon}{\partial r_1} = 0$ and get that

$$\sqrt{\left(\frac{1}{p_1} - 1\right)}\ \frac{1}{2\ r_1^{3/2}} = \sqrt{\left(\frac{p_1}{p} - 1\right)}\ \frac{1}{2\ (r - r_1)^{3/2}} \tag{5.15}$$

By simple mathematical manipulations we get

$$\left(\frac{1}{p_1} - 1\right)\ (r - r_1)^3 = \left(\frac{p_1}{p} - 1\right)\ r_1^3 \tag{5.16}$$

Noticed that Equations (5.13) and (5.16) together provide two simple conditions to optimize p_1 and r_1. By dividing Equation (5.13) by Equation (5.16) we obtain,

$$(r - r_1) = \frac{r_1\ p_1^2}{p} \tag{5.17}$$

Using Equation (5.17) in Equation (5.16) results that

$$\begin{gathered} \left(\frac{1}{p_1} - 1\right)\ \left(\frac{r_1\ p_1^2}{p}\right)^3 = \left(\frac{p_1}{p} - 1\right)\ r_1^3 \\ \left(\frac{1}{p_1} - 1\right)\ \frac{p_1^6}{p^2} = \left(\frac{p_1}{p} - 1\right) \end{gathered} \tag{5.18}$$

From this we get the following relation

$$p_1^6 - p_1^5 + p_1\ p^2 - p^3 = 0 \tag{5.19}$$

The only real solutions are $p_1 = \pm\sqrt{p}$. Since p_1 must be positive we get that $p = \sqrt{p}$. From this solution and Equation (5.17), it is implies that the optimal setting is

$$p_1 = p_2 = \sqrt{p}, \quad and \quad r_1 = r_2 = r/2$$

Consequently, the errors of the two steps are

$$\epsilon_1 = \epsilon_2 = 3\sqrt{\sqrt{p}(1-\sqrt{p})/(r/2)}$$

From this we obtain Proposition 2 and a bound on the error of,

$$6\sqrt{2}\sqrt{\frac{p\sqrt{p}(1-\sqrt{p})}{r}}$$

This concludes our analysis of the *Two-step estimation* algorithm.

Part II

Datacenter Networks

Chapter 6

OPTICAL MULTICAST FOR DATACENTER NETWORKS

In Part I of this thesis, we focused on wireless networks, more specifically, enabling scalable multicast in wireless networks. In this chapter, we focus on the problem of physical layer optical multicast in datacenter networks. Similar to wireless networks, the workload in datacenter networks is evolving and a large fraction consists of one-to-many traffic patterns. While IP multicast is an attractive option to reduce bandwidth consumption in datacenter networks, most datacenters rely on sequential unicast transmissions for transmitting one-to-many traffic [151]. The main challenge in enabling multicast is the the complexity of configuration at switches and router and lack of scalability [84].

We explore using advanced functionalities of optics for enabling multicast. Optical layer multicast [176] in the context of transport networks, has been used by researchers to increase the logical connectivity of the network and decrease hop distances at the routing nodes [178]. It is achieved either by passive splitting or frequency conversion in a periodically poled lithium niobate (PPLN) [184]. The main challenge in implementing an end-to-end system containing optical modules in data center networks, is the control and management integration with conventional data center networks. We address these challenges and propose a practical hardware-software architecture that enables optical multicast

and evaluate its performance through simulations and experiments.

Finally, this research was done in collaboration with Lightwave Research Laboratory, Columbia University with significant contributions from Payman Samadi and Keren Bergman. The hardware testbed for the evaluations was entirely built by collaborators at Lightwave Research Laboratory. The work presented in this chapter appeared as a poster in the proceedings of ACM SIGCOMM'14 [181] and in the proceedings of European Conference on Optical Communication'14 (ECOC) [182]. A paper with details about system design and evaluation appeared in the journal Optics Express [183].

6.1 Introduction

Traffic in cloud computing data centers has shifted in recent years from predominantly (80%) outbound (north-to-south) to mostly (70%) rack-to-rack (east-to-west) pattern [4,127]. This increase in rack-to-rack traffic that is also the case for High Performance Computing (HPC) has introduced complex patterns involving several nodes with large flow sizes such as multicast i.e., transmitting identical data from one-to-many nodes. Many data center applications that use distributed file systems for storage and MapReduce [81] type of algorithms to process data, inherently require multicast traffic delivery. Distributed file systems use state-machine replication as a fundamental approach to build fault tolerant systems. Many of these systems use Paxos [134] algorithm or its variations to provide strong data consistency guarantees. Paxos-type algorithms entail group communication primitives that are mainly multicast.

For example, Google File System (GFS) [92] uses Chubby [63] that is a Paxos-based system. Ceph [217] is also a distributed file system that relies on Paxos. Similar traffic patterns exist in Hadoop Distributed File System (HDFS) [61] and in the shuffle stage of the MapReduce algorithm. Parallel database join operation [148] includes multicast of several hundred megabytes, and the broadcast phase of Orchestra [74] controlled by Spark [2] involves 300 MB of multicast on 100 iterations [74]. Multicast traffic is also frequent in other data center applications such as Virtual Machine (VM) provisioning [65]

and in-cluster software updating [19] where 300–800 MB of data are transmitted among hundreds of nodes. Additionally, multicast traffic delivery will facilitate one-to-many VM migrations.

Current data center networks do not natively support multicast traffic delivery. Internet Protocol multicast (IP multicast) [166] is the most established protocol for one-to-many transmission in electronic networks. Supporting IP multicast requires complex configurations on all the switches and routers of the data center network [151]. Scaling IP multicast is also a challenge in multi-tier networks with several IP subnets. Due to the scaling and stability issues with IP multicast [84] and the importance of multicast in data centers, there is a growing interest in improving the performance of IP multicast. To improve scalability, [140] proposes a Software Defined Networking (SDN) [130] solution to manage multicast groups. LIPSIN [118] and ESM [138] rely on encoding forwarding states in in-packet Bloom Filters. Datacast [66] introduces an algorithm to calculate multiple edge-disjoint Steiner trees, and then distributes data among them. Despite these efforts, IP multicast is not supported in the majority of current data center networks and multicast traffic is transmitted either through sequence of unicast transmissions or through application layer solutions such as peer-to-peer methods [75]. These methods are inherently inefficient since they send multiple copies of the same data. Furthermore, such multicast traffic typically suffers from excessive latency that increases with the number of receivers as well as large connection overheads.

In conventional data centers, the interconnection network is an Electronic Packet Switching (EPS) network [34, 96]. Due to the switching cost and cabling complexity, providing a non-blocking EPS network in data centers is a challenge and networks are often forced to rely on over-subscription. Optical Circuit Switching (OCS) is data rate transparent and consumes less switching energy [62]. A hybrid architecture as shown in Fig. 6.1, providing OCS and EPS along with an SDN control plane to manage the traffic between them, can deliver a viable co-optimized solution [88,212]. Performing optical switching in data centers would make an immediate improvement in energy efficiency since it eliminates the optical-to-electrical conversions at the switches. Moreover, transmitting larger flows by optical links

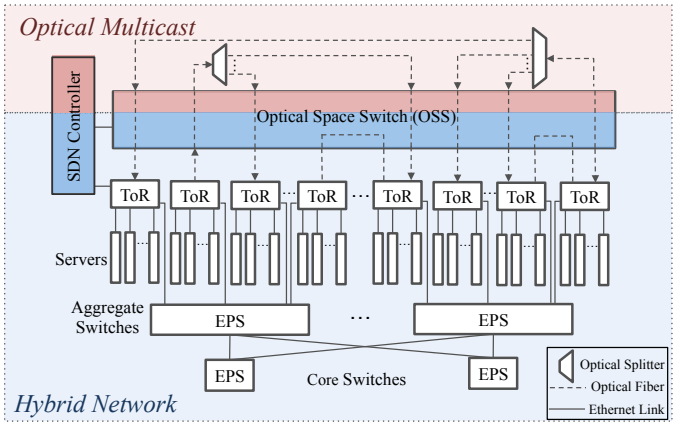


Figure 6.1: Optical multicast system network architecture built over a hybrid network, enabling optical multicast by passive optical splitters and an SDN control plane, (ToR: Top-of-Rack).

decreases the traffic load in the aggregation and core tiers and reduces the total switching cost by allowing higher over-subscription on the EPS network.

In [181, 213], the authors proposed the concept of using optics' advanced functionalities for faster delivery of complex traffic patterns such as multicast, incast and all-to-all-cast over an OCS substrate in data center networks. For example, leveraging passive optical splitters for multicast and time and wavelength multiplexing for incast. Optical layer multicast [176] in the context of transport networks, has been used by researchers to increase the logical connectivity of the network and decrease hop distances at the routing nodes [178]. It is achieved either by passive splitting or frequency conversion in a periodically poled lithium niobate (PPLN) [184].

In contrast to transport networks, the main challenge in implementing an end-to-end system containing optical modules in data center networks, is the control and management integration with conventional data center EPS networks. Moreover, larger multicast groups and faster reconfiguration is necessary. SDN along with cross-layer designs [60, 133] can overcome this critical challenge and provide the optical modules functionalities seamlessly to the higher layers.

In this chapter, we present the design and experimental evaluation of an Optical Multi-

cast System for Data Center Networks; an integrated hardware-software system architecture that enables native physical layer optical multicast in data center networks. We designed an application-driven control plane architecture to i) receive multicast connection requests from the application layer, ii) control the routing of the electronic packet switches, optical circuit switches, and connectivity of optical splitters in the data plane, and iii) optimally assign optical splitters to the flows with a resource allocation algorithm. The hardware architecture (Fig. 6.1) is built on a hybrid network, i.e. the Top-of-Rack switches are simultaneously aggregated by a L2/L3 EPS network and an OCS network provided by an Optical Space Switch (OSS) (OSS is a switching substrate that provides an optical circuit between any idle input and output ports, without optical to electronic conversion [3, 17]). The OSS is also the substrate to connect passive optical splitters to the optical network. The control plane software runs on the SDN controller and communicates with the hosts through the EPS network. We built a prototype to experimentally evaluate the performance of the system and also developed a simulation platform to numerically assess its performance at scale.

Experimental and numerical results show that optical multicast transmits multicast flows simultaneously to all the receivers. It provides similar throughput for delivering multicast flows as IP multicast but i) does not require applying complex configurations on all the switches/routers of the data center to enable IP multicast since multicast trees are directly created by the SDN controller, ii) has superior energy efficiency since it is built on an OCS network that consumes less energy than an EPS network, iii) is future-proof due to the data rate transparency of the system. In addition, optical multicast can be a compliment service to IP multicast for bulk traffic delivery in real-life scenarios that the Ethernet network is highly over-subscribed. Compared to unicast transmissions where the throughput is inversely proportional to the number of receivers, optical multicast have steady performance irrespective to the multicast group size. Compared to peer-to-peer multicast, it provides at minimum an order of magnitude higher throughput for flows with sizes under 250 MB. Also, it results in shorter and fixed connection overhead (OSS reconfiguration time) that is independent of the number of receivers. Furthermore, the optical multicast architecture is

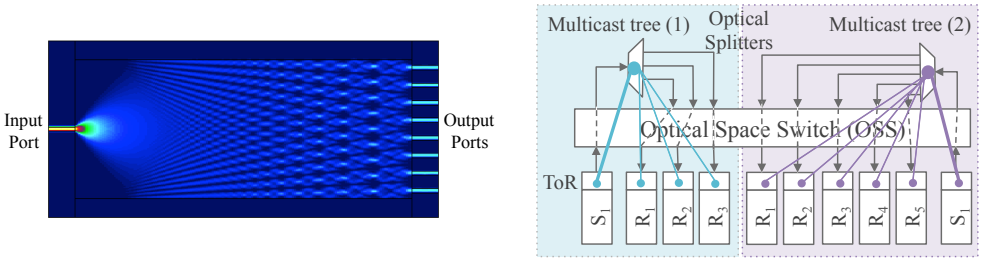
designed to enable direct integration with additional optical modules for optical incast and all-to-all cast functions in data center networks.

Adding the optical multicast system to a data center with a sole non-blocking EPS network decreases the total energy consumption by 50% while delivering 20 TB of data containing 15% multicast flows. The latency also drops by 55%. The improvements are more significant in the case of over-subscribed EPS networks and larger volumes of multicast flows. We also evaluated the resource allocation algorithm with an optimal and greedy heuristic solutions. Our numerical results show that the greedy algorithm is a practical and efficient approach for the control plane. Furthermore, the architecture is designed to enable direct integration with additional optical modules for optical incast and all-to-all cast functions in data center networks.

The rest of this chapter is organized as follows. In Section 6.2, we explain the details of optical multicast, the software and hardware architecture, and the prototype testbed. Section 6.3 shows the evaluation of different components of the control plane including the resource allocation algorithm. Section 6.4 is devoted to experimental and numerical evaluations for multicast traffic delivery as well as the cost and energy efficiency analysis of the architecture. Section 6.5 presents an end-to-end implementation of Ring Paxos on the optical multicast prototype. Section 6.6 explains the potential design to address incast using optical multicast architecture.

6.2 Architecture and implementation

The optical multicast system architecture consists of a 3-layered software component that runs on an SDN controller, and a hardware component built upon an optical circuit switching network. Passive optical splitters are connected to the ports of the OSS and a resource allocation algorithm assigns the splitters to the flows. In this section, we demonstrate the physical layer optical multicast enabled by passive optical splitters. We present the hardware and software architectures, demonstrate the prototype implementation, and discuss its scalability.



(a) (b)

Figure 6.2: (a) Intensity profile of an integrated optical splitter [15], that supports 1:8 optical multicast by dividing the input power, $(P_{out}(i) = \frac{P_{in}}{8}, \forall i = 1, \ldots, 8)$, (b) An example of multicast trees constructed by using passive optical splitters and configuring the OSS ports' connectivity of the senders and receivers ToRs.

Table 6.1: Insertion loss and cost of the commodity passive optical splitters [16].

Splitter Size	Insertion Loss (dB)	Cost ($)
1:4	7.3	14.00
1:8	10.5	18.70
1:16	13.8	36.10
1:32	16.8	62.00
1:64	20.5	132.00

6.2.1 Hardware architecture

Physical Layer Optical Multicast: Physical layer optical multicast is performed by optical splitters. As illustrated in Fig. 6.2(a), these are passive modules that divide the input optical signal to several optical signals by splitting the signal power at predetermined ratios. Optical splitters are manufactured by the Planar Lightwave Circuit (PLC) technology [198] and are commercially available up to 1:128 ratio with the footprint of 2 cm^2 [16]. These splitters are widely used in Passive Optical Networks (PON) [146] for Fiber-To-The-x (FTTx) [125] applications. Table 6.1 shows the insertion loss and the cost of commodity optical splitters.

As shown in Fig. 6.1, the hardware architecture is built on a hybrid network. ToR

switches are simultaneously aggregated by an optical circuit switching network provided by an OSS and an electronic packet switching network provided by a L2/L3 EPS. MEMS-based OSSs provide high port count optical switching substrates without optical to electronic conversions. Integrated OSS also exist with lower port counts but faster switching speed [147].

ToRs are connected to the OSS by point-to-point optical links and copper cables provide connectivity to the electronic packet switch. Integrated optical splitters have fiber connections and are connected to the ports of the OSS. The controller configures the routing of the OSS and ToRs via OpenFlow [153]. Optical splitters are passive and do not require any configuration.

Figure 6.2(b) demonstrates physical layer optical multicast between the racks in a data center network using passive optical splitters. Multicast trees are created between the senders and the receivers by configuring the OSS ports. The sender S_1, is connected to the input port of the splitter and receivers $R_1,\ldots,R_n$ are connected to the output ports. The upper bound for the multicast group size is set by the optical link power budget.

6.2.2 Software architecture

Application-driven Networks: Application-driven networking [47,111,116] and SDN are increasingly used for designing cloud-based data center networks. Big-data applications are also moving in this direction. For example, in Hadoop [218], NameNode and JobTracker are the compute and storage controllers that manage the HDFS and MapReduce tasks, respectively, over the nodes. Global knowledge of application processes and the storage systems, as well as the central management of the network, can be intelligently used to improve the performance of big data applications. While designing the software architecture, we take the application-driven approach, as it seems to be the emerging direction.

Figure 6.3 demonstrates the software architecture consisting of the application, control plane, and the data plane layers. The network controller (including the resource allocation component) is in the control plane layer. It receives multicast traffic requests from the application layer via the northbound API. It configures ToRs, OSS and optical splitters

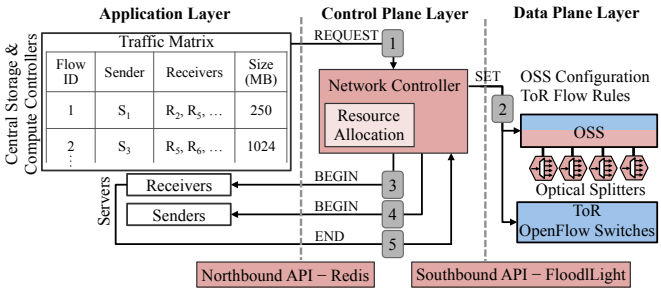


Figure 6.3: The 3-layered software architecture performs: i) Configuration of the OSS and ToRs, and connectivity of the optical splitters in the data plane layer, ii) Receipt of multicast traffic matrix at the application layer from the central compute and storage controllers, iii) Assignment of optical splitters to the flows by a resource allocation algorithm.

connectivity in the data plane accordingly through the southbound API.

The central storage and compute controllers provide the traffic matrix of multicast flows to the network controller. For each flow request, the traffic matrix provides the flow ID, the sender and receivers servers IDs and the flow size. The resource allocation algorithm processes the traffic matrix and assigns flows to the optical splitters with the goal of maximizing the traffic offloaded to the optical multicast system (see Section 6.2.3). The output of the resource allocation algorithm is a set of configurations corresponding to the selected flows and the splitters. The network controller sets the configurations to the ToRs and OSS through the Southbound API. Once the configurations are finished, the network controller notifies the servers involved in the scheduled flows to begin the transmission through the northbound API. Upon completing the transmission, the receivers notify the network controller and it releases the splitters and servers involved in the scheduled flows. The traffic matrix is updated with flows from the applications and the resource allocation algorithm selects the next set of flows. Servers can also request for multicast connection via Northbound API. The network controller aggregates multicast flows between the servers that share the same rack and inserts them in the traffic matrix.

6.2.3 Resource allocation algorithm

The objective of the resource allocation algorithm is to maximize the multicast traffic offloaded to the optical multicast system under the constraint of limited number of splitters. Depending upon the reallocation strategy, the resource allocation algorithm allocates flows to the splitters when a certain number of splitters are available. The reallocation strategy can be myopic (immediate reallocation of free splitters) or far-sighted (wait for a large number of free splitters before reallocation).

We model the resource allocation problem as an Integer Program. We denote by F the number of multicast flows and by R the number of racks. Multiple multicast flows can have the same sender and receiver racks. To simplify presentation, we refer to a flow as an aggregate of all the flows with the same set of senders and receivers. Each flow i has a size f_i and the number of optical splitters required s_i (if $s_i > 1$, splitters are cascaded). The binary variable a_i indicates if flow i is scheduled in the current computation. r_j is 1 if the rack j is available at the current iteration. The constant m_{ij} is 1 if the i^{th} flow requires rack j as a sender or a receiver. S denotes the number of splitters available and we assume that all modules have an identical number of ports (our model can be extended to support different number of ports). The problem can then be formulated as follows:

$$\max \sum_i a_i \cdot f_i \tag{6.1}$$

$$\sum_{i=1}^{F} a_i \cdot m_{ij} \leq r_j \quad \forall j = 1 \cdots R \tag{6.2}$$

$$\sum_{i=1}^{F} a_i \cdot s_i \leq S \tag{6.3}$$

At every iteration, the resource allocation algorithm selects the set flows from the traffic matrix that maximizes the objective (6.1). Each node has a single optical port and can serve only one flow at any instant, which is modeled by the constraint (6.2). Finally, the limit on the number of optical splitters is modeled by (6.3).

