Downstream Bandwidth Management for Emerging DOCSIS-based Networks

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Science

---

by
Gongbing Hong
December 2015

Accepted by:
Dr. James Martin, Committee Chair
Dr. Brian Dean
Dr. Feng Luo
Dr. James Westall

Abstract

In this dissertation, we consider the downstream bandwidth management in the context of emerging DOCSIS-based cable networks. The latest DOCSIS 3.1 standard for cable access networks represents a significant change to cable networks. For downstream, the current 6 MHz channel size is replaced by a much larger 192 MHz channel which potentially can provide data rates up to 10 Gbps. Further, the current standard requires equipment to support a relatively new form of active queue management (AQM) referred to as delay-based AQM. Given that more than 50 million households (and climbing) use cable for Internet access, a clear understanding of the impacts of bandwidth management strategies used in these emerging networks is crucial. Further, given the scope of the change provided by emerging cable systems, now is the time to develop and introduce innovative new methods for managing bandwidth.

With this motivation, we address research questions pertaining to next generation of cable access networks. The cable industry has had to deal with the problem of a small number of subscribers who utilize the majority of network resources. This problem will grow as access rates increase to gigabits per second. Fundamentally this is a problem on how to manage data flows in a fair manner and provide protection. A well known performance issue in the Internet, referred to as bufferbloat, has received significant attention recently. High throughput network flows need sufficiently large buffer to keep the pipe full and absorb occasional burstiness. Standard practice however has led to equipment offering very large unmanaged buffers that can result in sustained queue levels increasing packet latency. One reason why these problems continue to plague cable access networks is the desire for low complexity and easily explainable (to access network subscribers and to the Federal Communications Commission) bandwidth management.

This research begins by evaluating modern delay-based AQM algorithms in downstream DOCSIS 3.0 environments with a focus on fairness and application performance capabilities of single queue AQMs. We are especially interested in delay-based AQM schemes that have been proposed to combat the bufferbloat problem. Our evaluation involves a variety of scenarios that include tiered services and application workloads. Based on our results, we show that in scenarios involving realistic workloads, modern delay-based AQMs can effectively mitigate bufferbloat. However they do not address the other problem related to managing the fairness.

To address the combined problem of fairness and bufferbloat, we propose a novel approach to bandwidth management that provides a compromise among the conflicting requirements. We introduce a flow quantization method referred to as adaptive bandwidth binning where flows that are observed to consume similar levels of bandwidth are grouped together with the system managed through a hierarchical scheduler designed to approximate weighted fairness while addressing bufferbloat. Based on a simulation study that considers many system experimental parameters including workloads and network configurations, we provide evidence of the efficacy of the idea. Our results suggest that the scheme is able to provide long term fairness and low delay with a performance close to that of a reference approach based on fair queueing. A further contribution is our idea for replacing 'tiered' levels of service based on service rates with tiering based on weights. The application of our bandwidth binning scheme offers a timely and innovative alternative to broadband service that leverages the potential offered by emerging DOCSIS-based cable systems.

Dedication

This dissertation is dedicated to my beloved wife for your unconditional love, encouragement, unwavering support and endless sacrifice, and to my two wonderful boys for all you have to suffer and endure for the last several years when dad had to be absent from various activities of yours. You are my motivation to the life yet to come. This dissertation is also dedicated to our extended families on both sides of mine and my wife's. Your love and support have helped me to reach this milestone in my life.

To my earthly late father, now in heaven, and my mother, still thriving in her late eighties and surprising one of my sons with her ability of climbing a fruit tree to get him fresh fruit (you should have let your grandson do that himself, mom!), may this work of your son's continue to make both of you proud. Although you are illiterate living in one of the most remote villages in China, you have the vision and have sacrificed so much to send one of your sons to the farthest place on earth in study and life.

Acknowledgments

First and foremost I would like to thank my advisor, Dr. Jim Martin, for his guidance, motivation, and vision throughout my studies. Without his direction the work presented in this dissertation is simply impossible. From a more personal side, I also like to thank him for his extraordinary patience. He has served as a role model for me.

I would also like to thank Dr. Brian Dean and Dr. Feng Luo for serving on my committee. I was fortunate to have the opportunity to take Dr. Dean's *Advanced Data Structures* course. It was a very hard course but at the same time was also an eye-opener for me. It taught me to look at a problem sometimes from a very unusual perspective. I had to spend tremendous amount of time on this course. At the end it turned out to be a great course absolutely worth taking. On top of that, Dr. Dean is such an inspiring teacher for us to have and enjoy.

I would especially like to thank Dr. Mike Westall, also a member of the committee. I have had the privilege to work with him closely. I simply cannot thank him enough for his willingness to share his knowledge, experience, and ideas. He always amazed me with those of his ideas. There was one time I was made speechless after he suggested a much simpler and better alternative approach to a problem I was awkwardly addressing. Simply put, everything he touches, it immediately shines.

Lastly I would like to thank Dr. Mark Smotherman, who used to be the Director of Graduate Study when I got enrolled into the PhD program. He received me with great warmth when I was invited for a visit before I came to Clemson.

Table of Contents

List of Tables

List of Figures

List of Listings

Chapter 1

Introduction

Cable access network is one of the dominant broadband access network technologies. The Data Over Cable Service Interface Specification (DOCSIS) defines the operations of cable access broadband networks. Such networks traditionally use standard cable TV channels for data transfers between home / small business users and the Internet.

The evolution of cable network technology is at an intriguing crossroad. Traditional broadcast video is converging with Internet video broadcast. Multiple system operators (MSOs) must engineer their access networks to competitively support both traditional video broadcast service and broadband Internet access. This task is challenging because of the rapid evolution of technology and customer demands from both worlds. In the video broadcast domain, system operators must provide access to a mix of hundreds of standard and high definition television channels along with video-on-demand services. In the broadband access domain, standards are rapidly evolving to provide ever increasing data rates to end users.

Figure 1.1 illustrates how a DOCSIS-based cable network bridges between a home network and the Internet. In the middle of the figure DOCSIS network uses either a coaxial cable or a hybrid fiber-coaxial cable (HFC) to provide a two-way communication that at the end provides a transparent bi-directional transfer of Internet Protocol (IP) traffic.