Optimal Solution: The Integer Program above, is a variant of the NP complete multidimensional knapsack problem [90]. The Integer Program can be optimally solved by

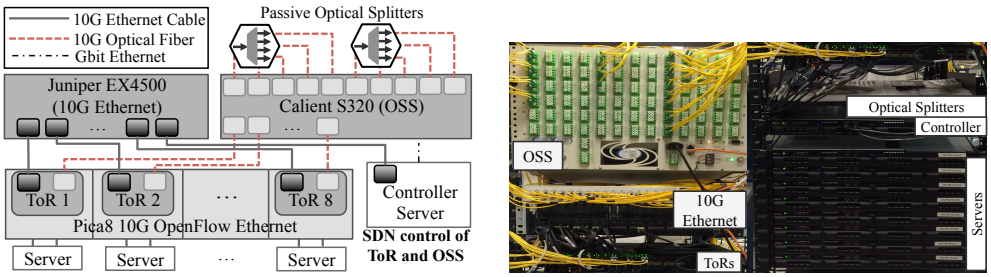
branch-and-bound methods. The optimal branch-and-bound methods can be potentially time consuming and lead to wasted optical resources (In Section 6.3.2, we show that optimal calculation for a large number of racks and flows can exceed the typical OSS reconfiguration time of 20-30 ms). Thus, we also employ a heuristic to efficiently solve the resource allocation problem.

Greedy Solution: The greedy algorithm iteratively selects the flows with the maximum values of traffic, $\sum_{j=1}^{H} f_i \cdot m_{ij}$ and checks if the flow can be scheduled (optical ports of all associated racks are free and required number of optical splitters are available). If yes, then the flow is scheduled and the racks and optical modules are marked as occupied. The greedy approach is faster but sub-optimal.

6.2.4 Prototype and testbed

Figure 6.4(a) shows the optical multicast system prototype configuration used in the experimental evaluations (see Section 6.4). It consists of 8 racks, each consisting of one server. Each server is equipped with a dual-port 10G Network Interface Controller (NIC), an Intel Xeon E5-2430 6-core processor and 24 GB of RAM. A Pica8 P-3920 10G OpenFlow switch is divided into 8 bridges that operate as 8 separate ToR switches. The EPS network among the ToRs is provided by a Juniper EX4500 switch. The Juniper EX4500 is a 40-port non-blocking 10G Ethernet switch that consumes 350 W and has a 2.7 μs latency. 10G Small Form-factor Pluggable (SFP+) Direct-Attached (DA) cables are used to connect ToRs to the Juniper switch.

The optical network is an OCS constructed by a Calient S320 OSS. The Calient S320 is a 320-port MEMS OSS with the connection setup time of 25 ms, 20 ns port-to-port latency, 45 W operation power, and a typical 2.0 dB insertion loss. 18 ports of the Calient switch are used to connect 8 ToRs and two 1:4 passive optical splitters. 1:4 optical splitters have 7.3 dB insertion loss and are connected to the Calient S320 by single-mode fibers. Optical to electronic conversions at ToRs are carried out by 10GBASE-ZR SFP+ transceivers and single-mode fibers provide optical links to the Calient S320. The controller server is also equipped with a dual-port 10G NIC, two Intel Xeon E5-2403 4-core processors and 24 GB



(a) (b)

Figure 6.4: Optical multicast system prototype: (a) Configuration, and (b) Picture. The prototype consists of an Ethernet switch, an OSS, 8 ToR emulated by an OpenFlow switch, 8 servers, two 1:4 optical splitters, and an SDN server that runs the control plane software.

of RAM.

We used Floodlight as the southbound API since the majority of commercial ToRs and OSSs are now OpenFlow enabled. For the OSSs that do not support OpenFlow, we developed a python-based API that controls the switch using TL1 commands. For the northbound API, we implemented a fast pub/sub messaging system using open-source libraries of Redis [185]. Byte size messages are transmitted through the EPS network from the network controller to the servers. The messaging system is much faster than the REST API, conventionally used as the northbound API.

6.2.5 Scalability

The scalability of the hardware architecture is determined by i) multicast group size, and ii) OSS port count. Every 1:2 optical multicast reduces the signal power by 3 dB. A 1:64 optical multicast requires 18 dB (20.5 dB in a manufactured device) link power budget that can be provided by SFP+ ZR transceivers. Cascading sixteen 1:32 passive optical splitters to the output ports of one 1:16 active optical splitter scales optical multicast group size to 512 racks using 545 optical ports. Active optical splitters provide lossless splitting using semiconductor optical amplifier (SOA) [77]. They can also provide tunability in the splitting ratio using Mach-Zehnder Interferometers (MZI) [141] to create asymmetric multicast trees.

MEMS OSS with 320 ports are commercially available [3] and 1100 port implementation was presented in [129]. Considering same number of splitter ports as number of racks, a 1100 port OSS can support 512 racks and maximum multicast group size of 512 (broadcast). Multiple OSSs (with an SDN controller to manage the traffic among them) can be placed in ring or tree topologies to support larger number of racks. Commodity OSSs are open-flow enabled and compatible to integrate with SDN data centers.

Software architecture scalability is imposed by the control plane delay. The control plane achieves an average end-to-end delay of 30–50 ms for processing 320 multicast jobs in a 320 rack data center (Section 6.3.1). The control plane delay grows slowly with increasing traffic matrix and rack sizes. This makes the system scalable to support hundreds of racks and numerous multicast flows.

6.3 Control plane evaluation

In this section, we evaluate the implementation of the prototype control plane. We measured the execution delay of different control plane components. The results indicate that the control plane incurs a very low overhead apart from the fixed OSS reconfiguration time. Moreover, we numerically study the optimal and greedy resource allocation algorithms, the reallocation strategies, and the impact of number of splitters. Our results show that: (i) the greedy heuristic is a practical and efficient solution to the resource allocation problem, (ii) a myopic reallocation strategy, can be more efficient than a far-sighted strategy, and (iii) deploying a large number of small splitters improves the performance of flows with small multicast group sizes.

6.3.1 Control plane

The total delay of the control plane consists of communicating via the northbound (Redis Messaging System) and southbound APIs, the resource allocation algorithm, and the network controller software. For Redis, we measured the average latency for transmitting 100 messages of 20 bytes between the controller and the servers. In our measurements, we did

Table 6.2: Average control plane delays measured on the prototype.

Control Plane Component	**Delay (ms)**
Northbound API (Redis)	0.65
Computing an optimal solution	22.40
Computing a greedy solution	1.90
Network Controller software	4.8
OSS reconfiguration	25
Total (Optimal):	**52.8**
Total (Greedy):	**32.3**

not include the ToR setup time (flow rule entry), since it is much faster (5–10 ms) than the OSS reconfiguration and is executed in parallel. Table 6.2 shows the average delay of different components for a configuration of 320 ToRs, processing traffic matrix of 320 multicast flows with ten 1:32 optical splitters on the prototype controller server. The total delay is 52.8 ms and 32.3 ms for the optimal and the greedy algorithms respectively.

6.3.2 Algorithm evaluation

We study the performance of the resource allocation algorithm through simulations. In our evaluations, the resource allocation algorithm runs on a traffic matrix of a given size, which is periodically repopulated with random flows. The flow sizes are uniformly distributed between 250 MB–2.5 GB and the multicast group is selected uniformly randomly among all the racks subject to a maximum group size. We modeled the OCS network with link speed and reconfiguration delays as measured on the prototype. The simulation time includes the OSS reconfiguration delay, the resource allocation algorithm computation delay, and the transmission time on the optical links. The results are averaged over several runs of 200 seconds simulation time. We define following metrics for our evaluations:

i) **Achievable Throughput**: Average throughput over the optical network excluding the throughput loss due to the idle time during resource allocation algorithm computation: $\frac{\text{Traffic Transmitted}}{\text{(Total Time - Total Algorithm Time)}}$. ii) **Effective Throughput**: Overall average throughput

over the optical network: $\frac{\text{Traffic Transmitted}}{\text{Total Time}}$.

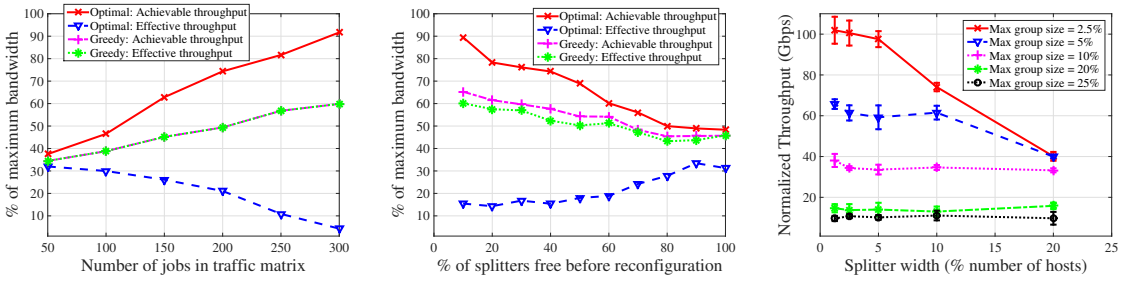
The achievable throughput is the theoretical maximum throughput that can be achieved by providing more computation power to the controller and using advanced optical switching technologies.

6.3.2.1 Optimal vs. greedy allocation

We computed the achievable and effective throughput for different traffic matrix sizes in a 320 rack network with twenty 1:16 splitters and maximum multicast group size of 32. Figure 6.5(a) shows the achievable and effective throughput as a percentage of the maximum bandwidth of the optical network. The achievable throughput for the both optimal and greedy solutions increases as the traffic matrix size increases. A larger traffic matrix increases the probability of scheduling larger flows, thus, increasing the achievable throughput. The optimal algorithm incurs large computation delay in processing large traffic matrix sizes and consequently, the effective throughput reduces with increasing matrix sizes. The difference between the achievable and effective throughput for the greedy algorithm is small due to fast algorithm computation. In summary, the greedy algorithm is an efficient heuristic in practice as it trade-offs the sub-optimal solution with faster performance. The optimal solution's computation time can be reduced by a more efficient implementation of the branch-and-bound method.

6.3.2.2 Reallocation strategy

We evaluate the reallocation strategies by measuring the achievable and effective throughput vs. the number of free splitters prior to reallocation. Figure 6.5(b) shows the results for an architecture of 320 racks and twenty 1:16 splitters. Myopic reallocation (waiting for a small number of splitters before reconfiguration) of free optical splitters, leads to a higher achievable throughput for both optimal and greedy solutions. Reconfiguring the switch as soon as a few splitters are available prevents wastage of optical resources (e.g. when a large flow and a small flow are scheduled together, a small flow will finish much faster). The far-sighted strategy (waiting for a large number of splitters before reconfiguration) leads to



(a) (b) (c)

Figure 6.5: (a) Achievable and effective throughput of the optimal and greedy algorithms vs. the traffic matrix size, (b) Impact of the reallocation strategy on the maximum achievable and effective throughput, and (c) Effect of optical splitter size on the maximum achievable throughput for 160-640 racks.

less frequent reconfigurations and consequently less overhead due to OSS reconfiguration and control algorithm delay. Since the optimal algorithm has a large control overhead, the effective throughput of the optimal algorithm is higher for the far-sighted strategy. However, far-sighted strategy results in a lower achievable throughput for the greedy algorithm. For the greedy algorithm, the control overhead is relatively low and the losses due to the idle time dominate.

6.3.2.3 Optical splitter sizing

Due to the limited number of ports on the OSS, only a fixed number of splitters can be used. Thus, it is important to determine the optimal number and size of the splitters. We evaluate the achievable throughput for the optimal algorithm for a traffic matrix of 100 multicast flows with maximum group size varying between 2.5%–25% of the number of racks. We consider the total number of racks ranging from 160 to 640 with an equal number of OSS ports in each scenario.

Figure 6.5(c) shows the achievable throughput vs. the optical splitter widths (as a percentage of the number of racks) for different maximum multicast group sizes. The average throughput values are normalized to the same scale after accounting for increased capacity

due to number of ports in each scenario. We observe that for smaller multicast group sizes, the achievable throughput is higher with narrower but larger number of splitters. The achievable throughput decreases as the multicast group size increases as several modules need to be cascaded to serve one single flow. As a rule of thumb, we will use splitter sizes between 5–10% of the total number of racks. This range of splitter size limits excessive cascading for larger multicast group sizes and provides higher throughput.

6.4 System evaluation

In this section, we evaluate the performance of Optical Multicast System for transmitting bulk multicast flows and compare it with IP multicast, unicast, and peer-to-peer methods. In all evaluations, optical multicast refers to physical layer optical multicast using passive optical splitters. We start by presenting experimental results measured on the prototype testbed. Next, we present numerical evaluations at scale, computed on our custom simulation platform and conclude with the cost and energy efficiency analysis. We use following metrics:

i) **Transmission Time:** Time to deliver a flow excluding the connection overheads.

ii) **Latency:** Total time to deliver a set of flows including all connection and control plane overheads.

iii) **Throughput:** Link throughput while transmitting one flow: $\frac{\text{Flow Size}}{\text{Transmission Time}}$.

iv) **Connection Overheads:** The total delay from a request to begin the flow transmission.

v) **Energy Consumption:** Energy (Joules) = Power (Watt)×Transmission Time (Seconds).

We compare the performance for delivering a traffic matrix of multicast flows with i) IP multicast that creates a multicast tree (star for the 1-hop topology in our prototype) using spanning-tree algorithm, manages the multicast group memberships, and replicates packets at the switch/router, ii) sequence of unicast transmissions on the EPS network, iii) peer-to-peer method that imitates multicast transmission by creating many-to-many connections using bittorrent [75], and iv) an optical circuit switching network not equipped with the

optical multicast system.

We perform comparison with both non-blocking and over-subscribed EPS networks as real-world data center networks are typically over-subscribed. Moreover, since the extra optical switching capacity in the optical multicast system allows further over-subscription of the EPS network, this comparison helps us to evaluate the benefits of adding this system to a data center network. Following are the network configurations in our evaluations:

- Physical layer optical multicast
- Transport layer IP multicast
  - Non-blocking EPS network
  - EPS network + background traffic
- Multicast through sequence of unicasts
  - OCS point-to-point network
  - Non-blocking EPS network
  - EPS network + background traffic
- Peer-to-peer multicast using Twitter Murder
  - Non-blocking EPS network
  - EPS network + background traffic

Implementation Details: We used Iperf [8] to generate data and measure the link characteristics. Iperf is a common network performance measurement tool that generates UDP and TCP datagrams and measures the network throughput. UDP is an unreliable but fast and efficient transmission protocol For a fair comparison against peer-to-peer multicast that guarantees data delivery, we optimized the UDP buffer size and service type to achieve an average 0.35% packet error rate for 4.2 Gbps throughput. The multicast transmission can be improved by using reliable multicast protocols [9,11] in which, data is transmitted on the

optical network and the NACKs on the electronic network [182]. Flows are read/written in the host memory on transmission/reception to make maximal use of the optical link bandwidth. The Juniper EX4500 switch in the testbed provides advanced Layer 3 functionality that allows IP multicast implementation. For peer-to-peer multicast, we implemented Murder [19] that is a bittorrent-based fast data distribution platform developed by Twitter. We implemented Murder using the open source Herd libraries [7]. To emulate over-subscription, we generated background traffic using Distributed Internet Traffic Generator (D-ITG) [5] on a spare NIC of the servers.

6.4.1 Experimental results

In the first experiment, we transmit a traffic matrix of 50 multicast flows, with uniform distribution of flow size (250 MB–2.5 GB) and multicast group size (1–4). These flow sizes are chosen based on the data center applications. As discussed in the introduction, parallel database join operations include multicast flows of several hundred megabytes. For software updates (e.g. OS updates) and VM migrations, the flow sizes are in the range of gigabytes. Figure 6.6(a) shows the latencies over different network configurations. Optical multicast and IP multicast have comparable latencies. IP multicast with background traffic leads to 32% higher latency than optical multicast. Transmitting multicast flows with sequence of unicast flows is twice as slower than optical multicast. In this case, adding background traffic increases the latency by over 3x. Peer-to-peer multicast on the EPS network is an order of magnitude slower than optical multicast with and without background traffic. Transmitting the same traffic matrix in a hybrid network not equipped with the optical multicast system takes twice as long. We observe that the OSS reconfiguration time does not noticeably affect optical multicast latency for bulk multicasting. Equipping an OCS network with optical multicast has significant impact on the multicast traffic delivery. Furthermore, peer-to-peer is a time-consuming multicast data delivery method.

In the next experiment, we evaluate the effect of multicast group size on transmitting one multicast flow. We measured the transmission time excluding the connection overheads. Figure 6.6(b) shows the results for a 250 MB flow. Optical multicast, IP multicast and peer-

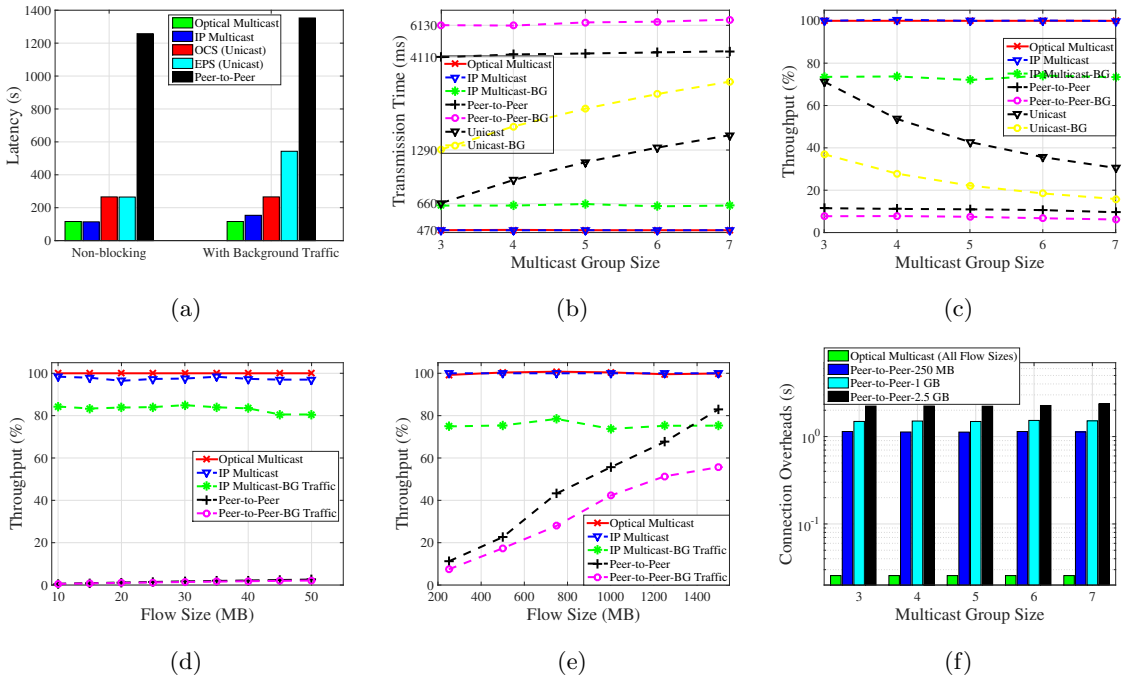


Figure 6.6: Experimental results: (a) Latency to deliver 50 multicast flows in a configuration of 8 racks and two 1:4 optical splitters. (b, c) Evaluating the effect of increasing number of multicast receivers on the transmission time and the throughput of a 250 MB flow, (d, e) Evaluating the effect of flow size on throughput for mice and elephant flows, (f) Evaluating the effect of flow and multicast group size on peer-to-peer multicast connection overheads, (BG: Background Traffic).

to-peer have constant transmission time regardless of multicast group size. Unicast method transmission time increases linearly with the number of receivers.