On the left side of the figure, a home or small office network is connected to the cable network through a user device called Cable Modem (CM). CMs are connected to a head-end

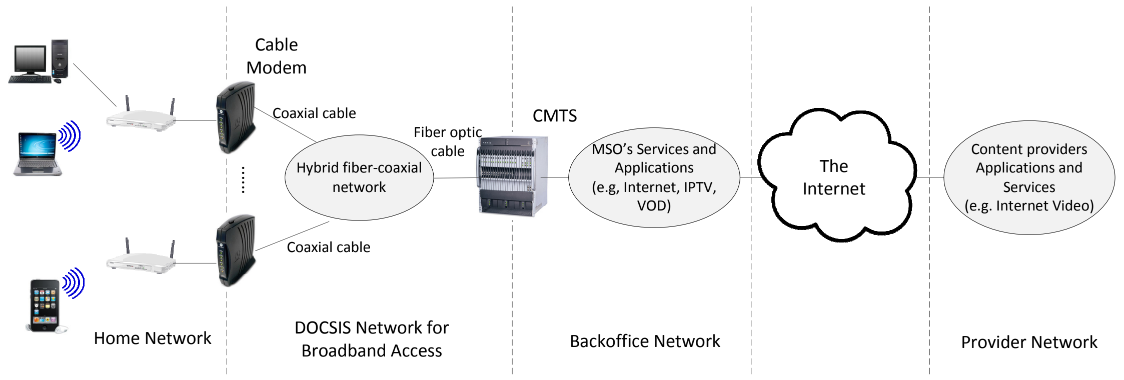

Cable Modem
Coaxial cable
Fiber optic cable
CMTS
Hybrid fiber-coaxial network
MSO's Services and Applications (e.g, Internet, IPTV, VOD)
The Internet
Content providers Applications and Services (e.g. Internet Video)
Coaxial cable
Home Network
DOCSIS Network for Broadband Access
Backoffice Network
Provider Network

Figure 1.1: DOCSIS System Architecture

device, called Cable Modem Termination System (CMTS), which control the access of the CMs to the DOCSIS-based cable network. The DOCSIS MAC protocol is used to establish the connections between the CMTS and the CMs. The CMTS is then connected to the Internet through a backoffice network of a cable operator. The backoffice network is the standard TCP/IP based network.

1.1 Research Motivation

In this research we consider the downstream bandwidth management problem that arises with emerging DOCSIS-based access networks. While bandwidth management is an established research area, the context of the issue is rapidly changing.

Broadband access networks based on the latest DOCSIS 3.1 (D3.1) are enabling Gbps bandwidth to our homes. Broadband network operators however have never managed such high bandwidth broadband networks in the past. Existing management methods are showing their limits. Given the new context, it is vital to study and develop new ways for bandwidth management to ensure successful deployments of the emerging high bandwidth broadband networks.

This work is motivated by several recent developments in both research and network communities. First, cable networks, the dominant broadband access technology in North America, are transitioning to the D3.1 standard [9]. By replacing the standard 6 MHz channel with channels up to 192 MHz, the number of households with Gbps access in the United States will increase dramatically. This can potentially lead to a significant increase in average service rates available to subscribers. The Federal Communications Commission (FCC) has recently redefined broadband service as one with a minimum speed of 25 Mbps downstream and 3 Mbps upstream respectively [25]. With Gbps channel speed, DOCSIS cable networks will be well positioned to support the new broadband service definition.

Second, current network engineering practices involving the use of network devices that utilize large unmanaged network buffers have been identified as problematic. This issue, generally referred to as bufferbloat, has sparked renewed interest in active queue management (AQM) [34]. Bufferbloat results in persistently high queue levels leading to large and sustained packet latency. Large packet latency has significant negative impact over many interactive and delay sensitive applications that now proliferate in the Internet.

The network community has identified the importance of reducing queueing delay in the latest RFC 7567 [4] recommending the use of AQM. The idea of delay-based AQM has been recently proposed. AQM algorithms such as CoDel [73, 72] and PIE [77, 75] implement a pro-active loss process designed to maintain a statistical packet latency target. The latest DOCSIS cable standards reflect this direction. D3.1 requires cable modems to support PIE AQM to manage the upstream queue. For downstream, the cable modem termination system (CMTS) is required to use a published AQM algorithm by default (although it can be disabled if so chosen by network providers). As the deployment of D3.1 is likely to occur over several years, the current DOCSIS 3.0 (D3.0) standard has been revised to recommend the immediate use of delay-based AQM. Although AQM has been widely studied and deployed (e.g., the Random Early Detection AQM algorithm is available in nearly every modern networking device), it has not been widely used. The AQM requirements for DOCSIS will likely lead to the first large scale use of AQM to enhance the performance of the network.

1.2 Research Direction

The research presented in this dissertation focuses on DOCSIS-based cable networks providing Internet access. Current practice involves the use of large unmanaged buffers with simple first-come-first-served (FCFS) scheduling. Network operators use pricing-based control models to provide higher end service in terms of service tiers, which allow network operators to limit the subscribers to a maximal usage and to better engineer their networks. The issues related to managing subscriber traffic based on consumption have a long contentious history. A significant backdrop for our research is the observation that D3.1 can potentially make significantly higher service rates available to subscribers.

With the D3.1 deployment imminent, we explore and evaluate a new bandwidth management scheme for emerging DOCSIS cable networks. We believe the focus of the bandwidth management in a network is fundamentally about achieving fairness and low queueing delay. To provide focus, we limit our study to downstream cable scenarios.

The research is divided into two phases. The first phase focuses on evaluating the effectiveness of modern delay-based AQM schemes to manage fairness and application performance in realistic, single or bonded channel downstream DOCSIS 3.0 cable network scenarios. We consider scenarios that involve different service tiers, where different cable users are provisioned with different service rates.

The second phase of our research develops and evaluates a new innovative bandwidth management approach. We introduce our idea first by identifying our original design goals:

1. The scheme should provide predictable latency property consistent with that of modern delay-based AQMs such as CoDel to address the bufferbloat problem

2. The scheme should provide predictable service levels consistent with service tiers purchased by subscribers. One reasonable goal is to approximate the standard practice of weighted max-min fair allocation [46].

3. The scheme must reflect a conscious compromise between fairness and algorithmic

complexity over a broad range of network scenarios and diverse workloads.

4. The scheme should exhibit high system efficiency in terms of bandwidth utilization.

5. The scheme should require minimal configuration parameter settings. The community has learned that complicated tuning required by any scheme hinders the acceptance of such scheme.

We introduce a novel packet scheduling method we refer to as adaptive bandwidth binning (ABB). ABB maintains a small number of queues, referred to as bins. Flows with similar consumption levels are quantized into a number of flow groups corresponding to the number of bins in use. Packets from the flows in the same group are then aggregated into the same bin. Bins are assigned weights according to the flows they hold. An outer scheduler serves the bins in a weighted deficit round-robin (WDRR) [87] manner. The algorithm adapts to changing workloads by periodically remapping flows to bins when flow consumption levels change. By normalizing recent bandwidth consumption with flow weights, we extend ABB to support *weighted* bandwidth binning which can be used to implement service tiering where tiers are defined by weights.

1.3 Problem Formulation and Contributions

Scheduling and queue management disciplines are fundamental to computer networking and have been studied from many different perspectives. Research includes work in areas related to packet scheduling, buffer management, congestion control, and a wide range of topics surrounding how networks can provide services that meet targeted performance levels. In spite of this body of knowledge, large scale networks such as the Internet still suffer from many known problems including bufferbloat, TCP round trip time (RTT) unfairness, and vulnerability to unresponsive flows [34, 73, 31, 29, 32, 82].

It has been established that TCP suffers from a mutlitude of problems. For example, TCP's end-to-end congestion control mechanism does not sufficiently enforce fairness for a number of common scenarios [60]. Examples include the scenario where long RTT flows competing with short RTT flows and the scenario where flows based on unresponsive UDP competing with flows based on TCP. The Internet was originally designed by assuming a 'dumb' network with 'smart' endpoints. This design likely is the reason why TCP/IP and the Internet have flourished. However, the design strategy has clear shortcomings with respect to performance and reliability. In spite of the enormous amount of research over the last 40 years, there has not been a significant change to the underlying service model provided by the Internet. These issues have led to notable events where service providers felt forced to manage traffic to "protect paying customers from others who don't share so well" [60].

Economic factors have led to the current service tiering model in terms of individually imposed maximum subscriber service rates. With the current service tiering model, network operators identify a discrete set of available subscriber service rates, called *service tiers*, for users to choose from. A network operator then uses *regulators* to limit subscribers achieved service rates from exceeding their prescribed max service rates.

The current standard practice is for all packets, after they are eligible for transmission following the regulator process, to be managed using a single aggregate queue with a simple FCFS scheduler and managed with simplest drop-tail (DT) queue management technique. While simple to implement and inexpensive, the scheme is known to suffer from the fairness issue mentioned earlier. While the use of regulators provides a mechanism that allows service providers to have an economic framework for establishing service tiers, the approach blurs network functions such as traffic shaping and rate limiting with congestion control and traffic management. If a network operator implements a service tiering model based solely on service rates enforced by the regulator process, without further support of more sophisticated packet scheduling such as weighted fair queueing, as we show in our prior work [38], the allocation outcomes will not match the service levels that are alluded to by the service tiers during the time of congestion. Further, we show that the known issues related to bufferbloat lead to unpredictable results. The Internet community is addressing the bufferbloat problem with delay-based AQMs. However our prior work [38] has shown that the use of single queue delay-based AQMs does not help addressing the unfairness issue but instead exacerbates the issue.

We address the combined problem of fairness and bufferbloat in modern DOCSIS-based cable access networks with a low complexity / cost solution. While the majority of CMTS equipment in use in the United States supports advanced scheduling techniques such as weighted fair queueing (WFQ) and most likely a wide range of other approximations of fair queueing (FQ) as well, it appears that such advanced schemes are not widely used. Although we do not know why they are not used, we do know they come at high complexity and/or high cost. For example, the work needed may be more than $O(1)$ per packet, per-flow queueing may be required, dynamic queue creation / deletion, or a large number of queues may be required to provide fairness, etc. The cable community would like to have available other options that represent a compromise between complexity, cost, invasiveness, and robust performance. In addition, it is likely that network operators may move towards a network function virtualization strategy where the small number of large traditional CMTS nodes are replaced by lighter-weight devices that might be deployed on demand.

The contributions of this research are two-fold. First, with large deployment of AQM imminent for DOCSIS-based networks, our evaluation of the modern single-queue delay-based AQMs provides timely feedback to the community concerning how delay-based AQM schemes are able to manage fairness and application performance in realistic downstream cable network scenarios. To the best of our knowledge, the two papers [57, 38] that have been published based on this work are the first that consider the impact of queue management on the dominant Internet applications such as Internet streaming based on HTTP-based Adaptive Streaming (HAS). Second, while the idea of using two queues to separately carry high bandwidth application flows (referred to as 'elephants') and much larger number of low bandwidth flows (referred to as 'mice') has been proposed, we are the first to evaluate an adaptive bandwidth binning scheme that provides approximate *weighted* fairness with low complexity and cost.

1.4 Dissertation Outline

This dissertation is organized as follows. In Chapter 2, we review background materials on the scheduling and queueing disciplines relevant to this research. We also summarize related work in the area. In Chapter 3, we present our system model and simulation model. We also discuss related metrics used in evaluating various bandwidth management approaches. In Chapter 4, we provide detailed analysis on the effectiveness of recent delay-based single-queue AQM schemes applied to downstream traffic in cable access networks. Our simulation framework utilizes realistic scenarios including FTP, VoIP, web, and HAS. We present a new bandwidth management scheme based on a multi-queue AQM approach in Chapter 5. We provide detailed results and analysis on how effective the scheme is able to approximate weighted fairness and provide low delays in various scenarios. We conclude our work with a summary of results in Chapter 6.

Chapter 2

Background

The background materials pertaining to this research fall into several areas. We first introduce the general concept of bandwidth management. We then review the related research covering packet scheduling and queue management techniques. These are followed by the introduction of DOCSIS. Additional background materials, when relevant, will be included in the chapters that follows.

2.1 Bandwidth Management

Subscriber bandwidth management is one of the core management tasks of the access network and is at the center of much of the change happening in broadband access. The fairness model inherent in the Internet, which is fairness based on 'TCP fairness' [29], is arguably outdated as it was based on assumptions that are decades old. It is well known that the allocation of Internet resources depends on the end-to-end round trip time as well as host TCP configuration and implementation. Applications are free to utilize parallel TCP connections to compete for more bandwidth. For these reasons, along with the economics surrounding broadband access, the access networks such as DOCSIS cable networks have historically used supplemental bandwidth management procedures. One such example is Comcast's protocol-agnostic congestion management system [64].

An attribute of bandwidth management is the granularity of information used as part of the management feedback loop. The information can be extremely granular (e.g., based on type of application) or based on limited information (e.g., based on bandwidth consumed). The issue is a touchy subject with government policy makers (who need to ensure that network operators do not implement monopolistic practices) and with the public (who will likely become enraged if they find out that certain applications are purposely treated unfairly). While there is no consensus on exactly how shared resources should be managed, there is agreement that bandwidth management must be done.

There are several broad dimensions to bandwidth management including service models, scope, and locations. An additional dimension of concern to this research is the time scale of control. Bandwidth management can operate at different time scales. The range of mechanisms for this dimension includes the following:

Microseconds Packet scheduling disciplines determine which packets get serviced when a link becomes idle. Packet queue management policies are implemented at this time scale as well.

Milliseconds End-to-end congestion control algorithms such as the ones supported by TCP stacks manage how a flow reacts to signs of network congestion.

Seconds-Minutes-hours Traffic management methods such as routing algorithms modify the allocation of resources based on control procedures that use relatively large time scales.

Days or weeks Admission control and capacity planning methods operate at very large time scale to ensure that the network is adequately provisioned to meet throughput and delay requirements.

A broadband access network likely requires bandwidth management operating at all time scales. Due to the inherent complexity of bandwidth management and due to sometimes unseen underlying interactions between the different levels of bandwidth management, it is often difficult to know which management schemes and their respective configurations are the best choice.