We also measured the throughput vs. multicast group size as shown in Fig. 6.6(c). Optical multicast and IP multicast achieve the highest throughput regardless of multicast group size and we compare all subsequent results with this value. Introducing background traffic decreases the throughput of IP multicast to 73% of its original throughput. Unicast method's performance decreases with increasing group size. For multicast group of 7 receivers, unicast method's throughput is almost 30% of optical multicast. Background traffic reduces the throughput of unicast further close to 16% of the optical multicast. Peer-to-peer multicast has a very low throughput for a 250 MB flow size close to 10% of optical multicast.

This result confirms that multicast group size does not change the performance of optical and IP multicast. However, unicast transmission performance is highly dependent on the number of receivers, as expected. Peer-to-peer method also has constant throughput over different multicast group sizes but it is an order of magnitude lower than optical multicast.

We also measured the impact of the flow size on the throughput for optical multicast, IP multicast, and peer-to-peer methods. For each measurement point, we performed 3 multicast transmissions with 2–4 multicast receivers and average the throughput. However, based on the previous experiment, the multicast group size has no impact on the performance of these methods. We plotted all the measurements compared to the highest throughput that was for optical and IP multicast.

Figures 6.6(d) and 6.6(e) show the results for mice and elephant flows, respectively. Optical and IP multicast achieve similar performance irrespective of the flow size. Introducing background traffic has lower impact on the throughput of mice than elephant flows. Peer-to-peer multicast has an average throughput of 1–2% for mice flows. Its performance improves as the flow size increases and it reaches similar performance as optical multicast for flows larger than 1.5 GB. We conclude that peer-to-peer multicast is not efficient for transmission of mice multicast flows. However, for very large flows, it achieves comparable performance as optical and IP multicast. For peer-to-peer multicast, the impact of background traffic is more notable on transmission of elephant flows.

We further compared the efficiency of optical multicast and peer-to-peer multicast by measuring the connection overheads on transmitting 250 MB, 1 GB and 2.5 GB flows. Optical multicast connection overhead is the OSS configuration time. For peer-to-peer multicast, it is the bittorrent file (metadata of the flow) generation and peer-to-seed connection times. Figure 6.6(f) shows the results for 3–7 multicast receivers. Peer-to-peer multicast has significantly higher and variable connection overhead that increases with the flow size.

We also measured the share of connection overheads in the latencies of the first experiment presented in Fig. 6.6(a). For optical multicast, it was 5% of the overall latency but 47% for the peer-to-peer multicast. We infer that peer-to-peer multicast has longer and

varying connection overheads that increases mainly by the flows size and slightly with the multicast group size. However, the connection overhead for optical multicast is the fixed OSS reconfiguration time.

To summarize, optical multicast achieves similar performance as IP multicast regardless of the flow size. The performance of optical multicast is not degraded by increasing the multicast group size. Multicast transmission through sequence of unicast flows has link stress proportional to the number of receivers and results in higher latencies. Peer-to-peer multicast is not suitable for transmission of mice flows. However, it can be a reasonable solution for low-priority bulk multicasting. Furthermore, adding the optical multicast system to a hybrid network will significantly improve the performance of multicast flow transmission.

6.4.2 Numerical results

In order to evaluate the performance of the optical multicast system at scale, we developed a custom packet-based simulation environment using NS3 libraries [13]. For optical multicast, we used testbed measurements to adjust the channel end-to-end delay. We also added the OSS reconfiguration time to the connection overheads and used on-off communication patterns to generate data. To get statistically meaningful results, we repeated each experiment 10 times and averaged the results. For our evaluations, we used following network configurations:

- Physical layer optical multicast
- Transport layer IP multicast
  - Non-blocking EPS network
  - 1:4 Over-subscribed EPS network
  - 1:10 Over-subscribed EPS network
- Multicasting over sequence of unicasts
  - Optical point-to-point OCS network

- Non-blocking EPS network
- 1:4 Over-subscribed EPS network
- 1:10 Over-subscribed EPS network

In the first evaluation, we consider a network of 320 racks and ten 1:32 optical splitters. A traffic matrix of 320 multicast flows with maximum multicast group size of 32 and flow size of 250 MB–2.5 GB (both uniformly distributed) is transmitted on the networks. We used the greedy resource allocation algorithm.

Figures 6.7(a) and 6.7(b) show the latencies for different network configurations. Optical multicast and non-blocking IP multicast achieve comparable latencies. For IP multicast, a 1:4 and 1:10 over-subscribed EPS network leads to 3x and 9x higher latencies, respectively. Multicast through sequence of unicast transmissions over a non-blocking EPS network results in 50% higher latency as compared to optical multicast. In this case, 1:4 and 1:10 over-subscription decrease the latencies by 5x and 13x, respectively. Confirming the experimental results, OSS reconfiguration time does not result in notable additional latency for bulk multicasting. Multicast through unicast transmissions is not efficient for transmitting multicast flows, especially in over-subscribed networks.

In the next set of numerical evaluations, we study the effect of multicast group size. Figure 6.7(c) shows the transmission time of a 250 MB flow to 5–100 multicast receivers. We observe that increasing the multicast group size does not affect the performance of optical and IP multicast. However, the performance of unicast transmission degrades as the multicast group size grows. Figure 6.7(d) shows the throughput with increasing multicast group size. Optical and IP multicast achieve close to line-rate (10 Gbps) throughput. Over-subscription by ratios of 4 and 10, decreases the throughput of IP multicast to 2.36 and 0.94 Gbps, respectively. Finally, the throughput of unicast transmission is inversely proportional to the number of receivers.

In order to better understand the impact of the optical multicast system on a data center network, we computed the latency for delivering 20 TB of data based on the overall switching capacity (including switching delays) for the following network configurations: i)

An EPS network in non-blocking, 1:4 and 1:10 over-subscription configurations, ii) A hybrid architecture consisting of an OCS network and an EPS network in non-blocking, 1:4 and 1:10 configurations, and iii) Optical multicast system with 320 splitter ports on a hybrid architecture with the EPS network in non-blocking, 1:4 and 1:10 over-subscription configurations. In the sole EPS configuration, the EPS layer serves both unicast and multicast (through sequence of unicast transmissions) flows. In the case of hybrid configurations, the optical layer first serves all multicast flows and then transmits unicast flows along with the EPS layer. We varied the volume of multicast flows 1–55% with average multicast group size of 16. As shown in Fig. 6.7(e), adding an extra OCS network results in 48% lower latency than a sole EPS network. With 15% of the total multicast flows, the optical multicast system reduces the latency by 55%. Adding the optical multicast system to an over-subscribed EPS network, the latency is reduced by 83% and 92% for 1:4 and 1:10 over-subscription ratios, respectively. The gains of adding the optical multicast system improve with increasing percentage of multicast flows and larger average multicast group sizes.

6.4.3 Cost and energy efficiency

Optical circuit switching consumes considerably less power than electronic packet switching. Table 6.3 shows the typical per port power values for commercially available EPS, OCS and the optical multicast system network components. In the optical multicast system prototype, the per port power consumption of optical switching is 60x lower than EPS. Furthermore, using OCS in data centers avoids unnecessary optical-electrical-optical conversions at the electronic packet switches. We computed the total switching energy to achieve similar latency values for different network configurations. It is calculated based on the per port energy consumption and the transmission time ($E_{(J)} = P_{(W)} \ \ t_{(s)}$), thus to achieve similar latency with electronic unicast as optical multicast, more EPS ports (more bandwidth) are required, i.e. more energy consumption. Figure 6.7(f) shows the switching energy as the multicast group size increases. To deliver a 250 MB multicast flow, optical multicast consumes an order of magnitude less switching energy than IP multicast. The difference grows to 2 orders of magnitude for IP multicast in a 1:4 over-subscribed network

Table 6.3: Power consumption and cost of the EPS, OCS and the Optical Multicast System network components.

Component	**Per Port Energy (W)**	**Per Port Cost** ($)
10GBASE-SR SFP+	1	260
10GBASE-ZR SFP+	1.5	605
10G EPS Switching	8.75	575
Optical Switching	0.14	350
Optical Splitter	0	2

Table 6.4: Cost increase in adding an OCS network + Optical Multicast System to a 320 rack data center EPS network under different over-subscription conditions.

Network Configuration	**Interconnection Network Cost Increase**
Non-blocking EPS + OCS + Optical Multicast	156%
1:4 OS EPS + OCS + Optical Multicast	104%
1:10 OS EPS + OCS + Optical Multicast	87%

as well as the unicast method in a non-blocking configuration.

For a more comprehensive energy efficiency analysis, we computed the total energy consumption (sum of the transceiver and the switching energy consumptions) for delivering 20 TB of data in a 320 rack data center with an average multicast group size of 16. In Fig. 6.8(a), we compare a solely EPS network with a hybrid network equipped with optical multicast. With multicast flows constituting 15% of the total volume of the flows, a hybrid network equipped with optical multicast consumes up to 50, 80, and 91% less switching energy than a solely EPS network in non-blocking, 1:4, and 1:10 over-subscribed configurations, respectively.

Enabling optical multicast in a hybrid network requires optical transceivers with higher output power and extra optical switching ports to attach the splitters. The per port costs and energy values for a typical transceiver used in hybrid networks (SFP+ SR) and the ones required for the optical multicast architecture (SFP+ ZR) are presented in Table 6.3. We

calculated the cost of building a hybrid network + optical multicast vs. a non-blocking EPS network. Adding an extra optical network increases the overall switching capacity of the data center. This allows for supporting more servers or increasing the over-subscription ratio of the EPS network. Table 6.4 shows the additional cost of building the optical multicast system, compared to a solely non-blocking EPS network in 3 configurations: i) non-blocking, ii) 1:4 over-subscribed, and iii) 1:10 over-subscribed. We ignored the links cost in our analysis as it is negligible. We assumed that the cost of an EPS network linearly reduces with increasing the over-subscription ratio. Building a hybrid + optical multicast network with a 1:4 over-subscribed EPS network will cost twice as much compared to a solely EPS non-blocking network. However, even with 10% of the total flows being multicast, it will provide approximately 80% lower latency (Fig. 6.7(e)) and energy consumption (Fig. 6.8(a)). Furthermore, considering that network is only 15% of the data center cost [95], the investment improves data center performance and reduces the operating costs. For large-scale data centers with more than 320 racks, the per port switch cost and switching energy consumption scales linearly.

6.5 Paxos with optical multicast

As discussed in Section 6.2, distributed files systems are widely used as a data storage solution in data center networks. Majority of these systems use Paxos algorithm or its variation to provide strong consistency guarantees. Paxos-type algorithms will significantly benefit from multicast-enabled networks. Ring Paxos [149] is a variation of Paxos that uses IP multicast to disseminate messages among the learners. We chose Ring Paxos since compared to other atomic broadcast protocols [43,137], it achieves higher throughput, lower latency, and steady performance as the number of receivers increases. We implemented Ring Paxos on the servers of our prototype testbed and evaluated its performance over the optical multicast and an IP multicast-enabled EPS network. The configuration is 5 Acceptors, 1–7 Learners, and 8, 16 and 32 kbytes message sizes. Figure 6.8(b) shows the receiving throughput of the Learners that confirms successful end-to-end implementation of

Paxos on the optical multicast system.

6.6 Optical incast

Optical multicast architecture is designed to enable direct integration of optical modules and subsystems such as Arrayed Waveguide Gratings (AWG) [200] and Wavelength Selective Switches (WSS) [49] to provide functionalities such as incast, all-to-all-cast, or aggregation/breakout of links with different data rates. These functionalities can also address inter data center network applications such as multicast and incast between data centers or providing rack-to-rack connectivity across data centers to improve scalability and reliability [180].

Optical splitters are bi-directional and work as combiners as well. This functionality allows enabling rack-to-rack optical incast. As demonstrated in Fig. 6.8(c), incast flows can be routed by i) building an incast tree between all the senders (S_1,...,S_n) and the receiver (R_1) using the optical combiner and, ii) time-division multiplexing the senders using an SDN controller to utilize the full link capacity. Compared to optical multicast, optical incast does not require high power transmitters since the optical signal of the senders are added rather than divided. However, achieving efficient time-division multiplexing of senders requires a fast northbound API to minimize the controller overhead.

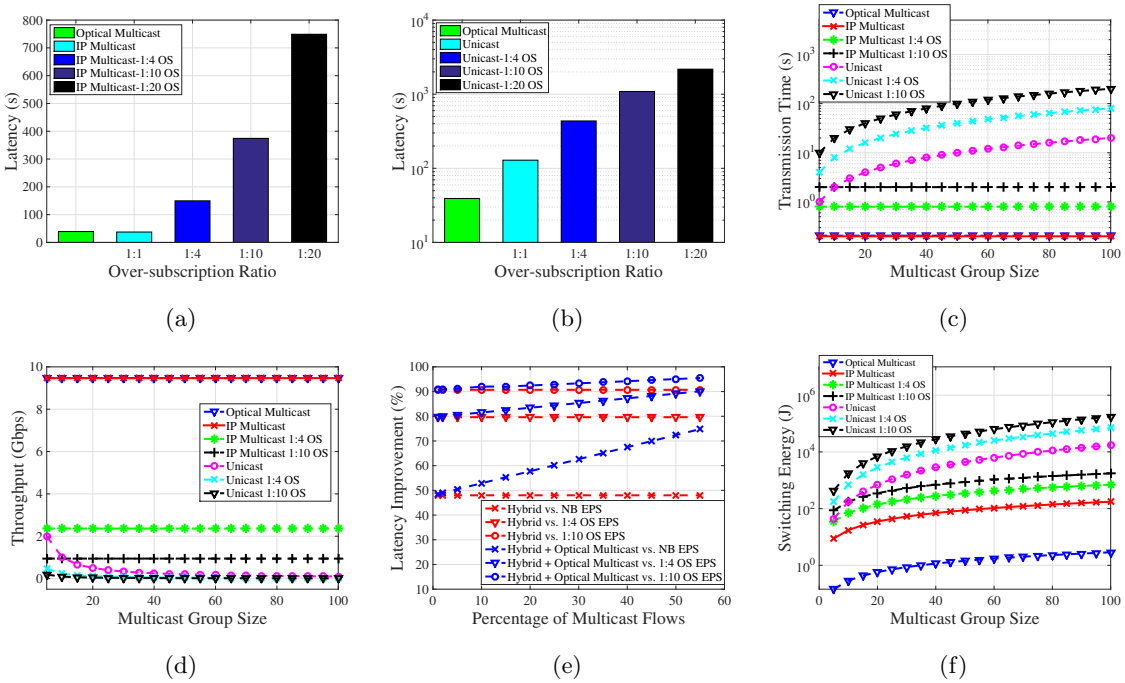
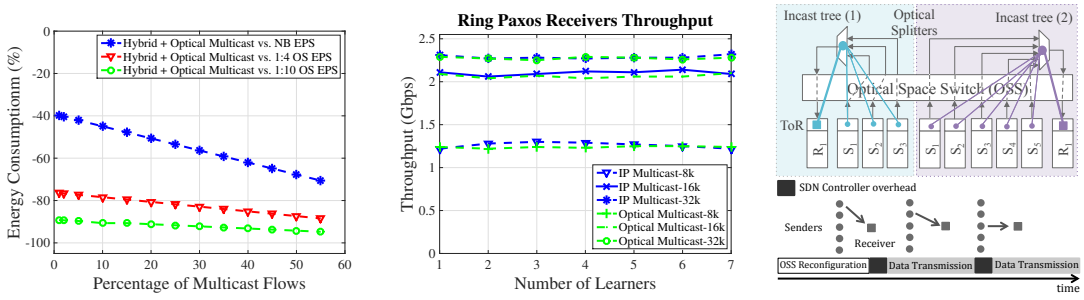


Figure 6.7: (a, b) Numerical results on the latency of delivering 320 multicast flows among 320 racks with ten 1:32 optical splitters, comparing optical with IP multicast and unicast on an EPS network in non-blocking and over-subscription configurations, (c, d) Effect of multicast group size on transmission time and throughput for a 250 MB flow, (e) Latency improvement of a hybrid and optical multicast-equipped data center network in delivering 20 TB of data compared to a sole EPS network vs. percentage of multicast flows, (f) Calculation of switching energy on delivering a 250 MB multicast flows to achieve similar latency vs. Multicast Group Size, (OS: Over-subscribed, NB: Non-blocking).



(a) (b) (c)

Figure 6.8: (a) Improvement in energy consumption on delivering 20 TB of data with Hybrid + optical multicast network compared to a sole EPS network in non-blocking, 1:4 and 1:10 configurations (OS: Over-subscribed, NB: Non-blocking), (b) Ring Paxos run on IP multicast supported EPS network and optical multicast-enabled network (message size: 8, 16 and 32 kbytes), (c) Enabling optical incast using passive optical combiners and Time-Division Multiplexing of the senders by the SDN controller.

Chapter 7

QCN BASED DATACENTER CONGESTION CONTROL

7.1 Introduction

Data centers host a large variety of applications including web search, social networks, recommendation systems, and database storage. Intensive concurrent communications can go among tens to hundreds or even thousands of servers. Heavily fan-in traffic patterns and microbursts are often seen with distributed applications. All these new trends are demanding stringent performances of throughput, latency and scalability from datacenter networks.

In addition, inter-datacenter communications often go through WAN peering links, which are generally high-end, premium links that network operators demand their high availability. Since packet losses are generally unacceptable over WAN links due to large RTTs, the edge links usually include a large buffer. Furthermore, datacenter operators pay peering link operators for their peak utilization (95 percentile) and consequently, ensuring high average throughput is cost-effective.

Fast and accurate congestion notification plays a crucial role in meeting these stringent demands. Conventionally, the network signals congestion by dropping packets, which could

be detrimental to applications' performance. Hence, modern datacenters adopt Explicit Congestion Notification (ECN) to mark packets in order to notify impending congestion. However, only receivers can echo ECN marks back to the senders, which leads to end-to-end latency. Besides, single bit ECN feedback is not sufficient to indicate the degree of congestion, and hence averaging ECN marks is a common practice that further increases the networks' response time.

Separately, a Layer 2 congestion management algorithm, QCN (Quantized Congestion Notification), has been standardized for Ethernet Data Center Bridging: IEEE 802.1Qau. QCN uses direct multi-bit network feedback as congestion notification and it has the following properties that make it a more reliable congestion metric: (i) direct feedback from congestion points, (ii) multi-bit feedback, and (iii) inclusion of higher-order queue growth. Besides, more information on where congestion occurred is useful for load balancing and maybe other telemetry purposes. Unfortunately, IP-routed datacenters limit QCN's adoption.

While most datacenter switches support QCN feedback, QCN's reaction functions require changes at servers' Network Interface Cards (NICs) where QCN's support is less common. We aim to leverage commonly available high-fidelity QCN's congestion signal but make it Layer 3 capable and show how it can be used in two different congestion management areas: congestion control (which acts in sub-RTT time scale) and load balancing (which acts in a multiple RTT time scale).

First, we propose using standard QCN feedback to tune TCP's congestion window instead of adjusting the rate of hardware rate-limiters: QCN-CC. There are several challenges that we need to address such as devising a minimum set of kernel level changes at end hosts to process QCN feedback, and enabling QCN messages to work at L3 of networking stack. With this we avoid the dependency of QCN on NICs that support it, hence making it immediately usable with any host NIC.

In addition, we propose to use QCN feedback for accurate congestion-aware load balancing: QCN-LB. While FlowBender [121] bases its decision on single-bit ECN marks for rerouting traffic. QCN-LB calculates aggregated QCN feedback quanta; and if it is over a

threshold, QCN-LB tries a different path.

Our main contributions are described below:

(i) **Implementational**: We show how to make QCN feedbacks across the boundary between Layer 2 and Layer 3 domains. Our proposed changes can be readily incorporated in most current commercial switches. We also overcome several issues with original QCN congestion control mechanism such as unfairness, overreaction to congestion notifications etc.