2.2 Review of Related Work

A tremendous amount of research has been published in the area of resource allocation. Seminal results exist for systems that assume connection oriented networks. It has been shown that guaranteed delay bounds can be provided in such networks by packet scheduling combined with rate limiting [96]. So one front focuses on regulating traffic at the source through traffic shaping or rate limiting. Leaky bucket algorithm [91] is one such mechanism that can control the bandwidth allocated to a source while simultaneously regulate the burstiness of the traffic. TCP end-to-end flow control [43] is another mechanism now ubiquitously implemented at the endpoints of a TCP connection. This mechanism regulates the sending rate of a connection via a dynamically sized congestion window. The window sizing is characterized by an exponential increase before a congestion threshold is reached and an additive increase/multiplicative decrease afterwards. Such control results in so called TCP fairness that inspires the requirement of 'TCP friendliness' for implementation of new protocols [29].

In connectionless networks such as packet switched networks that are based on best effort service model, neither traffic regulation nor TCP end-to-end flow control are sufficient to provide latency or throughput guarantees. Bandwidth management through packet scheduling and queueing disciplines has been explored.

At the heart of bandwidth management is how packets should be managed in the queues and scheduled. According to Keshav [46], there exists two orthogonal components to a scheduling discipline. The first component decides which packet gets serviced next at a congested link through a packet scheduling algorithm. The second component deals with how packets are queued through a queue management scheme.

Packet scheduling disciplines are often designed to provide fairness. The widely accepted fairness criterion is *max-min* fairness. An allocation is max-min fair if it is not

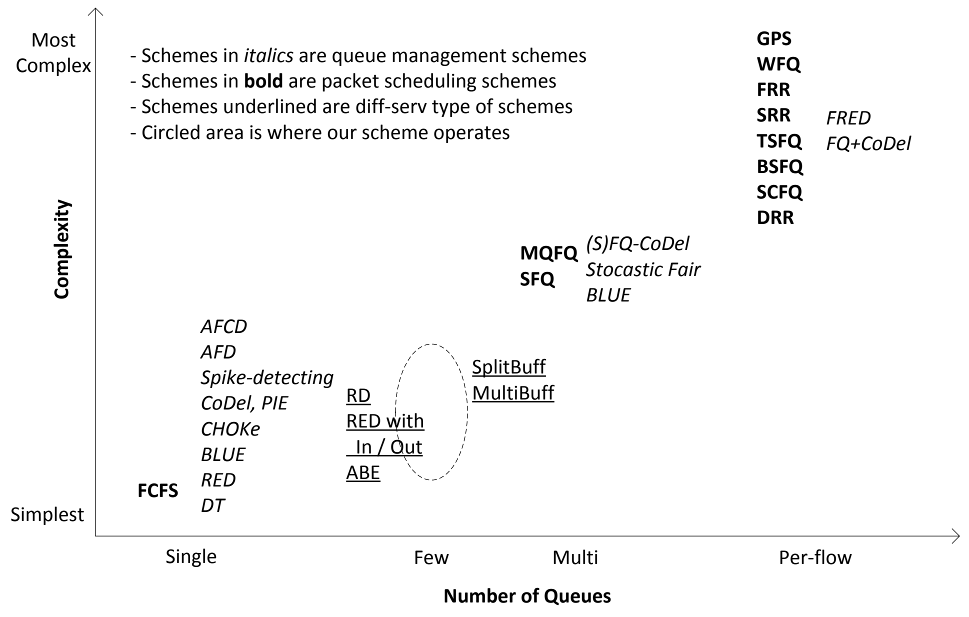


Figure 2.1: Ranges of Complexity of Various Bandwidth Management Methods

possible to increase the rate allocated to any user in the system without decreasing the rate allocated to any other user who is receiving an already lower or equal rate [51]. On the other hand, queue management techniques have been explored to provide low queueing latency for packets. Queueing latency for a packet is given by the duration between the packet entering and leaving a queue.

Among various conflicting objectives of the schemes, tremendous amount of research has explored ways to reduce complexity and cost while maintaining desired properties such as good fairness and low latency in the network. Figure 2.1 provides a number of schemes developed over decades and their relative complexity to each other. The schemes are given in the categories of packet scheduling and queue management. We will discuss several of the schemes depicted in the figure with details later.

In the literature complexity has been looked at from various angles depending on the focus of a particular research. It has been used to describe various complications related to algorithms, data structures, implementation, etc.:

- Algorithmic related complexity. For example, the amount work required per packet

[87], which can range from $O(n)$ to $O(1)$, where n is the number of flows to be scheduled. The number of operations needed per packet [63] must be minimized to support high speed networks.

- Number of queues / bins (aggregate queues) required. Some scheduling schemes require per-flow queueing. For n flows, n queues are required. Queues are dynamically created and destroyed as flows come and go. Such schemes are complex and not scalable. Other schemes require multiple queues [63]. The number of queues (k) required is independent of n. It is expected that $k < n$. These schemes are more scalable and less expensive. In the extreme case, schemes such as FCFS require only a single queue, which makes it simple and inexpensive to implement.

- State information required [78, 76, 54]:
  - Per-flow state information required
  - Per-flow state not required or stateless

In the remainder of the section, we review the related work from the perspectives of packet scheduling and queue management. We include a brief review of research presented in the context of Internet differentiated service (diff-serv) as well.

2.2.1 Packet Scheduling

Packet scheduling deals with selecting which packet among the packets queued at a congested link to be sent next through the link. The simplest packet scheduling algorithm is FCFS. It employs only a single queue for all flows. Incoming packets are queued and transmitted in the order they arrive at the queue. FCFS is very simple and efficient. Thus it is still widely used in the Internet. However, it cannot distinguish packets among different flows. Thus FCFS does not provide any protection or fairness. Instead it rewards flows with faster packet transmitting rates. Given limited queue space, a misbehaving flow can block other flows from accessing the queue for indefinite amount of time.

To address the problems with FCFS, researchers in the past developed many packet scheduling algorithms that maintain individual flow states in one way or another. As originally defined by Nagle [67] and then developed by many others, FQ is a class of packet scheduler algorithms that schedule data packets at congested links to provide *max-min* fairness and protection by maintaining a separate queue for each flow. These algorithms can be broadly divided into two categories: time-stamp based and frame or round-robin (RR) based algorithms.

An ideal fair queueing algorithm is generalized processor sharing (GPS) [79] based on a fluid flow model. It assumes the traffic is infinitely divisible. Thus GPS can serve an infinitesimally small amount of data to each backlogged flow within any finite time interval. GPS therefore can provide ideal flow isolation and fairness. Another similarly ideal fair queueing algorithm considers serving flows in a bit-by-bit round-robin (BR) fashion [17]. While the two scheduling algorithms provide perfect max-min fairness, neither is implementable in a packet-switched network where packets are not infinitely divisible or transmitted bit-by-bit. However, they do provide the performance basis for comparing other more practical approximations of the fair queueing algorithms.

Since a packet must be transmitted in its entirety in a packet-switched network, researchers have developed a number of packet-by-packet versions of the scheduling algorithms that approximate GPS. By emulating GPS in background, weighted fair queueing (WFQ) [17] maintains a virtual time clock and calculates a virtual timestamp or service tag for each packet under GPS. WFQ then selects the packet with the smallest service tag for next transmission opportunity.

While WFQ provides close approximation to GPS, it is computationally complex to emulate GPS behind the scene. The complexity of WFQ is dominated by the virtual timestamp computation for each packet, which requires $O(n)$ time where n is the number of flows. To address the complexity problem with WFQ, Golestani developed a self-clocked fair queueing (SCFQ) scheme [35]. Instead of using the virtual time derived from GPS system for the calculation of service tags for each incoming packet, SCFS uses the service tag of the packet currently receiving service as an estimate of the system virtual time. This significantly simplifies the computation of packet service tags while *still* maintains near optimal performance.

All time-stamp based scheduling algorithms must select a packet of the smallest service tag among the head packets of all flows. This comes at a cost of $O(\log(n))$ work per packet due to service tag sorting, where n is the number of packet flows. This selection process dominates the complexity of SCFQ. The Leap Forward Virtual Clock (LFVC) [88] algorithm further reduces the sorting complexity to $O(\log\log n)$ by coarsening the service tags calculated according to Virtual Clock (VC). This reduced complexity requires the use of a fairly complicated data structure called Van Emde Boas tree. LFVC solves the fairness problem in VC by temporarily moving oversubscribed flows into a low priority holding area. A flow is oversubscribed if the service tag of its head packet exceeds the system clock by a threshold. The threshold is formulated as "throughput condition", which, when met, guarantees throughput bound. The concept is based on parents disciplining their misbehaving children by temporarily removing their privileges until their behaviors improve.

Frame-based approach, on the other hand, divides time into frames and packets are sent within those frames. Frame-based approach avoids the sorting bottleneck of a timestamp based approach and achieves low complexity of $O(1)$. An example of frame based approach is strict round-robin [67]. Each flow uses a queue for incoming packets. Scheduler then polls each flow queue in a cyclic order and serves one packet at a time on any encountered non-empty queues. However this scheme is unfair if packet sizes vary.

Deficit round-robin (DRR) [87] is a scheme that addresses the varying packet size issue by maintaining a deficit count for each flow. At the beginning of each round, active flows are given a quota in unit of bits that is added to the deficit counts. Once a packet from a flow is selected for service, the deficit count of the flow is deducted by the size of the packet in bits. As long as the deficit count is enough for sending out another packet at the head of the queue, packets from the flow will continue to be sent out. Unused portion of the deficit count carries over to next round. Round-robin based schedulers operate at a time scale on the order of a 'round time'. That is the amount of time to serve each flow before returning to the first flow in the set of flows. The actual amount of a 'round time' varies depending on a number of factors such as the number of flows to be served and the link speed.

Timestamp sorting and per-flow queueing requirement can complicate the packet scheduling especially in a system that has to deal with large number of flows. Various scheduling algorithms have been proposed to reduce such complications through quantization. In the literature (e.g., [23]), the term quantization has been used to describe various schemes that classify flows based on certain quantities with or without using bins (or sub-queues) to aggregate packets from different flows belonging to the same flow class. We assume quantization refers to a technique that groups flows based on a particular attribute of the flows and/or the packets. Example attributes include flow weight, packet size, or simply a hash. Binning refers to a technique with which packets from a group of flows are aggregated into a sub-queue called *bin* to be scheduled FCFS. Quantization reduces the complexity related to scheduling such as timestamp sorting while binning helps reduce the number of queues required and simplifies scheduling. Quantization can be combined with binning to reduce the number of queues required as the end result is multiple flows are aggregated into a desired number of bins.

Bin sorting fair queueing (BSFQ) [10] combines the technique of quantization with binning. Quantization is applied in the virtual time space, which is divided into equal intervals (such as $\Delta = 20$) called bins (e.g., $[0, \Delta]$). When a packet arrives, the virtual timestamp for the packet is calculated with the same method as used by SCFQ. Packets with close timestamps that fall into the same bin are queued in the bin in FIFO order and then serviced FCFS to achieve $O(1)$ low complexity. The scheduler starts with the current bin $[t, t + \Delta]$ corresponding to virtual clock time t. Once all packets in current bin are transmitted, the virtual clock increments by Δ and the scheduler moves onto next bin $[t + \Delta, t + 2\Delta]$. So at a macro time scale of Δ, the packets are transmitted approximately according to their timestamps. When Δ is large, BSFQ degrades into FCFS. When Δ

is small, it operates similar to SCFQ, resulting better fairness and delay property. But the amount of state information increases. Finding a proper Δ to use can be a complex compromise among several tradeoffs.

Using quantization, stratified round-robin (SRR) [81] improves DRR over its delay bound property to be independent of the number of flows. Using an exponential grouping scheme, SRR "stratifies" flows into different weight classes (or groups). The scheduling is then organized into two levels: inter-class scheduling and intra-class scheduling. The inter-class scheduling is responsible for scheduling intervals to different flow classes while the intra-class scheduling uses DRR to schedule flows within a class. By combining the ideas of both timestamp based scheduling and round-robin scheduling, fair round-robin (FRR) [95] scheduling uses the same exponential grouping scheme but improves the worst-case fairness property over SRR. It shares the same scheduling structure of SRR by using two level scheduling. The inter-class scheduling is based on timestamps of each flow class. Due to the number of classes being small, the complexity is acceptable even with GPS emulation. Intra-class scheduling uses a modified DRR scheme. As with DRR, both SRR and FRR require per-flow queuing.

Tiered service fair queueing (TSFQ) [23] is another quantization based scheme. TSFQ is timestamp based scheduler with a $O(1)$ time complexity. Arriving packets are assigned timestamps using an efficient virtual time function (similar to what is used by SCFQ). To reduce the timestamp sorting bottleneck, TSFQ first quantizes flows based on flow weights (associated with service tiers) and then within a service tier further quantizes flows based on packet size. The assumption is that traffic is not likely to have arbitrary weights as often serviced at a number of service levels, flows are thus able to be grouped into a limited number of flow classes according to their weights (or service tiers), where flows in the same class have same weight. Within the same class, flows are further quantized into a number of token queues based on the sizes of their head packets for efficient timestamp sorting. Note this quantization is done using the sizes of head packets of the flows, flows are thus expected to be quantized into different token queues at different time. A limited number of token queues for each class are only needed due to the fact that IP packet sizes exhibit a few modes and majority of packets fall into one of those modes. Similar to SRR and FRR, TSFQ uses two levels of scheduling. The intra-class scheduler selects which flow to be served first within that class by looking at the limited number of token queues. This can be done within constant time. The inter-class scheduler then picks which class to be served based on the minimum timestamp of the limited number of tier classes. TSFQ also requires per-flow queueing and needs to maintain per-flow state.