(ii) **Design**: 1) Congestion Control (QCN-CC): we extend QCN's host-side functions to window-based transport layer, instead of using hardware rate limiters. Furthermore, *QCN-CC* is self-clocked and it precludes the need of high-performance hardware or software based timers. *QCN-CC* can provide flow-level isolation thus, avoiding arbitrarily sharing of the rate limiters by flows which might severely interfere with each other. 2) Load balancing: we also make use of QCN's feedback to guide load balancing. Aggregated feedback quota is used to indicate whether a path is under congestion and call for a reroute.

(iii) **Evaluation**: We compare the performance of *QCN-CC* and *QCN-LB* against other state-of-the-art congestion control and load balancing mechanisms using simulations based on realistic datacenter workloads. A key aspect of these evaluations is including both intra and inter-datacenter traffic which poses significant challenges for congestion management due to mismatch of flow RTTs by orders of magnitudes. Our simulations, based on real world workloads, show the benefit of direct QCN feedback: QCN-CC significantly reduces the tail latency of short flows as compared to competing ECN-based congestion control schemes DCTCP (by 3.5×) and DCQCN (by 2.0×). For traffic load balancing which works at a slower time scale than congestion control, QCN-LB is able to reduce flow's FCT by as much as 2.0× compared to ECN-based FlowBender.

The work presented in this chapter was done in collaboration with Abdul Kabbani,

Milad Sharif, and Rong Pan. Abdul Kabbani made major contributions to the design level details and algorithmic ideas. Milad Sharif, Rong Pan, and Abdul Kabbani provided important feedback and ideas behind the evaluation presented in this chapter.

7.2 Related Work

Providing high-fidelity feedback, congestion control, and load balancing for datacenter networks has received considerable attention. Below we briefly highlight the most relevant work.

Congestion control: Data Center TCP (DCTCP) [39], HULL [41], D2TCP [204] rely on Explicit Congestion Notification (ECN) marks aggregated over several packets due to obtain fine-grained congestion information. TCP Bolt [197] also relies on ECN and is essentially DCTCP without slow start. However, averaging ECN marks to derive congestion information over several RTTs could be slow and lead to delayed reaction to congestion. Further, ECN marking is generally not supported end-to-end outside the datacenter fabric and in fact, the ECN marks could be overwritten at external switches. This makes ECN based techniques unsuitable for dealing with congestion at the datacenter edge.

TIMELY [156] and DCQCN [232] are two recent proposals for congestion control for RDMA deployments. While TIMELY relies on accurate RTT measurements, DCQCN relies on ECN marks as a proxy for QCN feedback. However, such end-to-end congestion methods may require several RTTs to converge and may lead to packet losses especially for small flows. Further, TIMELY cannot be supported without specialized hardware to guarantee high fidelity RTT information.

ICMP Source Quench [94] relies on notifications from switches for managing congestion. However, due to challenges associated with practical deployments, its deployment never took off and has been recently deprecated. FastLane [229] leverages in-network notifications to avoid packet losses for small flows but it is not a congestion control scheme and it requires changes to switches. Several proposals have been proposed for reducing latency of TCP flows in datacenter and other environments. Some examples include pFabric [42],

Detail [228], Fastpass [165], RCP [86]. These proposals require significant changes to switch architectures, and to our knowledge, are not being deployed on a large scale.

Congestion control for datacenter-edge links has not been widely addressed. Most of the existing schemes rely on minimizing congestion at inter-datacenter or WAN links through traffic engineering, e.g., SWAN [111] and B4 [116]. TCP Fast Open [167] proposes techniques to reduce completion time of short flows over long RTTs by enabling data exchange during the handshake phase.

QCN: Several papers have looked at the performance of different aspects of QCN. [37] discusses the design and evaluation of QCN congestion control mechanism. [83] evaluated the performance of QCN in incast scenarios while [230] proposed an algorithm FQCN to improve fairness of multiple flows sharing one bottleneck link in incast scenarios. AF-QCN [120] addressed the issue of providing weighted-fairness for QCN congestion control in multi-tenant datacenters.

Load Balancing: Equal Cost Multiple Path (ECMP) forwarding is the standard mechanism used in today's datacenters for load balancing. However, ECMP results in static path assignments which may lead to large flows sharing congested paths. MPTCP [171] splits a TCP flow into multiple subflows where each subflow is randomly assigned a different path and flows with good performance are prioritized. However, MPTCP incurs significant implementation complexity on both senders and receivers.

FlowBender [121] leverages ECN marks for rerouting flows from congested paths. CONGA [38] employs congestion-aware flowlet switching on specialized hardware for load balancing. Centralized load balancing techniques such as Hedera [35], B4 [116], SWAN [111], and MicroTE [57] are complex and may not handle traffic volatility inherent in datacenter workloads.

7.3 Background

The Ethernet local area network standardization body, IEEE 802.1, started defining a set of enhancements to their protocols for data center environments as of 2005. The goal back

then was to bring together Ethernet, fiber channels and InfiniBand, for use with clustering and storage area networks. IEEE 802.1Qau, based on the QCN algorithm, was designed to provide congestion management for data center bridging traffic. In this section, we will review the design of QCN, analyze its pros and cons, and explain our motivation for overcoming QCN's limitation by designing *QCN-CC*.

7.3.1 QCN Algorithm

We start with a brief overview of the QCN algorithm, highlighting the more relevant aspects to our design. We refer to [22] and [37] for more details. QCN algorithm has two components:

(i) *Congestion Point (CP)*: This is the component in the switch that samples the incoming packets, measures the extent of the congestion, and conveys that information back to the source of the packet using a multi-bit feedback.

(ii) *Reaction Point (RP)*: This is a component at the sender side with a Rate Limiter (RL), which decreases the rate based on the feedback it receives from CP and actively probes for available bandwidth in the absence of congestion.

Congestion Point

The goal of CP is to maintain the queue at the desired buffer occupancy Q_{eq}. The CP randomly samples the incoming packets with a probability depending on the severity of the congestion [1]. It then computes the congestion score F_b as:

$$F_b = (Q - Q_{eq}) + w \cdot (Q - Q_{old}), \tag{7.1}$$

where Q is the instantaneous queue-size, Q_{old} is the queue-size when the last packet was sampled, and w is a non-negative constant which is typically set to 2. If $F_b > 0$, CP sends a feedback message containing the quantized value of F_b to the source of the sampled packet.

[1] The sampling period can be configured in a lookup table, against the computed congestion score, typically ranging between 150KB and 15KB.

The intuition is that F_b captures the queue-size excess $(Q - Q_{eq})$ as well as the derivative of the queue size $(Q - Q_{old})$. Hence, positive F_b means that either the buffer or the link are oversubscribed.

Reaction Point

The RP algorithm maintains two rates: (a) *current rate (R_C)*, which is the sending rate at any point in time, and (b) *target rate (R_T)*, which is current rate just before receiving the last congestion feedback. RP goes throught three main phases:

(i) *Rate decrease:* When RP receives a congestion notification, it cuts CR in proportion to the received F_b.

$$R_T \leftarrow R_C \tag{7.2}$$

$$R_C \leftarrow R_C(1 - G_d \cdot F_b) \tag{7.3}$$

where G_d is set such that the sending rate can be decreased by at most 50%.

(ii) *Fast Recovery (FR):* After cutting its rate, RP enters the FR phase to gradually recover the lost bandwidth and get back to the target rate. The recovery cycles are based on timer expirations or byte counter resets. At the end of each cycle, R_C is updated as follows:

$$R_C \leftarrow \frac{1}{2}(R_C + R_T) \tag{7.4}$$

(iii) *Active Increase (AI):* RP enters active increase phase after completing five cycles of FR to probe for extra bandwidth. During this phase, RP increases its sending rate by a configurable R_{AI} value in each stage (also based on byte-counting and a timer).

7.3.2 Pros and Cons of QCN

In this section, we discuss the advantages and limitations of QCN. On the positive side, QCN benefits from (i) a direct and granular multi-bit network congestion feedback, which is

fast and accurate; (ii) a Proportional Integral (PI) based controller to enhance end-to-end loop stability; (iii) rate recovery mechanism similar to BIC-TCP [224] at host, that has significant impact on stability of the control loop especially in the presence of increasing feedback delay [40]; (iv) a timer-based fast recovery mechanism to recoup from low rates; (v) rate-based mechanism, as opposed to window-based, which helps to alleviate bursts in the network, leading to more stable queues and reduced queueing latency. We discuss the impact of some of these features in more details in Section 7.4.1.

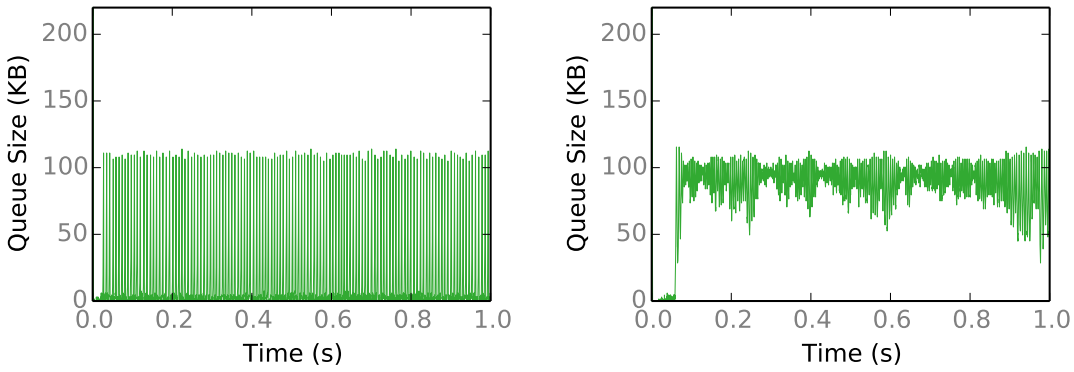
On the negative side, QCN's major limitations are: i) targeting Layer 2 domain only by design, while most data centers today operate in Layer 3 routed domain; ii) requiring hardware support both at the hosts as well as in the fabric switches (very few host NICs implement QCN's rate limiters even though there are widely spread QCN switches); (iii) not being scalable due to the relatively low number of rate limiters at NICs; and (iv) having many configuration parameters to tune.

7.4 Design

Several congestion control schemes have been proposed in the past decade, which have raised the bar of performance and implementability. In this section, we are interested in answering the following questions: what are the major features that fundamentally distinguish QCN as a congestion control scheme? How much of QCN's current design and implementation is it really worth to salvage or overhaul? And what are the minimal practical adjustments needed for immediately leveraging QCN on readily-deployable and commercially-available platforms?

7.4.1 Dissecting QCN

Before diving deep into the final design pf *QCN-CC* , it is important to understand the impact that each major component of QCN has on the performance of QCN as a congestion control scheme. We illustrate the value of the QCN feedback signal using a series of micro-benchmarks. In particular, we will discuss the advantages of: (i) sending direct feedback to



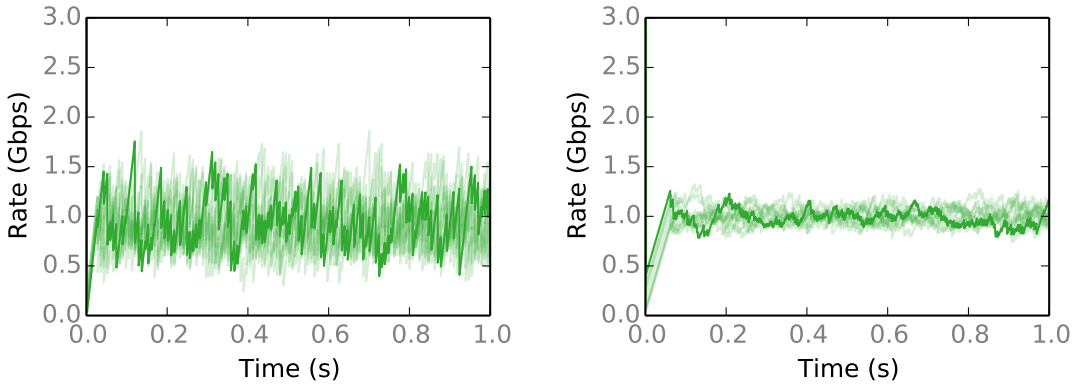
(a) $w = 0$ (b) $w = 2$

Figure 7.1: Impact of derivative term on queue size for QCN rate control with 6-bit congestion feedback.

the sender, (ii) differential term of the feedback, and (iii) multi-bit feedback (up to 6 bits) as opposed to a single bit feedback such as ECN marking.

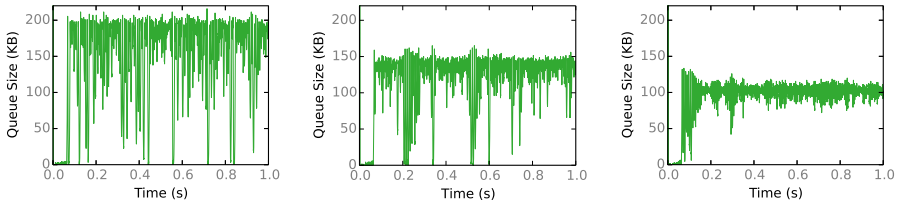
To evaluate the QCN feedback mechanism, we simulate a 10:1 incast scenario in a dumbbell topology using a packet-level simulator. We use following parameters for the QCN-based rate control algorithm described above: byte counter of 150KB, timer of 15ms, and linear increase of $R_{AI} = 5$Mbps. These parameters are close to the optimal values as determined by previous studies [37]. We also add a 5% jitter in the timers to avoid the buffer overflow caused by the synchronized senders. The buffer size at the congested switch is 1MB and the QCN feedback threshold Q_{off} is set to 90KB. For the sake of simplicity, we disable Hyper Active Increase (HAI) in the rate control mechanism since the main utility of HAI is to recover from severe congestion events.

In full QCN implementation, dynamic sampling probability varies between 1-10% based on the extent of the congestion at any switch. However, in our evaluations, we neglect the dynamic sampling rate and use a fixed sampling rate of 10%. The switches can be easily tuned to set such a fixed sampling rate. In current commodity switches, the sampling rate can go as high as 50%. Since the QCN notifications are small, this adds relatively low overhead.



(a) (b)

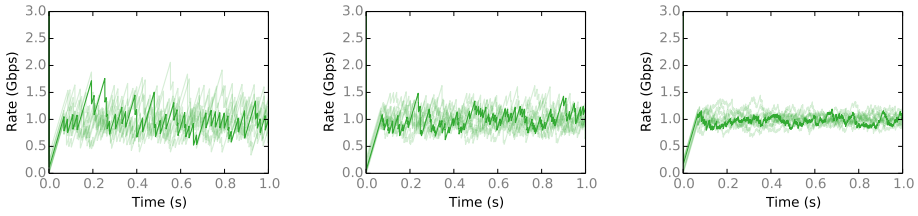
Figure 7.2: Impact of derivative term on flow rates in QCN rate control with 6-bit congestion feedback.



(a) 1-bit (b) 2-bit (c) 4-bit

Figure 7.3: Queue sizes for QCN rate control vs. the number of bits in QCN feedback. Results for 6-bit feedback are shown in Figure 7.1(b).

Impact of derivative: We first demonstrate the impact of the derivate term in the QCN congestion feedback. We run the incast scenaio for two QCN-based rate control with $W = 0$ and $W = 2$. Figure 7.1 shows the queue at the congestion point and Figure 7.2 shows the individual flow throughputs for $W = 0$ and $W = 2$. It is evident that the derivative term leads to higher queue stability closer to the desired queue occupancy. Furthermore, as shown in Figure 7.2, individual flow throughputs significantly benefit from a more stable queue. With $W = 0$, individual flow rates can deviate by as much as 50% from their fair share values, while with $W = 2$, the deviation from fair share is less than 25%.



(a) 1-bit (b) 2-bit (c) 4-bit

Figure 7.4: Flow rates for QCN rate control vs. the number of bits in QCN feedback. Flow rates for 6-bit feedback are shown in Figure 7.2(b).

Impact of multi-bit feedback: Next, we evaluate the benefits of multi-bit feedback on congestion control. Figure 7.3 and 7.4 show the queue and the individual rates of all the flows vs. the number of bits in the QCN congestion feedback. As it can be seen in the figure, single bit feedback notification leads to an unstable queue with occupancy as high as twice of the desired Q_{off} and flow rates as high as 70% more than the fair share value. As we increase the number of bits in the QCN feedback, the rate and queue stability improve. We get the best results with 6-bit congestion feedback (Figures 7.2(b) and 7.1(b)), with approximately 10% lower queue variation and 20% lower throughput variation compared to results with 4-bit feedback.

As shown, congestion control can benefit from fine-grained multi-bit notification as well as the derivative term in QCN feedback. Besides, QCN also benefits from direct feedback from the congestion point as opposed to ECN feedback, which requires at least one roundtrip time to convey the congestion information back to the sender. As we illustrate later (§7.4.5), exact location of the congestion can be used to make more intelligent routing decisions.

Next, we describe our approach to enable transmission of QCN notifications across an L3 network and highlight the required modifications for processing QCN feedback on the host side. Subsequently, we describe our window-based congestion control algorithm, *QCN-CC* , which relies on the above modifications.

7.4.2 L2 QCN in an L3 Network

One of the major contributions of our work is identifying simple fabric and host modifications that allows QCN's L2 notification frames to get routed properly in an L3 network. We leverage the standard L2 learning feature available on any L3 commodity switch today. L2 learning is a hardware feature in which a switch caches the source MAC address of a packet (data packet in QCN's case) along with the corresponding input switch port number. The switch can then properly forward an L2 packet (e.g. QCN notification frame) to a MAC address that has been already cached through the corresponding cached port number. Today's data center switches typically have L2 tables that can accommodate at the order of 100,000 MAC entries. This is more than sufficient for maintaining the cached MAC address of the sampled packet long enough before its corresponding congestion control frame traverses back in the reverse direction using the cached information. [2]

Hence, all needed for QCN's notification packets to get routed back to their L2 sources is preserving their L2 source MAC address throughout the fabric (i.e. don't over-write that value end-to-end) and turning on L2 learning. The caveat here is that switches can continue to simultaneously routed IP packets based on the IP table information while forwarding non-IP L2 frames based on the L2 table information that gets populated based on the L2 header of IP packets.

7.4.3 Host Modifications

The QCN standard ambitiously targeted an ultra-fast reaction mechanism at the hosts in order to promptly adjust the rates of the culprit flows at early onsets of congestion. Moreover, the rate recovery mechanism requires several timers that are best implemented in

[2]The worst case here is when as many packets as the capacity of the L2 table arrive with a different MAC address each at the highest speed (i.e. from all ports). Conservatively assuming 100,000 L2 entries and a 16-port 40Gbps switch, with a 0.5KB average packet size, it is impossible that an L2 cached entry could be evicted in less than $600\mu s$ from the time it was added, which is much more than the round-trip time in any reasonably designed data center today.

hardware for higher precision and lower overhead as opposed to their kernel-based counterpart. It is generally desirable to implement QCN's host-side mechanism entirely in hardware as a rate-based scheme based on the following assumptions: (i) the number of concurrently active flows at a host is small; (ii) sending a flow at the full line rate by default (instead of TCP's slow start) would help to minimize its completion time.

At the time, a hardware rate-limiter-based approach sounded reasonable given that the kernel networking latency was still relatively slow and that NIC vendors were already ramping up on better rate limiting designs and other advanced capabilities. The issue, however, is that years after the QCN standard was standardized, network operators still fear dealing with a complicated NIC that is not easy to configure (timers, byte counters, jitter ranges, active-increase, and hyper-active-increase increments) and might require some advanced congestion control understanding. In fact, we are not aware of any good QCN-capable NIC in the market that provides the QCN implementation in hardware and that scales well on rate limiting [168]. Sharing rate limiters across flows in an arbitrary random way subjects flows to potentially severe head of line blocking effects: different flows are exposed to different congestion signals and are essentially desired to send at different rates depending on the paths they traverse.

Our take on the hardware rate-based approach is multifold. Our intuition is that QCN's main advantage is in providing a rich multi-bit feedback signal directly to end hosts. We believe that other optimizations, such as packet-pacing at the host (being rate-based) or timer-triggered fast rate recovery, can be removed to simplify the design, and more importantly, to avoid the dependency on a limited set of specific NICs. In §7.5 we also illustrate why timers used in QCN rate control scheme are not necessary.

We next describe two use cases of our proposed changes to L2 QCN and host-side processing on kernel: (a) *QCN-CC* - QCN based Congestion Control, and (b) *QCN-LB* - QCN based Load Balancing. Note that, we did not make any attempts to optimize the QCN-based schemes and rather modified new designs in the recent literature to highlight the advantages of usign QCN as explicit congestion notification.

7.4.4 QCN-Based Congestion Control

In this section we discuss *QCN-CC* , a window-based congestion control, that uses QCN feedback to adjust the congestion window. *QCN-CC* preserves the core functionality of TCP such as slow start, additive increase in congestion avoidance, and recovery from packet losses. The key difference, however, is the way that *cwnd* is updated in response to congestion and in recovery phase, which is inspired by the original QCN rate control algorithm.

Similar to rate-based QCN, *QCN-CC* maintain a target window, $cwnd_T$. When *QCN-CC* receives a QCN feedback, it sets the $cwnd_T$ to its current *cwnd* and reduces *cwnd* and *ssthresh* in proportion to the QCN feedback as follows

$$cwnd_T \leftarrow cwnd \tag{7.5}$$

$$cwnd \leftarrow cwnd(1 - G_d \cdot F_b) \tag{7.6}$$

We set the weighted-factor, G_d, such that *cwnd* is reduced by three-fourths of its current value for the highest value of F_b.

Upon receiving a QCN congestion notification, *QCN-CC* enters the fast-recovery phase. The fast-recovery phases is similar to BIC-TCP [224] and helps to recover *cwnd*, after receiving a QCN feedback. Since QCN messages are received directly from the congestion point, fast-recovery mechanism can avoid conservative *cwnd* selection. At the end of each cycle of fast-recovery, when *QCN-CC* receives an ACK, it increases the *cwnd* exponentially and sets the congestion window to

$$cwnd \leftarrow cwnd + \frac{cwnd_T - cwnd}{2cwnd} \tag{7.7}$$

After 5 cycles of fast-recovery, *QCN-CC* goes to active increase phase and increments the *cwnd* similar to TCP Sack whenever it receives an ACK. We select a value of 5 for the fast-recovery threshold based on in its counterpart in rate-based QCN which is proven to perform well in practice. *QCN-CC* congestion control algorithm is described in Algorithm 5 in more details.

Algorithm 5 *QCN-CC* algorithm outline

on QCN Feedback:
$qcn\_fast\_recovery \leftarrow 0$
$target\_cwnd \leftarrow cwnd$
$cwnd \leftarrow cwnd \times (1 - \frac{Fb}{2^{bits+1}})$
$ssthresh \leftarrow ssthresh \times (1 - \frac{Fb}{2^{bits+2}})$
on ACK of new packet:
if $qcn\_fast\_recovery \geq 5$ **then**
    if $cwnd < ssthresh$ **then**
        $cwnd \leftarrow cwnd + 1$
    else
        $cwnd \leftarrow cwnd + \frac{1}{cwnd}$
else
    $cwnd \leftarrow cwnd + \frac{(target\_cwnd - cwnd)}{2 \times cwnd}$
    $qcn\_fast\_recovery++$

7.4.5 QCN-Based Load Balancing

We now describe *QCN-LB* , which is a simple host-based dynamic load balancer that uses QCN congestion feedbacks to make routing decisions. We borrow the core functionality of *QCN-LB* from FlowBender [121]. Similar to FlowBender, *QCN-LB* senses the extent of the congestion on the current route and tries to reroute the traffic to avoid congested or failed routes. The main difference between the two designs is that *QCN-LB* uses QCN feedbacks to detect congestions, while FlowBender relies on ECN marks. Whenever, a QCN congestion notification is received, *QCN-LB* checks the QCN feedback value and if F_b exceeds a certain threshold T it marks the route as congested. *QCN-LB* only reroutes the flow if the congestion is persistent, i. e. F_b exceeds the threshold for N consecutive notifications. As suggested in [121], we use VLAN tags as simple mechanism for the hosts to change the path of each flow. We refer the readers to [121, 190] to how the fabric and hosts should be configured to enable host-based rerouting in the network.

Compared to FlowBender, *QCN-LB* benefits from a much faster reaction time to congestion due to the direct QCN feedbacks. More importantly, *QCN-LB* can exploit the

Algorithm 6 *QCN-LB* algorithm outline

```
num_feedbacks ← 0
on QCN Feedback:
    F = Fb / 2^(bits+1)
if F ≥ T then
    num_feedbacks ← num_feedbacks + 1
    if num_feedbacks ≥ N then
        if qcn_cp_id ≠ dst_id then
            Change V
        num_feedbacks ← 0
```

information about the location of the bottleneck link in the feedbacks and avoid spurious reroutes. In a common scenario in datacenters, in which the congestion happens at the destination ToR, the bottleneck link cannot be avoided as all of the paths to the destination cross the congested link. In such scenarios, load balancers such as FlowBender would actually hurt the performance due to excessive packet reordering as highlighted in §7.5.2. *QCN-LB* , on the other hand, can check the source of the QCN notification and ignore the message if it is from a destination ToR. Another advantages of *QCN-LB* is that it avoids rerouting multiple large flows simultaneously since the QCN feedback messages for different flows are naturally jittered due to sampling.

QCN-LB is fairly simple to tune and requires only configuring two parameters T and N. In our evaluations, we observed that the *QCN-LB* performs well for wide range of T between 2% to 20%. Further, we set $N = 5$ to prevent rerouting of small flows or flows with smaller *cwnd*. Algorithm 6 describes the *QCN-LB* in more detail.

7.5 Evaluation

In this section, we evaluate the performance of our QCN-based designs for various network functions, using extensive packet-level simulations in ns2 [25]. More specifically, we compare the performance of *QCN-LB* (§7.4.5) and our QCN-based congestion control (§7.4.4) to other state-of-the-art schemes. First, we illustrate the value of QCN feedback as a direct

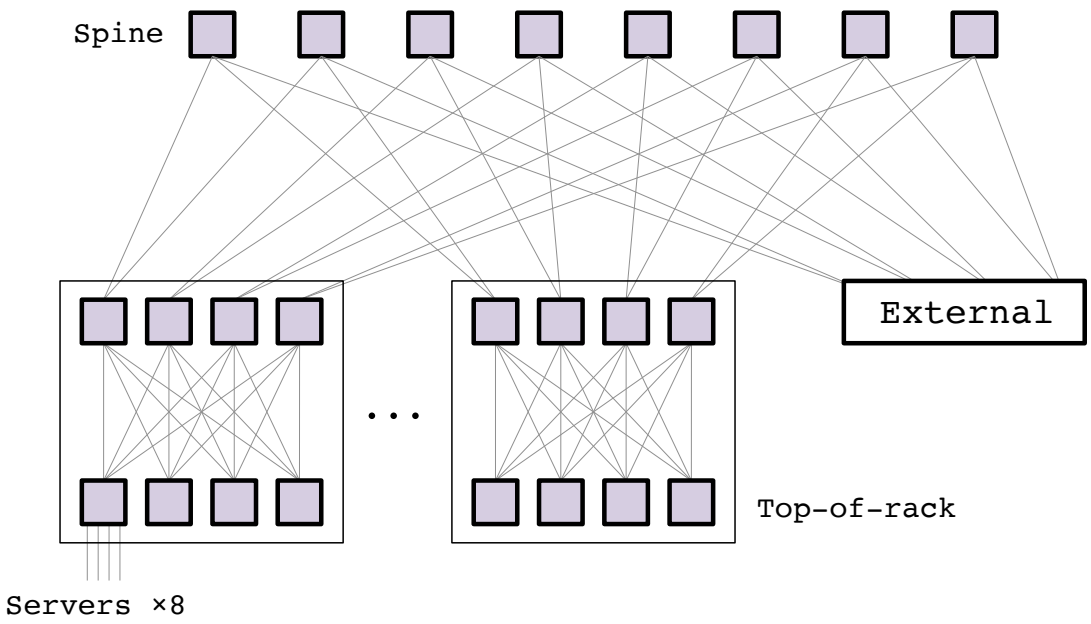


Figure 7.5: Fat-tree datacenter network topology.

congestion signal in load balancing as well as congestion control, using a series of micro-benchmarks. We then evaluate *QCN-CC*'s performance in a more realistic network running mix of workloads observed in typical production deployments.

7.5.1 Methodology

Topology: We use the fat-tree network [136] shown in Figure 7.5 in our simulations. The fabric interconnects 128 servers organized into four pods. Each pod consists of four aggregation switches and four top-of-rack (ToR) switches. Aggregation switches are connected to eight core switches resulting in a fabric with overall 4:1 oversubscription. Similar to Google's Jupiter architecture [192], the fabric is directly connected to the inter-cluster networking layer with an external cluster border router. Each pod is provided with a pool of 25% of aggregate intra-cluster bandwidth [192] for external connectivity.

We use 10Gbps point-to-point Ethernet links across our entire network. All the switches in the topology have a per-port buffer capacity of 1MB. We also configure the host delay

and intra-datacenter switching delay to be 1μs and 2μs, respectively. Thus, the minimum RTT between two servers on different pods of a datacenter is 14μs. In order to incorporate the effect of long RTT of inter-datacenter traffic, we set the propagation delay of the links connecting hosts to the external switch to $10ms$, resulting in RTT of roughly $\sim 20ms$ for datacenter-edge traffic.

Workloads: We simulate empirical workloads based on observed distributions in production data centers. In particular, we consider two flow size distributions from a cluster running data-mining workload [96] and a Hadoop cluster [177]. Both of the flow size distributions are heavy-tailed with majority of the flows being less than 10KB. The data-mining workload is more skewed and hence, is easier to handle because it is less likely to have multiple large flows competing for network resource concurrently.

We generate mix of inter- and intra-datacenter traffic with roughly 1:5 ratio similar to Facebook's production network [177]. In order to simulate the high utilization of the inter-datacenter fabric, we generate competing traffic originating from the external hosts. We keep the link utilization of the external links at about 80% [116]. For all our simulations, we select the source-destination pairs uniformly across all of the hosts. Furthermore, we use ECMP as our multipath routing scheme unless mentioned otherwise.

Performance evaluation: Similar to prior work [38, 42] we use average and tail Flow Completion Time (FCT) as our metric to evaluate the performance of *QCN-CC* . We compare *QCN-CC* to standard TCP, DCTCP [39], and DCQCN [232]. DCTCP leverages ECN to convey congestion information to the end hosts and adjusts the congestion window size based on the fraction of marked bytes. As WAN networks generally do not support end-to-end ECN marking, we only enable ECN marking on intra-datacenter switches. [3]

DCQCN [232] is another rate-based protocol that also relies on ECN marks. We adopt DCQCN*, a window-based version of DCQCN to for our performance comparisons. In DCQCN, the rate reduction mechanism is similar to DCTCP, while the rate recovery mech-

[3]We did evaluate with ECN enabled on all of the switches, end-to-end, and have found marking on the inter-datacenter fabric to have almost no impact on our results, as most of the congestion happens locally in the intra-datacenter fabric.

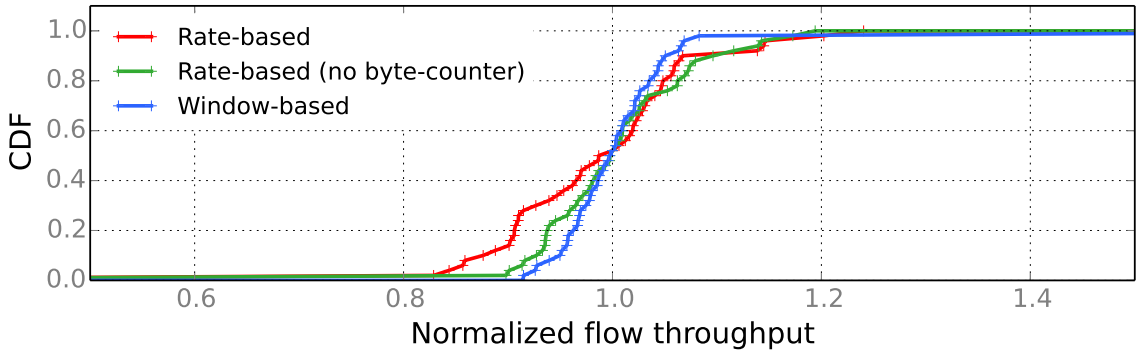
anism is same as the QCN rate-based scheme. For DCQCN*, we keep the rate reduction mechanism same as DCQCN, but we change the recovery mechanism to mimic that of TCP-BIC. For similar reasons as discussed earlier when comparing QCN rate-based scheme to *QCN-CC*, the performance of DCQCN and DCQCN* should be similar. We also evaluated the performance of *QCN-CC* against TIMELY [156]. Consistent with the findings in [233], the performance of TIMELY was worse than DCQCN, and therefore, we have omitted TIMELY's results for the sake of brevity.

On the load balancing front, we compare *QCN-LB*'s results to ECMP and FlowBender [121]. ECMP is the natively supported load balancing mechanism on Ethernet switches today, based on oblivious hashing, and FlowBender uses ECN marks to sense the congestion on the path to destination to dynamically reroute connections at the roundtrip timescale.

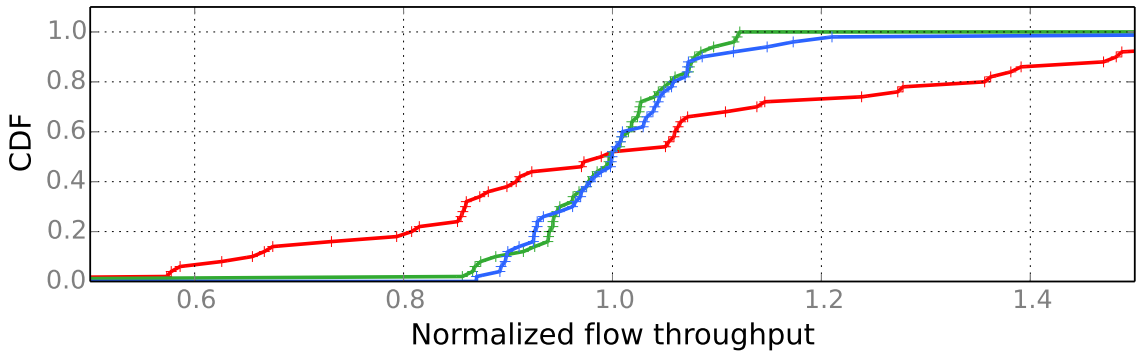
Parameters: We use ns-2 FullTCP Sack implementation as our standard TCP protocol and build other schemes on top of it. For DCTCP, we set the parameters as described in [39]: (1) g, the factor for exponential weighted averaging, is set to $\frac{1}{16}$, and (2) K, the buffer occupancy threshold for setting the CE-bit, is set to $90KB$ (typical for 10 Gbps links). For a fair comparison, for QCN-based schemes, we set Q_{off} equal $90KB$. All other TCP functionalities are the same as in FullTCP Sack implementation.

For FlowBender, we use the settings suggested in [121]: (1) N, the number of congested RTTs before a sender reroutes the traffic, is set to 1, and (2) T, the threshold for the fraction of marked acks to consider a route as congested, is set to 5%. As recommended in the paper, we put the connection in a *locked* state after 5 consecutive reroutes to avoid excessive packet reordering.

An important factor in flow completion times is the Retransmission Timeout (RTO) of TCP as dropped packets are retransmitted after expiration of an RTO. We use commonly used RTO values for inter and intra datacenter traffic [206,229]. For intra-datacenter traffic, we set the RTO value as $1ms$. For inter-datacenter traffic, we use an RTO value of $100ms$.



(a) RTT = 100μs



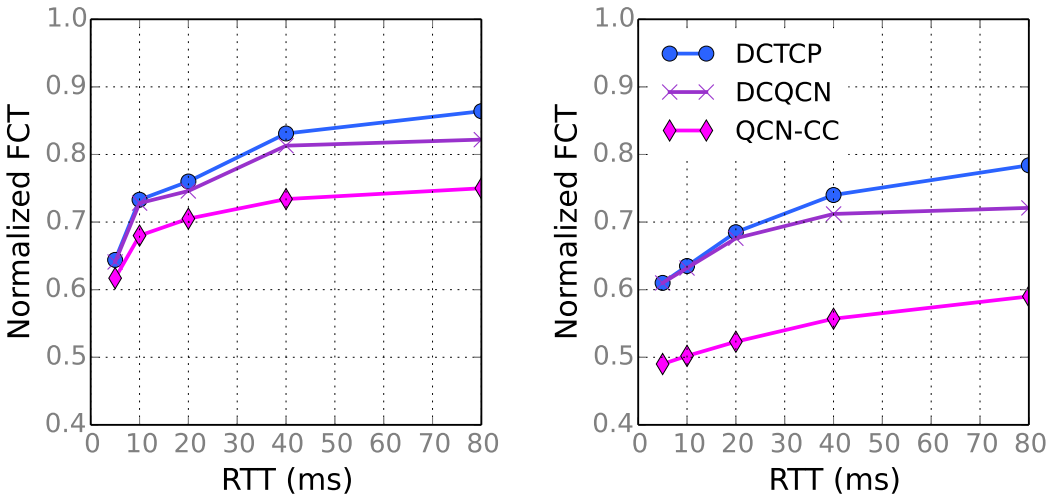
(b) RTT = 400μs

Figure 7.6: CDF of individual flow rates with rate-based and Window-based QCN.

7.5.2 Benchmark

Window-based QCN: To compare *QCN-CC* to its rate-based counterpart as described in [37], we simultaneously generate 50 large flows sharing a single bottleneck link and measure the individual throughput. We repeat the same experiment with two different RTTs i. e. 100 and 400μs. The CDF of the individual flow rates normalized by the median throughput is shown in Figure 7.6. For a fair comparison, we show the results achieved by rate-based QCN with and without the byte-counter to highlight its impact on the fairness of QCN.

QCN-CC yields similar performance as the rate-based QCN congestion control without



(a) Average (b) 99th percentile

Figure 7.7: Normalized FCT of large transfers for various RTTs.