Stochastic fair queueing (SFQ) [63] is a binning scheme. It uses a hash function to randomly quantize flows into a limited, fixed number of bins. The number of bins to be used is irrespective of the number of flows. The bins are scheduled round-robin. It can be viewed as a probabilistic variant of fair queueing. Due to its use of the limited number of bins, some flows will likely collide and be mapped to the same bin. Good fairness is only possible with the use of a fairly large number of bins such as at the order of thousands or more. To avoid the same set of flows continuously colliding on the same bin, SFQ periodically perturbs the hash function so that flows that collide at one time will less likely collide at another time. SFQ does not handle flow weights. Multiple queue fair queueing (MQFQ) [33] is a scheme similar to SFQ but MQFQ uses more than one hash function to map a flow to multiple queues (bins). Then each flow has the option to queue its packets on the shortest queue associated with the flow. Since packets from the same flow under MQFQ can be queued on multiple bins at any time, MQFQ has to deal with packet reordering issue all the time. While on the surface, this can be avoided by always placing the packet on the shortest queue mapped to a flow. But the exact identification of the shortest queue depends on a few factors such as the deficit counts of the queues. Extra effort is required to avoid packet reordering.

In terms of the complexity as described early in the section, we summarize the above schemes as below:

- Per-flow queue required: WFQ, SCFQ, DRR, SRR, FRR, TSFQ

- Per-flow state required (but not per-flow queue): BSFQ
- Quantization used: SRR, FRR, TSFQ, BSFQ, SFQ
- Binning used: SFQ, BSFQ, FCFS

2.2.1.1 Diff-serv Scheduling

Besides the scheduling algorithms mentioned above and largely for best-effort data traffic, there have been a class of scheduling algorithms developed for types of traffic providing guaranteed services or differentiated services [96, 20].

Differentiated services represents the extensions to TCP/IP and the Internet architecture required to provide an alternative service to the best effort datagram service. These scheduling algorithms are aimed to provide quality of service (QoS). The system model assumes traffic is marked by the generating host or on entry to a diff-serv managed network requesting a specific diff-serv behavior.

Despite tremendous efforts that have been made in the past to get the Internet to provide support for QoS through differentiated services (this is evident by the existence of more than a dozen of RFCs on diff-serv), the Internet still largely delivers only one type of service (that is best-effort). Teitelbaum et al. explained this in a paper titled "*Why Premium IP Service Has Not Deployed (and Probably Never Will)*" [89]. For this reason, there has been a focus shift to make best-effort service to provide relative QoS [41].

The diff-serv research that is of more relevance to our direction are the approaches that require a network monitor to classify a flow as being within its traffic specification ('in spec') or as having violated its specification. In the former, packets in the flow might be marked as operating within its profile or as out of its profile in the latter. The packet scheduling at routers would map the traffic marked as 'in spec' to an appropriate diff-serv service. For example, RED with In and Out [11] assumes a service allocation profile is associated with each flow. Two queues are needed to offer two levels of service assurances. Each queue uses RED preferentially drop packets that are marked 'in' or 'out' of its profile.

A few diff-serv schemes address the different requirements of flows in terms of low latency or high throughput. Alternative best effort (ABE) [41] assumes packets are marked as either green or blue. By using two queues, routers guarantee green packets with a low delay bound. But during congestion, green packets are more likely to be dropped than blue packets. A rate-delay (RD) network service differentiation scheme [80] is proposed to give the user an opportunity to choose between low delay and high throughput. A RD router uses two queues to support such services.

There have been several recent papers that describe ideas that treat ‘elephant’ flows and ‘mice’ flows differentially based on observed behaviors at the scheduler. MultiBuff [66] is a scheme that is proposed to isolate short flows for low delays and long flows for high throughputs into different buffers in a data center network environment. SplitBuff [42] is a scheme that isolates flows of different RTTs into multiple buffers of varying sizes to provide low delays for short RTT flows and high throughput for long RTT flows.

2.2.2 Queue Management

The simplest buffer management scheme, DT, drops arriving packets when the queue has reached the configured maximum capacity. There are well known problems that arise in certain situations involving DT. RFC 7567 [4] identifies four major drawbacks of this simple scheme: 1) Full queues, which can cause significantly long packet delays. 2) Lock-out, in which situation one flow can monopolize queue space, starving others. 3) Mitigating the impact of packet bursts, which can disrupt the TCP control loop and reduce performance of flows. 4) Control loop synchronization characterized by all TCP flows holding back nearly at the same time causing network resource under utilized.

AQM has long been considered the appropriate solution to these issues. The Random Early Detection (RED) algorithm [32] manages a queue by randomly dropping packets in a manner in which the random drop rate is dynamically adjusted based on an average queue size estimate and a configured maximum allowed drop rate (referred to as *maxp*). Most RED implementations offer the ‘gentle’ option where the drop rate increases linearly from *maxp* to 1 once the average queue level exceeds the target queue size [82, 28]. While RED is widely available, it is not widely used. It has been shown that the average queue delay with RED is sensitive to traffic loads and to parameter settings [62, 54]. Adaptive RED (ARED) is a simple extension to RED that further adapts the random drop process such that the average queue level tracks a target queue level [30]. This adaptation is performed periodically. We refer to this parameter as the *control_interval*. Due to the difficulty in choosing the appropriate set of parameters for RED, RFC 7567 no longer recommends the use of (A)RED.

Feng et al [27] indicated that queue length is a wrong congestion indicator to use. Instead, they proposed BLUE active queue management that uses packet loss and link idle event to adapt the mark/drop probability. BLUE requires little or no tuning. To isolate unresponsive flows, Feng et al further proposed stochastic fair blue (SFB) to identify unresponsive flows to be rate-limited. How such flows should be limited is not clear.

Long delay associated with bloated buffer is a much larger issue with many applications. CoDel [73] and PIE [77] are two recent delay-based AQMs to directly tackle the delay associated with bloated buffer. Both AQMs proactively drop packets to ensure average packet queueing delay remains less than a configured latency target. We refer to this as the *target_delay* parameter. Both AQMs expose a second configuration parameter analogous to the *control_interval* of ARED that defines the time scale of control.

CoDel's delay estimate is based on a per packet latency monitor. PIE's delay estimate is based on an estimate of the recent departure rate at the queue. The two AQMs both tolerate infrequent bursts of traffic. However, the details of the burst control mechanisms differ and are described in [73] and [77] respectively.

The PIE algorithm performs early packet drops as packets arrive at the queue. The CoDel algorithm, as originally proposed in [73, 71] performs early packet drops as queued packets are serviced. The DOCSIS vendor community has expressed concern in implementing a head-drop AQM such as CoDel in a CM due to complications with hardware buffer control logic. This is likely to be true in other low cost network devices such as

Ethernet switches or WiFi APs.

2.2.2.1 AQM with Fairness

AQMs that do not distinguish between traffic flows drop packets from all flows indiscriminately with same probability resulting unfair sharing of the link capacity. AQMs involving per-flow accounting or the use of multiple queues have been proposed to address the combined problem of fairness and bufferbloat.