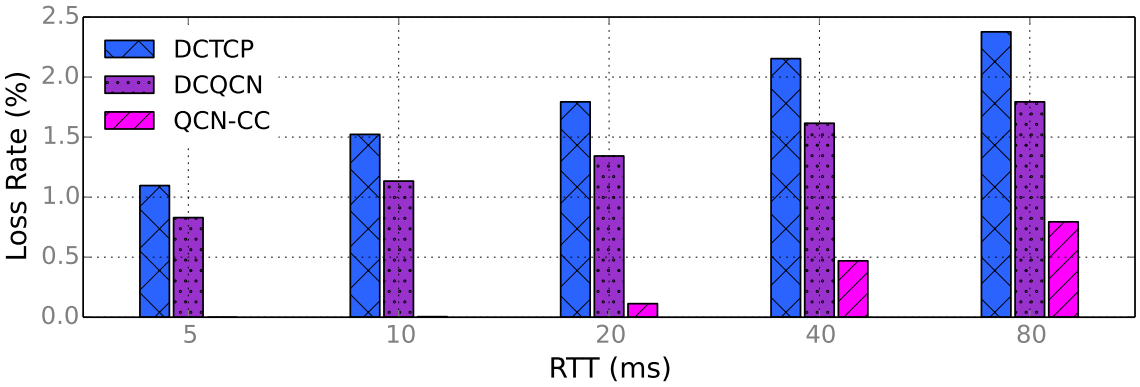
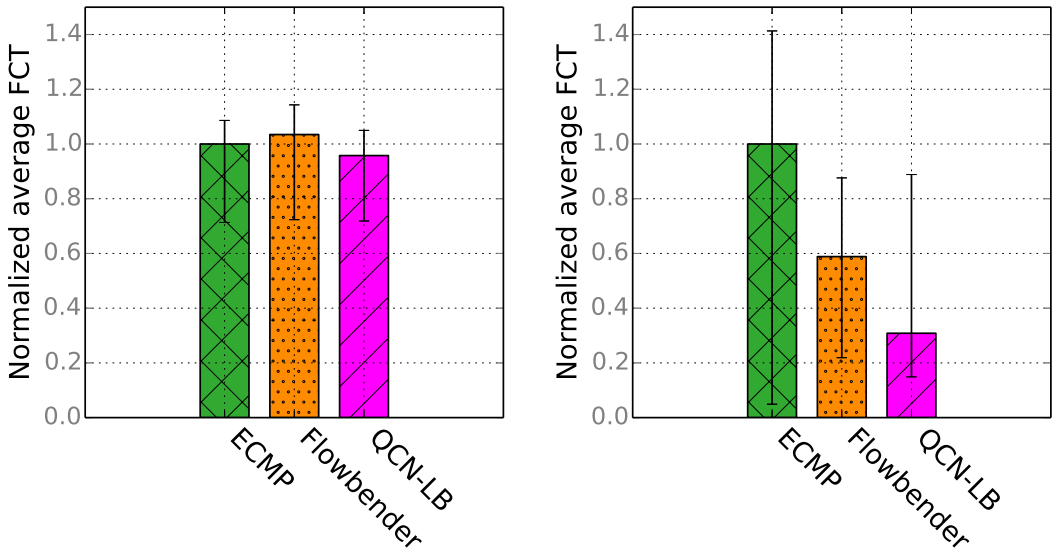


Figure 7.8: Packet loss rate at the congestion point.

the byte counter. However, adding byte counter leads to significant unfairness, especially at higher RTT values. The byte counter unfairly prioritizes flows with higher rates which is well-known [120]. The window-based *QCN-CC* algorithm overcomes these issues.

Edge Caching: In order to illustrate the importance of direct feedback of QCN, we consider a scenario in which the congestion happens at the edge of the network, which is common for networks with edge caches for content delivery.



(a) Congesstion at destination ToR

(b) Congestion at source ToR

Figure 7.9: Normalized average FCT. Error bars show the maximum and minimum completion times.

To model such a scenario, we simulate a 20:1 incast with varying RTT between 20ms to 80ms. We set the delay from the transmitters to the congestion point to $50\mu s$. Each sender, transmits large transfers of 50MB to a single host. Once each transfer is complete, senders immediately initiate the next transfer. We run the simulation for 10,000 transfers and compute the average and 99th percentile FCT. Figure 7.7 shows the results for *QCN-CC*, DCTCP, and DCQCN*. The numbers are normalized by the FCT achieved by *TCP-DropTail* . *QCN-CC* leads to almost 10% smaller average FCT and 20% smaller tail FCT when compared to DCTCP and DCQCN. This is mostly due to roudtrip timescale reaction of DCTCP and DCQCN* to the congestion, which is at order of tens of milliseconds. *QCN-CC*, on the other hand, exploits the direct congestion feedback of QCN and reacts much faster that other scheme. This also lowers the drop rate ($< 1\%$) at the congestion points (Figure 7.8).

Load balancing efficiency:

We now show the benefits of QCN's direct feedback in load balancing. We consider a 4:1 oversubscribed leaf-spine topology with 8 spines switches. We simulate two different scenarios to evaluate the performance of *QCN-LB* . In the first experiment, we generate large transfers from servers in one rack destined to single host in another rack. In this scenario, as congestion happens at the destination ToR, each flow always experiences congestion regardless of the selected path by the load balancer. In fact, the dynamic reroutings by the load balancer could actually hurt the performance of TCP due to packet re-ordering.

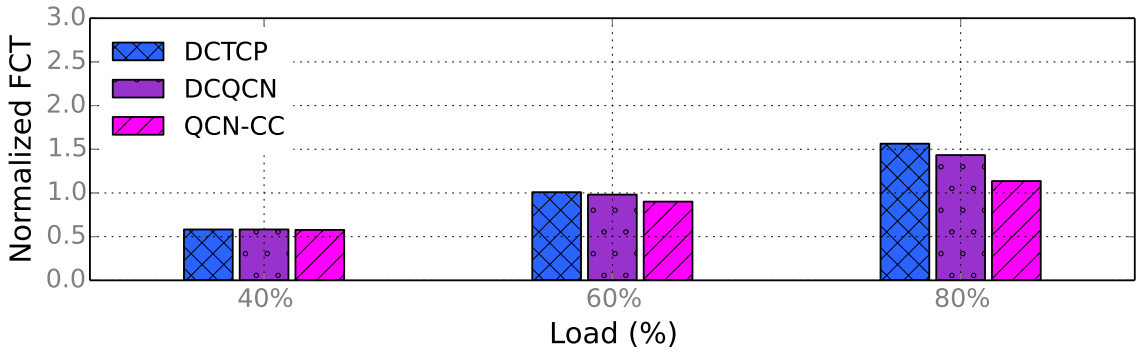
In the other experiment, servers in one rack, send large transfers to all the other servers. Here, the congestion happens at the source ToR, and routing decisions by the load balancer directly impact the flow completion times. As all flows are of equal size, more efficient load balancer, improves both the average and the maximum flow completion times. Figure 7.9 shows the average flow completion time for the two experiments, comparing the performance of ECMP, *QCN-LB*, and FlowBender. The error bars show the maximum and minimum completion times. As shown in Figure 7.9(a), FlowBender performs slightly worse than other schemes mostly due to spurious rerouting and packet-reordering, while *QCN-LB* does not react to the QCN feedback from the destination ToR and avoids unnecessary reroutings. In the second scenario (7.9(b)), *QCN-LB* , leads to 2x smaller flow completion times than FlowBender, due to its fast reaction to congestion.

7.5.3 Overall Performance

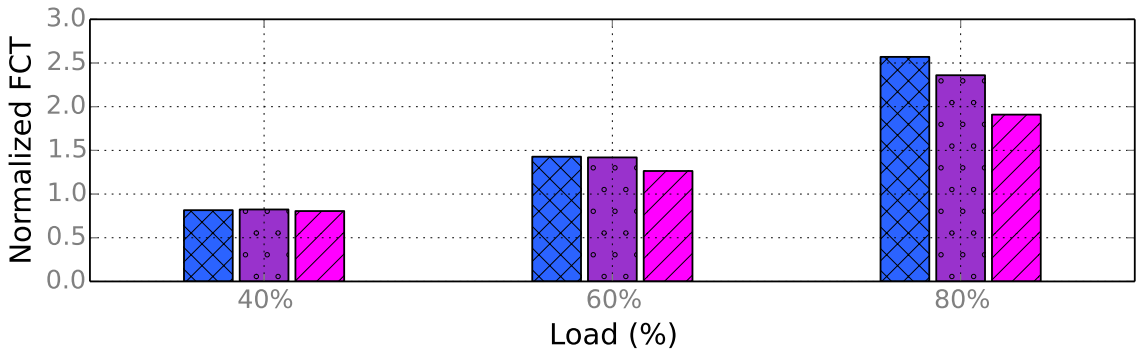
In this section, we show the overall performance of *QCN-CC* in the network shown in Figure 7.5. We use two different realistic workloads with traffic scenarios where there is a mix of intra- and inter-datacenter flows.

Figures 7.11 and 7.10 show the average completion times for intra-datacenter traffic achieved by each scheme as the fabric load varies from 40% to 80% for the two workloads. The results are obtained for simulations with more than 500,000 flows and normalized to the FCT achieved with *TCP-DropTail* at 40% load. We break down the results for small $[0, 10KB]$, medium $[10KB, 100KB]$, and large (> 100 KB) flows.

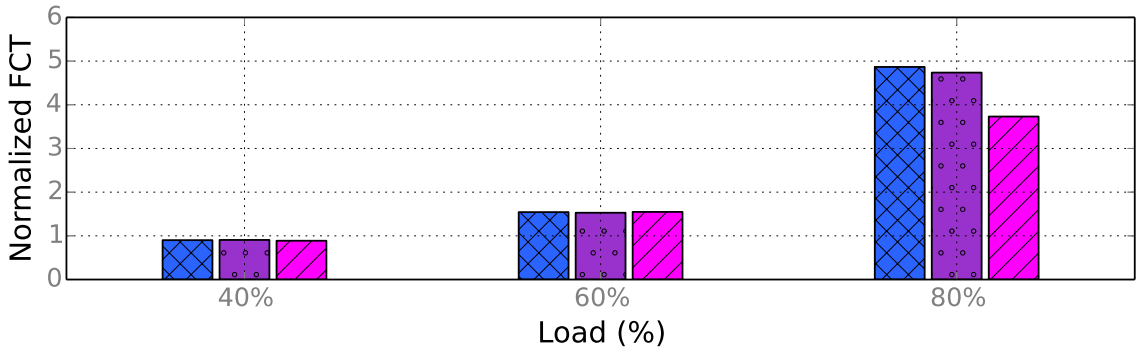
For the data-mining workload, *QCN-CC* achieves about $4-5\times$ lower average FCT com-



(a) $< 10KB$

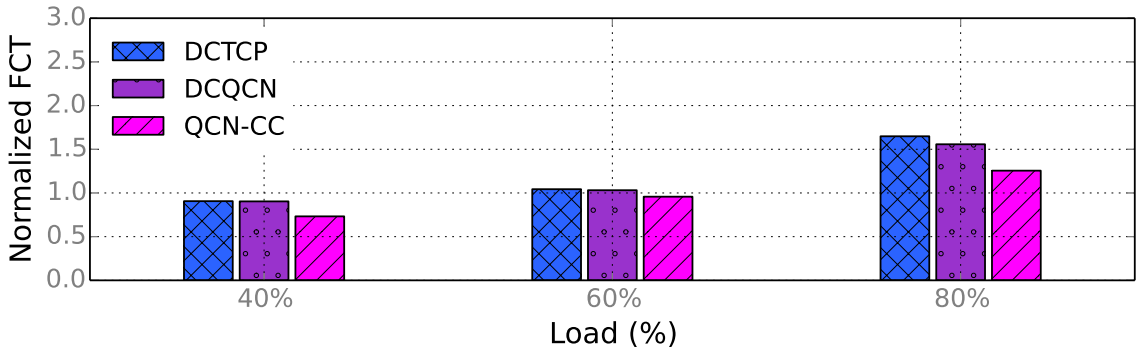


(b) $(10K, 100KB)$

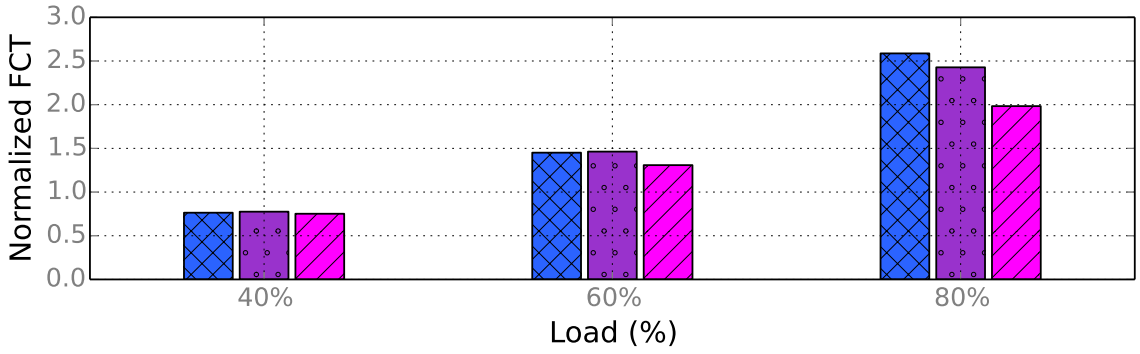


(c) $> 100KB$

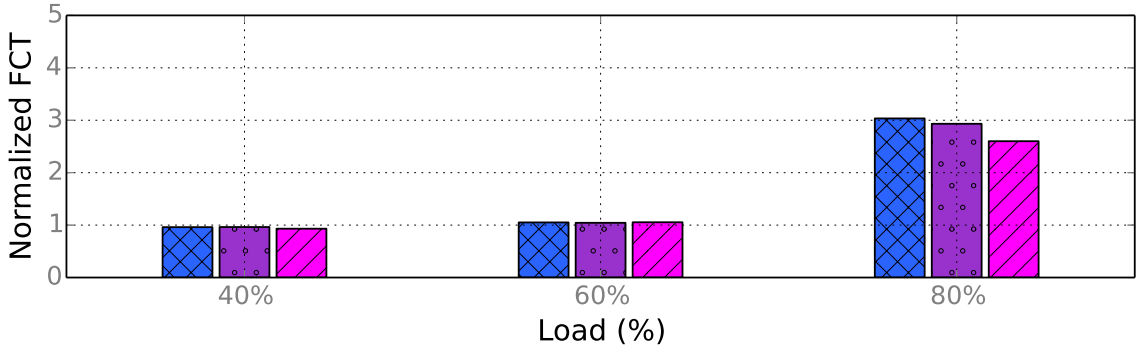
Figure 7.10: Average flow completion times for data mining workload. Numbers are normalized to FCT achieved by TCP-DropTail at 40% load. Note that the range of the y-axis is different for (c).



(a) $< 10KB$

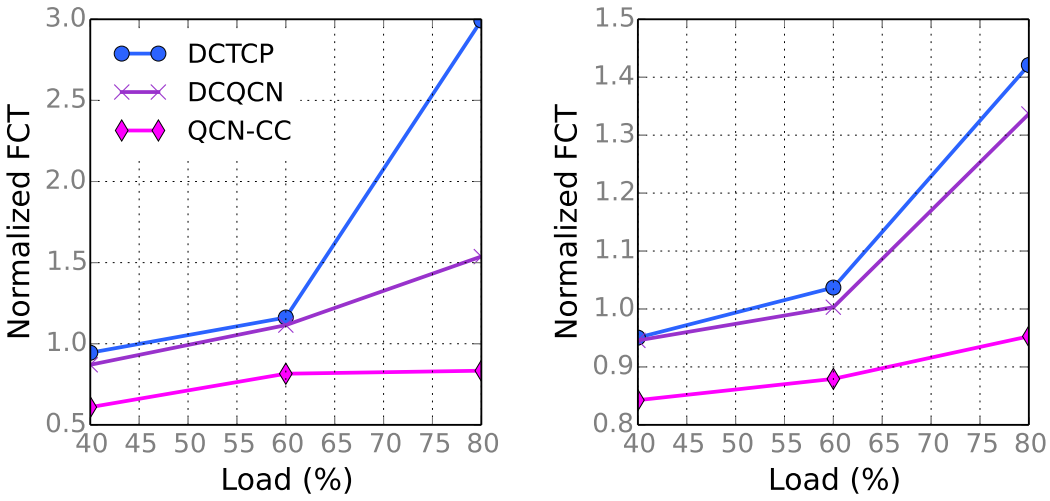


(b) $(10KB, 100KB)$



(c) $> 100KB$

Figure 7.11: Average flow completion times for Facebook workload. Numbers are normalized to FCT achieved by TCP-DropTail at 40% load.



(a) data mining

(b) facebook

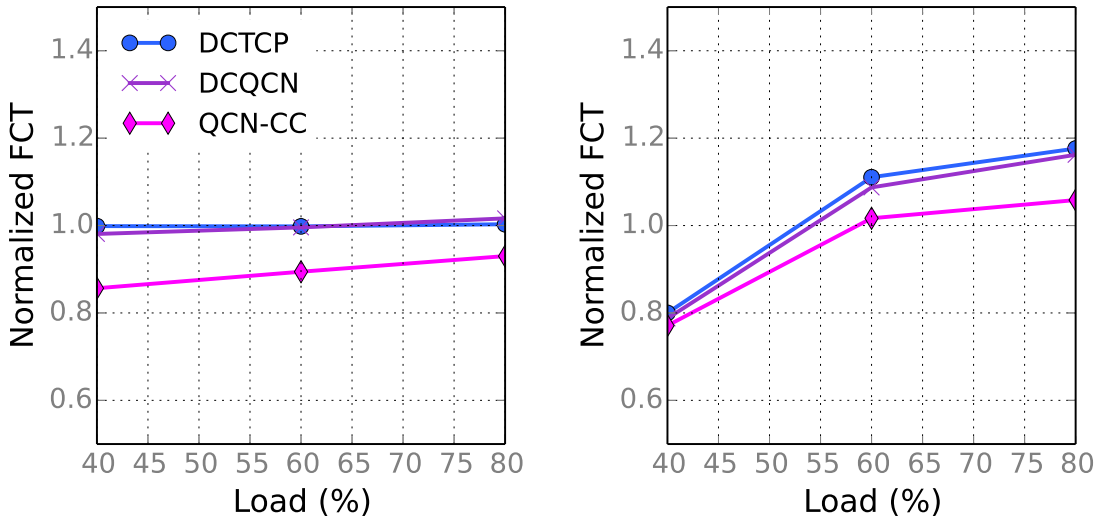
Figure 7.12: 99th percentile FCT for small intra-datacenter flows.

paring to *TCP-DropTail* for small and medium flows. Note that TCP-DropTail does not appear in Figure 7.10 as its performance is outside the plotted range. As expected, DCTCP and DCQCN* have comparable performance as they both use ECN as the congestion feedback. However, *QCN-CC* outperforms DCQCN, acheiving 30% lower average FCT across all flow sizes at high load.

For the Facebook workload, the average flow completion times for *QCN-CC* are $4-5\times$ smaller than *TCP-DropTail* for small flows. Similar to the other workload, *QCN-CC* achieves $16-31\%$ lower average FCT comparing to DCTCP and $12-24\%$ lower than DCQCN.

The benefits of *QCN-CC* are more apparent in the tail latencies of small flows. Figure 7.13 shows the 99.9 percentile of FCTs for small flows for the two workloads. For data-mining workload, *QCN-CC* leads to $5\times$, $3.5\times$, and $2\times$ lower tail latency at 80% fabric load comparing to *TCP-DropTail* , DCTCP, and DCQCN, respectively. For Facebook workload, *QCN-CC* outperforms DCQCN and achieves 50% lower tail latency at high load.

It is worth mentioning that *QCN-CC* significantly improves both average and tail la-



(a) data mining

(b) facebook

Figure 7.13: Overall average FCT for inter-datacenter flows.

tencies of intra-datacenter flows, while improving the performance of the inter-datacenter flows as well. Figures 7.13(a) and 7.13(b) show the overall average FCT for all the inter-datacenter traffic for data-mining and Facebook workload, respectively. As it can be seen, *QCN-CC* improves the overall average FCT by $\sim 10\%$ comparing to other schemes. Note that inter-datacenter FCTs do not vary much as we always maintain the link utilization of the external links at 80% regardless of the intra-datacenter fabric load by generating competing flows.

Part III

Conclusions

This thesis presented novel systems and algorithms for wireless and datacenter networks that enable and accelarate new applications. Below, we highlight general conclusions and possible future directions.

Adaptive Wireless Multicast

In Chapters 2 and 3, we presented the design and large-scale experimental evaluation of the *AMuSe* system for large scale content delivery via wireless multicast. *AMuSe* only needs access to the channel quality measurements such as RSSI and Packet Delivery Ratio on WiFi devices and can be implemented as an application layer protocol on existing devices. Chapter 2 focussed on the *AMuSe* feedback scheme which provides a scalable and efficient mechanism to monitor the quality of WiFi multicast services to a large group of users. One of the key observations in our work is the presence of a few *abnormal nodes*, which experience low service quality even at very low multicast transmission bitrate. Existing feedback alternatives only allow tuning the network parameters for multicast according to the weakest receiver, which results in low network utilization. *AMuSe* can overcome this obstacle by collecting reports from a sufficient number of receivers.