Flow random early detection (FRED) [54] is a modified RED that uses per-flow accounting to impose different drop rate on each flow. Several schemes attempted to further reduce the complexity of flow-based AQMs through approximation. CHOKe [78] is an AQM that discriminately drops more packets from a flow that sends more packets than is allowed by its fair share to approximate fair bandwidth allocation. However, CHOKe does not directly maintain per-flow states. Instead the information is implicitly extracted from the queue it manages (without much accuracy). When a new packet arrives, CHOKe randomly picks a packet from the queue. If the two packets belong to the same flow, the packet drop probability for the packet(s) of the flow is made higher. Improving over CHOKe, approximate fair dropping (AFD) [76] further expands the idea. AFD uses a small shadow buffer to keep recent packet headers (insertion and deletion implied). It also uses a flow table based on hash table to hold just enough flow information for the limited number of recent active flows in the shadow buffer. AFD can potentially provide more accurate flow information than CHOKe resulting in more accurate and granular bandwidth management. There is clearly a cost associated with maintaining the flow table and shadow buffer per packet.

Delayed-based AQMs that also address fairness issues have been proposed. Approximated-fair controlled-delay (AFCD) queueing [94] is a single queue AQM that blends AFD and CoDel to achieve fairness based on per-flow accounting information. With AFCD, per-flow delay targets are calculated to adjust for flow sending rates. A flow whose sending rate exceeds its fair share will be controlled using a shorter delay target compared to other flows so that more packets can be drop from such flow to signal the flow to reduce its sending rate.

FlowQueue-CoDel [36] is a multiple queue AQM that blends a modified DRR packet scheduler with CoDel managing each queue to simultaneously address fairness and bufferbloat. FlowQueue-CoDel is referred to as FQ-CoDel. FQ-CoDel uses a fixed number of queues irrespective of the number of flows. An implementation of FQ-CoDel for ns2 is called SFQ-CoDel [70]. Broadly the combination of any FQ scheduler based on per-flow queues with CoDel can also likely be called FQ-CoDel. We identify this class of scheduling as FQ+CoDel in Figure 2.1 to avoid any confusion. (S)FQ-CoDel stochastically hashes incoming packets into a fixed number of queues (bins) based on packet headers. The bins share a fixed-size common buffer and are scheduled with a modified DRR. Each queue is separately managed by CoDel but with a common delay target. (S)FQ-CoDel is shown to be very fair while maintaining delay target consistent to that of CoDel. But it does not provide weighted fairness when flows are weighted. (S)FQ-CoDel has been actively evaluated [47, 93]. A Linux implementation is also available [22].

AQM, as summarized in [4], provides these advantages for responsive flows: 1) reduced number of packets dropped by network devices; 2) lower delay for interactive services; 3) preventing buffer lock-out behavior by ensuring there will almost be a buffer for an incoming packet; 4) reduced probability of control loop synchronization.

2.2.3 Summary of Related Work

The issues surrounding packet scheduling and buffer management are perhaps the most widely studied issues in networking. We have been motivated particularly by the following directions:

1. Low complexity packet schedulers that approximate fair queueing through quantization, which defines a hierarchical scheduling framework. The framework involves an inner scheduler that maintains fairness among flows within the same bin and an outer scheduler that maintains fairness across bins.

2. Combined scheduling and buffer management schemes such as (S)FQ-CoDel and AFCD.

Fundamental to assessing the performance such as fairness, latency, and system efficiency is the choice of time scale. The crux of the problem centers on identifying the time scales of interest. Fair queueing based on GPS maintains fairness for any infinitesimal time scale. Approximations of FQ loosen the time scale of fairness to times based on a single packet transmission time, a round-robin 'round', and possibly many RTTs control feedback loops potentially require. AQM schemes that maintain a small amount of state relax the time scale to the amount of time it takes to identify and differentiate the elephants from the mice. The intuition behind these schemes is that the vast majority of traffic over relatively short time scales (e.g., minutes) can be classified as either an elephant or a mouse.

Our direction assumes time scales of fairness similar to that of AQMs with state information. But we attempt to improve the accuracy of the fairness outcome by assuming that subscriber bandwidth management requires more fairness granularity than just identifying elephants or mice. As illustrated in Figure 2.1, our direction is positioned roughly between AFD/AFCD and (S)FQ-CoDel (this assumes a scenario that involves a single tier). When compared to AFD/AFCD, we seek to provide better fairness (more than just separating elephants and mice). When compared to (S)FQ-CoDel, the proposed scheme reduces complexity and cost with the use of just a few aggregate queues (bins). We do not intend to provide short-term fairness as (S)FQ-CoDel does. We focus more on fairness over large time scale such as tens of seconds.

2.3 Overview of DOCSIS Operation

The Data Over Cable Service Interface Specification [8] is a set of protocols and standards developed by Cable Television Laboratories (CableLabs) to facilitate the delivery of Internet services over traditional Cable TV networks.

DOCSIS system uses shared medium cable as its physical layer. At customers' premises, cable modems (CMs) are attached to the cable. Their access to the cable medium is controlled by a device called Cable Modem Terminating System or CMTS at the cable operator's plant. Modern cable networks involve a hybrid fiber-coaxial infrastructure that establishes the connection with a DOCSIS MAC operating between the CMTS and the CMs.

Data packets flow between the CMTS and the CMs through so called channels that are each shared by multiple CMs. A CMTS manages multiple channels simultaneously. The most widely deployed version of the standard, DOCSIS 3.0, uses 6 MHz (or 8 MHz in certain regions) of bandwidth for the shared downstream channel and up to 6.4 MHz of bandwidth for the shared upstream channel. The 6 MHz downstream channel supports physical layer data rates up to 42.88 Mbps (55.62 Mbps in 8 MHz regions) and the upstream channel supports data rates up to 30.72 Mbps. Packets sent over the downstream channel are broken into 188 byte MPEG frames each with 4 bytes of header and a 184 byte payload. Prior to DOCSIS 3.0, CMs were limited to a single downstream channel and a single upstream channel (although, for the purposes of load balancing, the CMTS could dynamically change a CM's downstream or upstream channel assignment). DOCSIS 3.0 allows a CM to support multiple downstream or upstream channels. A bonding group is a set of channels that specific subscriber traffic can use. A downstream service group is the complete set of channels that can potentially reach a CM. A bonding group is a specific subset of channels from the service group assigned to carry subscriber traffic.

Downstream channels employ time division multiplexing for sharing. A downstream scheduler at the CMTS manages the allocation of bandwidth among competing service flows. A DOCSIS service flow is a transport service that provides unidirectional transport of packets. A service flow consists of one or more TCP/IP connections terminating at a specific CM. Service flow traffic may be shaped and prioritized based on QoS traffic parameters associated with the flow. For downstream, the service parameters that define a service flow include priority, settings for rate shaping and limiting (sustained traffic rate, traffic burst size), a minimum reserved traffic rate, a peak traffic rate, and target latency. The standard does not specify how a specific scheduling implementation should differentially treat traffic from different priority levels.

The upstream channel is time division multiplexed with transmission slots referred to as mini-slots. Permission to transmit data in a block of one or more mini-slots must be granted to a CM by the CMTS. The CMTS grants mini-slot ownership by periodically transmitting a frame called the MAP on the downstream channel. In addition to ownership grants, the MAP also typically identifies some mini-slots as contention slots in which CMs may bid for quantities of future mini-slots. To minimize collisions in the contention slots, a non-greedy backoff procedure is employed. When a CM has a backlog of upstream packets it may also “piggyback” a request for mini-slots for the next packet at the tail of the current packet.