In Chapter 3, we designed a novel multicast rate adaptation algorithm (*MuDRA*) that utilizes the *AMuSe* feedback scheme to provide high throughput while satisfying service requirements. The design of *MuDRA* is based on insights learned from extensive experimental observations. *MuDRA* detects when the system operates at the target rate, which is the optimal rate at which receivers *MuDRA's* performance on the ORBIT testbed with hundreds of nodes shows that it can reliably support applications such as large scale multimedia content delivery.

In Chapter 4, we focused on dynamically tuning multicast transmission rate and video rate for enhancing video QoE. We presented the *DYVR-M* and *DYVR-A* algorithms for maximizing the video rate while ensuring low loss rate, buffer underflows, and video rate switches. The *DYVR* algorithms do not require future estimates of channel state at the receivers. They can be easily incorporated within existing ABR video streaming frameworks and have low computational complexity. Our analysis shows that *DYVR-M* and

DYVR-A can achieve average utility within an additive term $O(\frac{1}{W})$ of the optimal utility, where W affects how closely the QoE constraints are satisfied. Extensive simulations and experimental evaluations indicate that *DYVR-M* and *DYVR-A* outperform other video rate adaptation algorithms such as BBA or PBA.

The *QPRC* and *QARC* problem formulations and the *DYVR* algorithms provide a flexible method for network operators, video providers, or receivers to specify and meet their QoE requirements. With new standards such as [114] enabling server-based push to clients to alleviate network congestion, we believe that the presented architecture could provide a way to exploit multicasting opportunities and improve video streaming performance over wireless networks. In future work, we plan to conduct experimental evaluations with the 802.11n and 802.11ac standards which may provide enhanced performance. We also plan to explore Scalable Video Coding (SVC) techniques and reformulate the video QoE optimization problem for SVC.

In Chapter 5 we presented the *Dynamic Monitoring* (*DyMo*) system for large scale monitoring of LTE eMBMS services, based on the concept of *Stochastic Group Instructions*. Our extensive simulations show that *DyMo* achieves accurate, close to optimal, estimation of the maximum SNR threshold so that only a small number of UEs (User Equipments) with SNR below the threshold suffer from poor service. It can improve the spectral efficiency for eMBMS operation while adding a low reporting overhead.

Some possible future directions include further optimizations of multicast rate adaptation algorithms distinguishing between losses due to channel conditions and collisions. Moreoever, we will also consider evaluation of proposed ideas in *AMuSe* with the multicast specifications in the new IEEE 802.11aa standard. We believe that techniques such as *DyMo* are attractive monitoring for a variety of large scale wireless systems such as Machine-to-Machine communications and Internet of Things (IoT) networks and our future work will explore some of these directions.

Datacenter Networks

In Chapter 6, we presented a unique hardware-software system architecture to integrate circuit-based optical modules with datacenters' Ethernet network. We developed an optical multicast system, to enable efficient physical layer multicast through passive optical splitters. It is built on a hybrid architecture that combines traditional electronic packet switching with optical circuit switching networks. The optical space switch is the switching fabric of the optical network and also the connectivity substrate of splitters. Network management and configurations are handled through a 3-layered SDN control plane. We built a hardware prototype and developed a simulation environment to evaluate the performance of the system.

Optical multicast delivers multicast flows to all receivers simultaneously irrespective of the multicast group size, similar to IP multicast. However, optical multicast performs a more efficient multicast in data centers since: (i) it is built on an optical circuit switching substrate with lower energy consumption than electronic packet switching, and (ii) does not require applying complex configurations on all switches and routers to enable IP multicast since multicast group management and tree formation is handled by the SDN controller. Compared to application layer multicast using peer-to-peer methods, optical multicast achieves considerably higher throughput for large range of flows sizes (up to 1.5 GB) with fixed, minimal connection overheads. Our future work includes enabling optical multicast in an inter data center network to perform long-haul one-to-many virtual machine migration and utilizing optical multicast system architecture to enable optical incast.

In Chapter 7, we recast L2 QCN, already commonly available on existing Ethernet switching platforms, as an effective mechanism that is very powerful for better congestion control and load balancing purposes in L3 networks. The proposed changes encompass (i) simple switch configuration changes for enabling L2 learning that would suffice for properly forwarding QCN's feedback frames as well as (ii) straightforward TCP kernel changes for processing the notification packets within the kernel. We dissect the various aspects of QCN and retain those that are more critical for its performance, hence simplifying its

deployability and configuration.

Our simulations, based on real world workloads, show that *QCN-CC* significantly reduces the average latency of short flows by at least 15% and tail latency by 2× when compared to DCTCP and DCQCN. Moreover, our load balancing scheme *QCN-LB* takes clear advantage of the congestion point location information piggybacked in the QCN signal to avoid rerouting flows when rerouting can only hurt. Furthermore, QCN's direct feedback yields 2× smaller flow completion times for large flows as compared to FlowBender. While we demonstrate QCN's importance by showcasing its value via two simple congestion control and load balancing algorithms, there are other opportunities to design other schemes that can further leverage the QCN feedback such as telemetric systems for datacenters.

Part IV

Bibliography

Bibliography

[1] "AngularJS." [Online]. Available: https://angularjs.org/

[2] "Apache spark: Lightning-fast cluster computing," https://spark.apache.org.

[3] "Calient S320 optical circuit switching," http://www.calient.com.

[4] "Cisco global cloud index: Forecast and methodology: 2015 – 2020," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf.

[5] "D-ITG: Distributed Internet Traffic Generator," http://traffic.comics.unina.it/software/ITG/.

[6] "Django project." [Online]. Available: https://www.djangoproject.com/

[7] "Herd," https://github.com/russss/Herd.

[8] "Iperf," https://iperf.fr/.

[9] "JGroups," http://www.jgroups.org/.

[10] "Minstrel," https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel.

[11] "NACK-Oriented Reliable Multicast (NORM)," http://www.nrl.navy.mil/itd/ncs/products/norm.

[12] "NGINX." [Online]. Available: https://www.nginx.com/resources/wiki/

[13] "NS3," http://www.nsnam.org/.

[14] "ORBIT testbed," http://orbit-lab.org/.

[15] "PhotonDesign: A 1×8 planar MMI coupler," http://www.photond.com/products/fimmprop/fimmprop_applications_03.htm.

[16] "PLC optical splitter," http://www.fiberstore.com/1x64-fiber-plc-splitter-with-fan-out-kits-p-11606.html.

[17] "Polatis series 6000," http://www.polatis.com.

[18] "PostgreSQL." [Online]. Available: http://www.postgresql.org/

[19] "Twitter murder: Fast data center code deploy using bittorrent," http://engineering.twitter.com/2010/07/murder-fast-datacenter-code-deploys.html.

[20] "Video dataset," https://media.xiph.org/video/derf/.

[21] "Yinzcam," http://www.yinzcam.com/.

[22] "802.1Qau congestion notification," 2010, http://www.ieee802.org/1/pages/802.1au.html.

[23] "Cisco, white-paper, optimizing enterprise video over wireless LAN, 2010," 2010. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/wireless/ps6302/ps8322/ps10315/ps10325/white_paper_c11-577721.pdf

[24] "Cisco, white-paper, Cisco connected stadium Wi-Fi solution," 2011. [Online]. Available: http://www.cisco.com/web/strategy/docs/sports/c78-675063_dSheet.pdf

[25] "NS-2 network simulator," 2011, http://nsnam.sourceforge.net/wiki/index.php/Main_Page.

[26] "3GPP TS 37.320 V12.2.0 , 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Universal Terrestrial Radio Access

(UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRA); Radio measurement collection for Minimization of Drive Tests (MDT); Overall description; Stage 2 (Release 12)," Sept. 2014. [Online]. Available: http://www.3gpp.org/DynaReport/37320.htm

[27] "3GPP TS 26.346 V13.1.0, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs (Release 13)," June 2015. [Online]. Available: http://www.3gpp.org/DynaReport/26346.htm

[28] "Apple http streaming," 2017. [Online]. Available: https://developer.apple.com/streaming

[29] "Cisco visual networking index: Forecast and methodology, 2016-2021," 2017. [Online]. Available: http://www.cisco.com

[30] "Microsoft smooth streaming," 2017. [Online]. Available: http://www.iis.net/downloads/microsoft/smooth-streaming

[31] J. Ababneh and O. Almomani, "Survey of error correction mechanisms for video streaming over the internet," in *IJACSA*, vol. 5, 2014, pp. 155–161.

[32] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," *Proc. ACM SIGCOMM'04*, 2004.

[33] I. F. Akyildiz, Y.-B. Lin, W.-R. Lai, and R.-J. Chen, "A new random walk model for PCS networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 7, pp. 1254–1260, 2000.

[34] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM'08*, 2008.

[35] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *Proc. USENIX NSDI'10*, 2010.

[36] O. Alay, T. Korakis, Y. Wang, and S. Panwar, "Dynamic rate and FEC adaptation for video multicast in multi-rate wireless networks," *ACM Mobile Netw. and Appl.*, vol. 15, no. 3, pp. 425–434, 2010.

[37] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, "Data center transport mechanisms: Congestion control theory and IEEE standardization," in *Proc. IEEE Allerton'08*, 2008.

[38] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM SIGCOMM'14*, 2014.

[39] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proc. ACM SIGCOMM'11*, 2011.

[40] M. Alizadeh, A. Kabbani, B. Atikoglu, and B. Prabhakar, "Stability analysis of QCN: the averaging principle," in *Proc. ACM SIGMETRICS'11*, 2011.

[41] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center," in *Proc. USENIX NSDI'12*, 2012.

[42] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *Proc. ACM SIGCOMM'13*, 2013.

[43] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella, "The totem single-ring ordering and membership protocol," *ACM Trans. Comput. Syst.*, vol. 13, no. 4, pp. 311–342, Nov. 1995.

[44] A. Aziz, D. Starobinski, P. Thiran, and A. El Fawal, "EZ-Flow: Removing turbulence in IEEE 802.11 wireless mesh networks without message passing," in *Proc. ACM CoNEXT'09*, 2009.

[45] E. Baik, A. Pande, and P. Mohapatra, "Cross-layer coordination for efficient contents delivery in LTE eMBMS traffic," in *Proc. IEEE MASS'12*, 2012.

[46] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "A quest for an internet video quality-of-experience metric," in *ACM HotNets'12*, 2012.

[47] F. Balus, M. Pisica, N. Bitar, W. Henderickx, and D. Stiliadis, "Software driven networks: Use cases and framework," *IETF, Internet Draft*, 2011.

[48] A. Basalamah, H. Sugimoto, and T. Sato, "Rate adaptive reliable multicast MAC protocol for WLANs," in *IEEE VTC'06*, 2006.

[49] G. Baxter, S. Frisken, D. Abakoumov, H. Zhou, I. Clarke, A. Bartos, and S. Poole, "Highly programmable wavelength selective switch based on liquid crystal on silicon switching elements," in *Proc. OSA OFC'06*, 2006.

[50] Y. Bejerano, J. Ferragut, K. Guo, V. Gupta, C. Gutterman, T. Nandagopal, and G. Zussman, "Scalable WiFi multicast services for very large groups," in *IEEE ICNP'13*, 2013.

[51] ——, "Experimental evaluation of a scalable WiFi multicast scheme in the ORBIT testbed," in *Proc. GENI Research and Educational Experiment Workshop (GREE'14) (invited)*, 2014.

[52] Y. Bejerano, V. Gupta, C. Gutterman, and G. Zussman, "AMuSe: Adaptive multicast services to very large groups project overview," in *Proc. IEEE ICCCN'16 (invited)*, 2016.

[53] Y. Bejerano, C. Raman, C.-N. Yu, V. Gupta, C. Gutterman, T. Young, H. Infante, Y. Abdelmalek, and G. Zussman, "DyMo: dynamic monitoring of large scale LTE-multicast systems," *IEEE/ACM Trans. Netw. (under review)*, 2017.

[54] ——, "DyMo: dynamic monitoring of large scale LTE-multicast systems," in *Proc. IEEE INFOCOM'17*, 2017.

[55] ——, "DyMo: dynamic monitoring of large scale LTE-multicast systems," in *arXiv:1601.06425 [cs.NI]*, 2017.

[56] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM IMC'10*, 2010.

[57] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. ACM CoNEXT'11*, 2011.

[58] C. Bettstetter, "Mobility modeling in wireless networks: categorization, smooth movement, and border effects," *Mobile Comput. Commun.*, vol. 5, no. 3, pp. 55–66, 2001.

[59] J. Bicket, "Bit-rate selection in wireless networks," *PhD thesis, MIT*, 2005.

[60] B. Birand, H. Wang, K. Bergman, and G. Zussman, "Measurements-based power control - a cross-layered framework," in *Proc. OSA OFC'13*, 2013.

[61] D. Borthakur, "The Hadoop Distributed File System: Architecture and design," 2007, http://hadoop.apache.org/core/docs/current/hdfsdesign.pdf.

[62] J. Bowers, "Low power 3D MEMS optical switches," in *Proc. IEEE OMN'09*, 2009.

[63] M. Burrows, "The chubby lock service for loosely-coupled distributed systems," in *Proc. USENIX OSDI'06*, 2006.

[64] Y. Cai, S. Lu, L. Zhang, C. Wang, P. Skov, Z. He, and K. Niu, "Reduced feedback schemes for LTE MBMS," in *Proc. IEEE VTC'09*, 2009.

[65] R. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments," in *Proc. IEEE ICPP'11*, 2011.

[66] J. Cao, C. Guo, G. Lu, Y. Xiong, Y. Zheng, Y. Zhang, Y. Zhu, and C. Chen, "Datacast: a scalable and efficient reliable group data delivery service for data centers," *IEEE J. Sel. Areas Commun.*, vol. 12, no. 31, pp. 2632–2645, 2012.

[67] R. Chandra, S. Karanth, T. Moscibroda, V. Navda, J. Padhye, R. Ramjee, and L. Ravindranath, "DirCast: A practical and efficient Wi-Fi multicast system," in *Proc. IEEE ICNP'09*, 2009.

[68] J. Chen, M. Chiang, J. Erman, G. Li, K. Ramakrishnan, and R. K. Sinha, "Fair and optimal resource allocation for LTE multicast (eMBMS): group partitioning and dynamics," in *Proc. IEEE INFOCOM'15*, 2015.

[69] J. Chen, A. Ghosh, J. Magutt, and M. Chiang, "Qava: quota aware video adaptation," in *Proc. ACM CoNEXT'12*, 2012.

[70] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A tutorial on video quality assessment," *IEEE Commun. Surv. & Tut.*, vol. 17, no. 2, pp. 1126–1165, 2015.

[71] H.-T. Chiao, S.-Y. Chang, K.-M. Li, Y.-T. Kuo, and M.-C. Tseng, "WiFi multicast streaming using AL-FEC inside the trains of high-speed rails," in *IEEE BMSB'12*, 2012.

[72] K. Chintalapudi, A. Iyer, and V. Padmanabhan, "Indoor localization without the pain," in *Proc. ACM MOBICOM'10*, 2010.

[73] N. Choi, Y. Seok, T. Kwon, and Y. Choi, "Leader-based multicast service in IEEE 802.11v networks," in *IEEE CCNC'10*, 2010.

[74] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM SIGCOMM'11*, 2011.

[75] B. Cohen, "The bittorrent protocol specification," 2008, http://bittorrent.org/beps/bep_0003.html.

[76] R. Combes, A. Proutiere, D. Yun, J. Ok, and Y. Yi, "Optimal rate sampling in 802.11 systems," in *IEEE INFOCOM'14*, 2014.

[77] M. J. Connelly, "Semiconductor optical amplifiers," *Springer*, 2002.

[78] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *Int. J. Hum. Comput. St.*, vol. 64, no. 8, pp. 637–647, 2006.

[79] R. Crepaldi, J. Lee, R. Etkin, S.-J. Lee, and R. Kravets, "CSI-SF: Estimating wireless channel state using CSI sampling and fusion," in *IEEE INFOCOM'12*, 2012.

[80] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[81] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[82] S. Deb, S. Jaiswal, and K. Nagara, "Real-time video multicast in WiMAX networks," in *IEEE INFOCOM'08*, 2008.

[83] P. Devkota and A. N. Reddy, "Performance of quantized congestion notification in TCP incast scenarios of data centers," in *Proc. IEEE MASCOTS'10*, 2010.

[84] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the ip multicast service and architecture," *IEEE Netw.*, vol. 14, no. 1, pp. 78–88, 2000.

[85] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM'11*, 2011.

[86] N. Dukkipati, "Rate control protocol (RCP): Congestion control to make flows complete quickly," Ph.D. dissertation, Stanford University, 2007.

[87] J. Erman and K. K. Ramakrishnan, "Understanding the super-sized traffic of the super bowl." in *Proc. ACM IMC'13*, 2013.

[88] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM'10*, 2010.

[89] Z. Feng, G. Wen, C. Yin, and H. Liu, "Video stream groupcast optimization in WLAN," in *IEEE ITA'10*, 2010.

[90] A. Fréville, "The multidimensional 0–1 knapsack problem: An overview," *Eur. J. Oper. Res.*, vol. 155, no. 1, pp. 1–21, 2004.

[91] L. Georgiadis, M. J. Neely, L. Tassiulas *et al.*, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[92] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[93] P. Giaccone, E. Leonardi, and F. Neri, "On the interaction between TCP-like sources and throughput-efficient scheduling policies," *Performance Evaluation*, vol. 70, no. 4, pp. 251–270, 2013.

[94] F. Gont, "Deprecation of ICMP source quench messages," 2012.

[95] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2008.

[96] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM'09*, 2009.

[97] H. Gudmundsdottir, E. I. sgeirsson, M. H. L. Bodlaender, J. T. Foley, M. M. Halldrsson, and Y. Vigfusson, "Wireless scheduling algorithms in complex environments," in *ACM MSWiM'14*, 2014.

[98] S. K. S. Gupta, V. Shankar, and S. Lalwani, "Reliable multicast MAC protocol for wireless LANs," in *IEEE ICC'03*, 2003.

[99] V. Gupta, C. Gutterman, Y. Bejerano, and G. Zussman, "Experimental evaluation of large scale WiFi multicast rate control," in *Proc. IEEE INFOCOM'16*, 2016.

[100] ——, "Experimental evaluation of large scale WiFi multicast rate control," in *arXiv:1601.06425 [cs.NI]*, 2016.

[101] V. Gupta, Y. Bejerano, J. Ferragut, K. Guo, C. Gutterman, T. Nandagopal, and G. Zussman, "Light-weight feedback mechanism for WiFi multicast to very large groups experimental evaluation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3826–3840, 2016.

[102] V. Gupta, R. Norwitz, S. Petridis, C. Gutterman, Y. Bejerano, and G. Zussman, "Demo: WiFi multicast to very large groups experimentation on the ORBIT testbed," 2015.

[103] ——, "Demo: AMuSe: Large-scale WiFi video distribution experimentation on the ORBIT testbed," in *Proc. IEEE INFOCOM'16*, 2016.

[104] V. Gupta, H. Pasandi, X. Lianghua, and G. Zussman, "Demo video for mobility scenario," 2017. [Online]. Available: https://goo.gl/MiOZvv

[105] ——, "QoE optimization for video streaming over wireless multicast (technical report)," 2017. [Online]. Available: goo.gl/LQDwYY

[106] V. Gupta, L. Xu, B. Wu, C. Gutterman, Y. Bejerano, and G. Zussman, "Demo: Evaluating video delivery over wireless multicast," in *Proc. IEEE INFOCOM'17*, 2017.

[107] N. Hajlaoui and I. Jabri, "On the performance of IEEE 802.11n protocol," in *ACM WiNTECH'12*, 2012.

[108] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," in *ACM SIGCOMM'10*, 2010.

[109] ——, "Predictable 802.11 packet delivery from wireless channel measurements," in *Proc. ACM SIGCOMM'10*, 2010.

[110] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *ACM MOBICOM'01*, 2001.

[111] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM'13*, 2013.

[112] H. Hu, X. Zhu, Y. Wang, R. Pan, J. Zhu, and F. Bonomi, "Proxy-based multi-stream scalable video adaptation over wireless networks using subjective quality and rate models," *IEEE Trans. Multimedia*, vol. 15, no. 7, pp. 1638–1652, 2013.

[113] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. ACM SIGCOMM'15*, 2015.

[114] R. Huysegems, J. van der Hooft, T. Bostoen, P. Rondao Alface, S. Petrangeli, T. Wauters, and F. De Turck, "HTTP/2-based methods to improve the live experience of adaptive streaming," in *Proc. ACM MM'15*, 2015.

[115] "IEEE draft standard for information technology telecommunications and information exchange between systems local and metropolitan area networks - specific requirements, part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications - amendment: MAC enhancements for robust audio video streaming," IEEE, July 2011.

[116] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM'13*, 2013.

[117] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," in *Proc. ACM CoNEXT'12*, 2012.

[118] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking," in *Proc. ACM SIGCOMM'09*, 2009.

[119] G. Judd, X. Wang, and P. Steenkiste, "Efficient channel-aware rate adaptation in dynamic environments," in *ACM MobiSys'08*, 2008.

[120] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar, "Af-qcn: Approximate fairness with quantized congestion notification for multi-tenanted data centers," in *Proc. IEEE HOTI'10*, 2010.

[121] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks," in *Proc. ACM CoNEXT'14*, 2014.

[122] A. Kamerman and L. Montebani, "WaveLAN-ii: a high-performance wireless LAN for the unlicensed band," *Bell Labs technical journal*, vol. 2, no. 3, p. 118133, 1997.

[123] S. Kaul, M. Gruteser, and I. Seskar, "Creating wireless multi-hop topologies on space-constrained indoor testbeds through noise injection," in *IEEE TRIDENTCOM'06*, 2006.

[124] A. Kaya, D. Calin, and H. Viswanathan, "On the performance of stadium high density carrier Wi-Fi enabled LTE small cell deployments," in *Proc. IEEE WCNC'15*, 2015.

[125] G. Keiser, "FTTX concepts and applications," *John Wiley and Sons*, 2006.

[126] A. A. Khalek, C. Caramanis, and R. W. Heath, "A cross-layer design for perceptual optimization of H. 264/SVC with unequal error protection," *IEEE J. Sel. Areas in Commun.*, vol. 30, no. 7, pp. 1157–1171, 2012.

[127] D. Kilper, K. Bergman, V. W. Chan, I. Monga, G. Porter, and K. Rauschenbach, "Optical networks come of age," *Opt. Photon. News*, vol. 25, no. 9, pp. 50–57, 2014.

[128] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: collision-aware rate adaptation for IEEE 802.11 WLANs," in *IEEE INFOCOM'06*, 2006.

[129] J. Kim, C. J. Nuzman, B. Kumar, D. F. Lieuwen, J. S. Kraus, A. Weiss, C. P. Lichtenwalner, A. R. Papazian, R. E. Frahm, and J. V. Gates, "Training 1100 x 1100-port MEMS-based optical crossconnect switches," in *Proc. OSA CLEO'04*, 2004.

[130] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *ArXiv e-prints*, June 2014.

[131] J. K. Kuri and S. Kumar, "Reliable multicast in multi-access wireless LANs," *ACM/Kluwer Wirel. Netw.*, vol. 7, pp. 359–369, 2001.

[132] M. Lacage, M. Manshaei, and T. Turletti, "IEEE 802.11 rate adaptation: a practical approach," in *ACM MSWiM'04*, 2004.

[133] C. Lai, D. Brunina, B. Buckley, C. Ware, W. Zhang, A. Garg, B. Jalali, and K. Bergman, "First demonstration of a cross-layer enabled network node," *IEEE J. Lightw. Technol.*, vol. 31, no. 9, pp. 1512–1525, May 2013.

[134] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, May 1998.

[135] D. Lecompte and F. Gabin, "Evolved multimedia broadcast/multicast service (eMBMS) in LTE-advanced: overview and rel-11 enhancements," *IEEE Comm. Mag.*, vol. 50, no. 11, pp. 68–74, 2012.

[136] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Trans. Comput.*, vol. 100, no. 10, pp. 892–901, 1985.

[137] R. Levy, "The complexity of reliable distributed storage," *PhD Thesis, EPFL*, 2008.

[138] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "ESM: efficient and scalable data center multicast routing," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 944–955, 2012.

[139] P. Li, H. Zhang, B. Zhao, and S. Rangarajan, "Scalable video multicast with joint layer resource allocation in broadband wireless networks," in *Proc. IEEE ICNP'10*, 2010.

[140] X. Li and M. J. Freedman, "Scaling IP multicast on datacenter topologies." in *Proc. ACM CoNEXT'13*, 2013.

[141] Y. Li and L. Tong, "Mach-Zehnder interferometers assembled with optical microfibers or nanofibers," *Opt. Lett.*, vol. 33, no. 4, pp. 303–305, Feb. 2008.

[142] Z. Li and T. Herfet, "HLBP: a hybrid leader based protocol for MAC layer multicast error control in wireless LANs," in *IEEE GLOBECOM'08*, 2008.

[143] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE J. Sel. Areas in Commun.*, vol. 32, no. 4, pp. 719–733, 2014.

[144] W.-S. Lim, D.-W. Kim, and Y.-J. Suh, "Design of efficient multicast protocol for IEEE 802.11n WLANs and cross-layer optimization for scalable video streaming," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 780 –792, 2012.

[145] K. Lin, W. Shen, C. Hsu, and C. Chou, "Quality-differentiated video multicast in multi-rate wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 1, pp. 21–34, January 2013.

[146] Y. Luo, X. Zhou, F. Effenberger, X. Yan, G. Peng, Y. Qian, and Y. Ma, "Time- and Wavelength-Division Multiplexed Passive Optical Network (TWDM-PON) for Next-Generation PON Stage 2 (NG-PON2)," *IEEE J. Lightw. Technol*, vol. 31, no. 4, pp. 587–593, Feb. 2013.

[147] X. Ma and G.-S. Kuo, "Optical switching technology comparison: optical MEMS vs. other technologies," *IEEE Commun. Mag.*, vol. 41, no. 11, pp. 16–23, Nov. 2003.

[148] W. Mach and E. Schikuta, "Parallel database join operations in heterogeneous grids," in *Proc. PDCAT'07*, 2007.

[149] P. Marandi, M. Primi, N. Schiper, and F. Pedone, "Ring paxos: A high-throughput atomic broadcast protocol," in *Proc. IEEE/IFIP DSN'10*, 2010.

[150] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz, "Simple heuristics for unit disk graphs," *Networks*, vol. 25, pp. 59–68, 1995. [Online]. Available: citeseer.nj.nec.com/marathe95simple.html

[151] H. McBride and H. Liu, "Multicast in the data center overview," *IETF, Internet Draft*, 2012.

[152] M. McBride and C. Perkins, "Multicast WiFi problem statement," Working Draft, IETF Internet-Draft, 2015, http://www.ietf.org/internet-drafts/draft-mcbride-mboned-wifi-mcast-problem-statement-00.txt.

[153] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[154] L. Militano, D. Niyato, M. Condoluci, G. Araniti, A. Iera, and G. M. Bisci, "Radio resource management for group-oriented services in LTE-A," *IEEE Trans. Veh. Technol.*, vol. 64, no. 8, pp. 3725–3739, 2015.

[155] P. Mirowski, H. Steck, P. Whiting, R. Palaniappan, M. MacDonald, and T. K. Ho, "KL-divergence kernel regression for non gaussian fingerprint based localization," in *IPIN'11*, 2011.

[156] R. Mittal, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proc. ACM SIGCOMM'15*, 2015.

[157] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "QDASH: a QoE-aware DASH system," in *Proc. ACM MMSys'12*, 2012.

[158] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[159] A. B. Owen, *Monte Carlo theory, methods and examples*, 2013.

[160] Q. Pang, V. Leung, and S. Liew, "A rate adaptation algorithm for IEEE 802.11 WLANs based on MAC-layer loss differentiation," in *IEEE BroadNets'06*, 2006.

[161] K. Papagiannaki, M. Yarvis, and W. S. Conner, "Experimental characterization of home wireless networks and design implications," in *IEEE INFOCOM'06*, 2006.

[162] E. Park, S. Han, H. Kim, K. Son, and L. Jing, "Efficient multicast video streaming for IPTV service over WLAN using CC-FEC," in *IEEE ICICSE'08*, 2008.

[163] Y. Park, C. Jo, S. Yun, and H. Kim, "Multi-room IPTV delivery through pseudo-broadcast over IEEE 802.11 links," in *IEEE VTC'10*, 2010.

[164] K. Pelechrinis, T. Salonidis, H. Lundgren, and N. Vaidya, "Experimental characterization of 802.11n link quality at high rates," in *ACM WiNTECH'10*, 2010.

[165] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," in *Proc. ACM SIGCOMM'14*, 2014.

[166] B. Quinn and K. Almeroth, "IP multicast applications: Challenges and solutions," *IETF, Internet Draft*, 2001.

[167] S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan, "TCP fast open," in *Proc. ACM CoNEXT'11*, 2011.

[168] S. Radhakrishnan, Y. Geng, V. Jeyakumar, A. Kabbani, G. Porter, and A. Vahdat, "SENIC: Scalable NIC for end-host rate limiting." in *Proc. USENIX NSDI'14*, 2014.

[169] B. Radunovic, A. Proutiere, D. Gunawardena, and P. Key, "Dynamic channel, rate selection and scheduling for white spaces," in *ACM CONEXT'11*, 2011.

[170] H. Rahul, F. Edalat, D. Katabi, and C. Sodinii, "Frequency-aware rate adaptation and MAC protocols," in *ACM MOBICOM'09*, 2009.

[171] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM'11*, 2011.

[172] T. S. Rappaport, *Wireless Communication Principle and Practice, 2nd edition.* Prentice Hall, 2002.

[173] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee, "Diagnosing wireless packet losses in 802.11: Separating collision from weak signal," in *IEEE INFOCOM'08*, 2008.

[174] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-based models of delivery and interference in static wireless networks," in *Proc. ACM SIGCOMM'06*, 2006.

[175] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the levy-walk nature of human mobility," *IEEE Trans. Netw.*, vol. 19, no. 3, pp. 630–643, 2011.

[176] G. Rouskas, "Optical layer multicast: rationale, building blocks, and challenges," *IEEE Network*, vol. 1, no. 17, pp. 60–65, 2013.

[177] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM SIGCOMM'15*, 2015.

[178] L. H. Sahasrabuddhe and B. Mukherjee, "Light trees: optical multicasting for improved performance in wavelength routed networks," *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 67–73, 1999.

[179] P. Salvador, L. Cominardi, F. Gringoli, and P. Serrano, "A first implementation and evaluation of the IEEE 802.11 aa group addressed transmission service," *ACM SIGCOMM Comp. Comm. Rev.*, vol. 44, no. 1, pp. 35–41, 2013.

[180] P. Samadi, J. Xu, and K. Bergman, "Virtual machine migration over optical circuit switching network in a converged inter/intra data center architecture," in *Proc. OSA OFC'15*, 2015.

[181] P. Samadi, V. Gupta, B. Birand, H. Wang, G. Zussman, and K. Bergman, "Poster: Accelerating incast and multicast traffic delivery for data-intensive applications using physical layer optics," in *Proc. SIGCOMM'14*, 2014.

[182] ——, "Software-addressable optical accelerators for data-intensive applications in cluster-computing platforms," in *Proc. ECOC'14*, 2014.

[183] P. Samadi, V. Gupta, J. Xu, H. Wang, G. Zussman, and K. Bergman, "Optical multicast system for data center networks," *Optics Express*, vol. 23, no. 17, pp. 22 162–22 180, 2015.

[184] N. Sambo, G. Meloni, G. Berrettini, F. Paolucci, A. Malacarne, A. Bogoni, F. Cugini, L. Poti, and P. Castoldi, "Demonstration of data and control plane for optical multicast at 100 and 200 gb/s with and without frequency conversion," *IEEE J. Opt. Commun. Netw.*, vol. 5, no. 7, pp. 667–676, 2013.

[185] S. Sanfilippo and P. Noordhuis, "Redis," http://redis.io.

[186] S. Scellato, I. Leontiadis, C. Mascolo, P. Basu, and M. Zafer, "Evaluating temporal robustness of mobile networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 1, pp. 105–117, 2013.

[187] S. Sen, N. K. Madabhushi, and S. Banerjee, "Scalable WiFi media delivery through adaptive broadcasts," in *Proc. USENIX NSDI'10*, 2010.

[188] Y. Seok and Y. Choi, "Efficient multicast supporting in multi-rate wireless local area networks," in *IEEE ICOIN'03*, 2003.

[189] V. Sgardoni, M. Sarafianou, P. Ferre, A. Nix, and D. Bull, "Robust video broadcasting over 802.11a/g in time-correlated fading channels," *IEEE Trans. Consum. Electron.*, vol. 55, no. 1, pp. 69–76, 2009.

[190] M. Sharif and A. Kabbani, "Flicr: Flow-level congestion-aware routing for direct-connect data centers," in *Proc. INFOCOM'17*, 2017.

[191] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts, Sixth Edition*. McGraw-Hill, 2010.

[192] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," in *Proc. ACM SIGCOMM'15*, 2015.

[193] R. Sivaraj, A. Pande, and P. Mohapatra, "Spectrum-aware radio resource management for scalable video multicast in LTE-advanced systems," in *Proc. IFIP Networking'13*, 2013.

[194] M. R. Souryal, L. Klein-Berndt, L. E. Miller, and N. Moayeri, "Link assessment in an indoor 802.11 network," in *IEEE WCNC'06*, 2006.

[195] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: near-optimal bitrate adaptation for online videos," in *Proc. IEEE INCOFOM'16*, 2016.

[196] V. Srinivas and L. Ruan, "An efficient reliable multicast protocol for 802.11-based wireless LANs," in *IEEE WoWMoM'09*, 2009.

[197] B. Stephens, A. L. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter, "Practical DCB for improved data center networks," in *Proc. IEEE INFOCOM'14*, 2014.

[198] A. Sugita, A. Kaneko, and M. Itoh, "Planar lightwave circuit," *US Patent 6304706*, Oct. 2001.

[199] M.-T. Sun, L. Huang, A. Arora, and T.-H. Lai, "Reliable MAC layer multicast in IEEE 802.11 wireless networks," in *IEEE ICPP'02*, 2002.

[200] H. Takahashi, S. Suzuki, K. Kato, and I. Nishi, "Arrayed-waveguide grating for wavelength division multi/demultiplexer with nanometre resolution," *IEEE Electron. Lett.*, vol. 26, no. 2, pp. 87–88, 1990.

[201] Y. Tanigawa, K. Yasukawa, and K. Yamaoka, "Transparent unicast translation to improve quality of multicast over wireless LAN," in *IEEE CCNC'10*, 2010.

[202] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic http streaming," in *ACM CONEXT'12*, 2012.

[203] ——, "Towards agile and smooth video adaptation in dynamic http streaming," in *Proc. ACM CoNEXT'12*, 2012.

[204] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," in *Proc. ACM SIGCOMM'12*, 2012.

[205] A. W. Van der Vaart, *Asymptotic statistics*. Cambridge university press, 2000.

[206] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained TCP retransmissions for datacenter communication," in *Proc. ACM SIGCOMM'09*, 2009.

[207] J. Vella and S. Zammit, "A survey of multicasting over wireless access networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 718–753, 2013.

[208] J. Villalon, P. Cuenca, L. Orozco-Barbosa, Y. Seok, and T. Turletti, "Cross-layer architecture for adaptive video multicast streaming over multirate wireless LANs," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 699 –711, 2007.

[209] A. Vlavianos, L. Law, I. Broustis, S. Krishnamurthy, and M. Faloutsos, "Assessing link quality in ieee 802.11 wireless networks: Which is the right metric?" in *IEEE PIMRC'08*, 2008.

[210] K. N. D. Vukobratovic, "A survey on application layer forward error correction codes for IP datacasting in DVB-H," in *3rd COST 2100 MCM*, 2007.

[211] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-layer wireless bit rate adaptation," in *ACM SIGCOMM'09*, 2009.

[212] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *Proc. ACM SIGCOMM'10*, 2010.

[213] H. Wang, Y. Xia, K. Bergman, T. E. Ng, S. Sahu, and K. Sripanidkulchai, "Rethinking the physical layer of data center networks of the next decade: Using optics to enable efficient *-cast connectivity," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 52–58, 2013.

[214] X. Wang, L. Wang, Y. Wang, and D. Gu, "Reliable multicast mechanism in WLAN with extended implicit MAC acknowledgment," in *IEEE VTC'08*, 2008.

[215] X. Wang, L. Wang, Y. Wang, Y. Zhang, and A. Yamada, "Supporting MAC layer multicast in IEEE 802.11n: Issues and solutions," in *IEEE WCNC'09*, 2009.

[216] X. Wang, J. Chen, A. Dutta, and M. Chiang, "Adaptive video streaming over whitespace: SVC for 3-tiered spectrum sharing," in *IEEE INFOCOM'15*, 2015.

[217] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. USENIX OSDI'06*, 2006.

[218] T. White, "Hadoop: the definitive guide," *O'Reilly Media Inc.*, 2009.

[219] S. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *ACM MOBICOM'06*, 2006.

[220] F. Wu, Y. Yang, O. Zhang, K. Srinivasan, and N. B. Shroff, "Anonymous-query based rate control for wireless multicast: Approaching optimality with constant feedback," in *Proc. ACM MOBIHOC '16*, 2016.

[221] M. Wu, S. Makharia, H. Liu, D. Li, and S. Mathur, "IPTV multicast over wireless LAN using merged hybrid ARQ with staggered adaptive FEC," *IEEE Trans. Broadcast.*, vol. 55, no. 2, pp. 363 –374, 2009.

[222] S. Xiang, L. Cai, and J. Pan, "Adaptive scalable video streaming in wireless networks," in *Proc. ACM MMSys'12*, 2012.

[223] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "Pistream: Physical layer informed adaptive video streaming over lte," in *Proc. ACM MobiCom'15*, 2015.

[224] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proc. IEEE INFOCOM'04*, 2004.

[225] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM CoNEXT'15*. ACM, 2015.

[226] J. Yoon, H. Zhang, S. Banerjee, and S. Rangarajan, "MuVi: A multicast video delivery scheme for 4G cellular networks," in *Proc. ACM MobiCom'12*, 2012.

[227] A. H. Zahran, J. Quinlan, D. Raca, C. J. Sreenan, E. Halepovic, R. K. Sinha, R. Jana, and V. Gopalakrishnan, "Oscar: an optimized stall-cautious adaptive bitrate streaming algorithm for mobile networks," in *Proc. ACM MoVID'16*, 2016.

[228] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: reducing the flow completion time tail in datacenter networks," in *Proc. ACM SIGCOMM'12*, 2012.

[229] D. Zats, A. P. Iyer, G. Ananthanarayanan, R. Agarwal, R. Katz, I. Stoica, and A. Vahdat, "FastLane: making short flows shorter with agile drop notification," in *Proc. ACM SoCC'15*, 2015.

[230] Y. Zhang and N. Ansari, "On mitigating TCP incast in data center networks," in *Proc. IEEE INFOCOM'11*, 2011.

[231] K. Zheng, F. Liu, L. Lei, C. Lin, and Y. Jiang, "Stochastic performance analysis of a wireless finite-state markov channel," *IEEE Trans. on Wireless Commun.*, vol. 12, no. 2, pp. 782–793, 2013.

[232] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," in *Proc. ACM SIGCOMM'15*, 2015.

[233] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "Ecn or delay: Lessons learnt from analysis of DCQCN and TIMELY," in *Proc. ACM CoNEXT'16*, 2016.

[234] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha, "Can accurate predictions improve video streaming in cellular networks?" in *Proc. ACM HotMobile'15*, 2015.