2.3.1 DOCSIS 3.1

The latest development to DOCSIS is the DOCSIS 3.1 standard. To achieve its ultimate goal of supporting 10 Gbps downstream and 1 Gbps upstream network capacity, DOCSIS 3.1 allows cable operators allocate spectrum differently from before and make efficient use of the bandwidth. DOCSIS 3.1 adds a new physical layer that uses wideband orthogonal frequency division multiplexing (OFDM) channels downstream and orthogonal frequency division multiple access (OFDMA) channels upstream. Unlike a traditional QAM channel fixed at a width of 6 or 8 MHz with a single carrier, an OFDM channel can be configured to occupy a spectrum from 24 MHz up to 192 MHz in the downstream, which is composed of a large number of subcarriers that each is 25 kHz or 50 kHz apart. Similarly OFDMA channels are also multicarrier channels of 25 kHz or 50 kHz subcarriers. An OFDMA channel can occupy a spectrum up to 96 MHz. Due to the use of many subcarriers in an upstream channel, multiple CMs on the same upstream channel can now send data packets to CMTS simultaneously on different subcarriers. The new scheme now enables a very large data pipe through the use of only a single channel.

Other significant changes include new adaptive modulation and coding, and higher QAM modes. At physical layer a new FEC coding called low-density parity-check (LDPC) code is defined and higher QAM modulation (from 1024 to 4096) is allowed. These changes provide more efficient use of the spectrum.

At the downstream MAC to PHY convergence layer, DOCSIS 3.1 no longer uses MPEG-2 as an intermediary and now allows MAC packets to be encoded into codewords directly. This allows IP data packets to be encapsulated more efficiently.

At the MAC layer, DOCSIS 3.1 continues to support channel bonding. This feature now allows the bonding of OFDM / OFDMA channels leading to high capacity with just a few channels bonded. In addition, it allows the mix bonding of the legacy single-carrier channels and the OFDM / OFDMA channels, where an OFDM / OFDMA channel is treated like a single-carrier channel.

DOCSIS 3.1 clearly spells out that it requires AQM to reduce the buffering latency in CM and CMTS. This requirement is perhaps one of its most visible requirements. It has a goal of improving responsiveness for applications and quality of experience for users. CMTS must support a default AQM scheme. However, the exact AQM scheme to be used for CMTS is not specified by DOCSIS. This opens the door for CMTS vendors to come up with innovative schemes. On the contrary, CM must support the PIE AQM algorithm [77]. By default, AQM must be enabled.

2.3.2 Review of DOCSIS Related Work

Early work on bandwidth management of DOCSIS networks is well represented by Droubi [21] and Kuo [50]. The focus was on packet scheduling. Droubi et al. proposed a CMTS scheduling mechanisms for downstream and upstream transmissions and studied their performance for delivering QoS. The scheduling algorithm in use was SCFQ. Kuo et al. proposed a scheduling service and bandwidth allocation algorithm for upstream flows. Representative literature on DOCSIS simulation and performance models include [59, 84, 85].

Most of the early research focused on upstream bandwidth allocation algorithms. This is because the downstream scheduling prior to DOCSIS 3.0 was relatively trivial. With the introduction of channel bonding feature in DOCSIS 3.0, downstream packet scheduling is no longer trivial. Nikolova et al. proposed two DRR based multi-channel downstream packet schedulers to support rate based services [74]. Moser et al. studied downstream fair packet scheduling and dynamic load balancing in channel bonded networks [65, 37]. In DOCSIS 3.1, AQM has become one focus in the management of DOCSIS networks. White et al. studied the latest CoDel AQM and its performance in comparison to simple drop-tail scheme in a simulated DOCSIS 3.0 environment [92].

In summary emerging DOCSIS systems are going through significant changes. Future systems will involve Gbps access speed and delay-based AQMs. These developments collectively represent systems that have not been studied by the academic community.

Chapter 3

System Description

In this chapter, we lay out foundations of the research from a systems perspective. We first present a conceptual system model, including our definition of fairness. We then present our simulation model for the experiments and analysis. We further introduce the traffic models used in the simulation and the performance metrics used in the analysis.

3.1 System Model

In this section, we first present a reference bandwidth management model. The model is inherently complex. We then introduce a low complexity approximation for the model. How the model is applied to DOCSIS is also discussed.

3.1.1 Reference Bandwidth Management Model

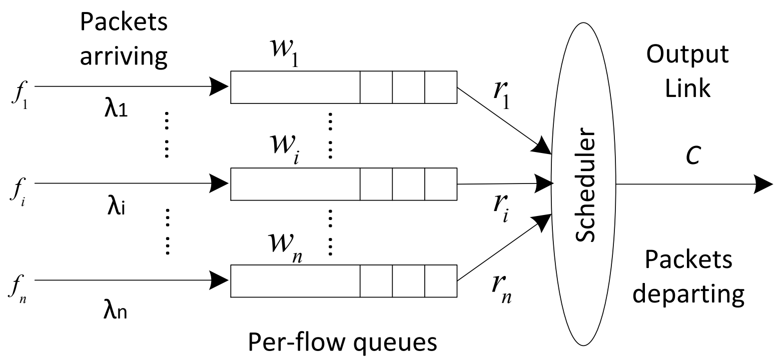


Figure 3.1: Reference Bandwidth Management Model

Figure 3.1 shows the reference bandwidth management model based on fair queueing. At time t, there are n flows $\{f_i, 1 \leq i \leq n\}$ with an arriving rate λ_i respectively sharing a total bandwidth of C. The flows are scheduled with a weighted fair queueing scheduler that requires one queue per flow. Each flow f_i is assigned a weight w_i. The bandwidth allocated to each flow should be in proportion to its weight. Let B be the set of backlogged flows. A flow is backlogged if it has packets to send at the time. If f_i is backlogged, the bandwidth or service rate it should receive is given by:

$$r_i = \frac{w_i}{\sum_{f_j \in B} w_j} C$$

Over a heterogeneous set of active flows, the reference model will lead to allocations that are weighted max-min fair.

3.1.1.1 Application to DOCSIS

In the literature, the term *flow* generally means a stream of IP packets that collectively form an end-to-end communication session uniquely identified using a 5-tuple IP packet header information (the source/destination IP address and port number plus protocol number). We refer to this definition of a flow as an IP flow. A DOCSIS service flow represents any number of IP flows that terminate at a specific CM and that are treated in aggregate according to the service and QoS definitions for the service flow. For example, a CM by default is provisioned with 2 service flows, one for all downstream best effort traffic and one for all upstream traffic. To support telephony, a CM might have two additional service flows provisioned allowing the system to meet specified QoS requirements assigned to the service flows. In the work presented in this dissertation, we assume all downstream and upstream traffic associated with a CM map to the two default service flows. When we use the term flow, we assume the DOCSIS perspective meaning a flow represents one or more IP flows.

Applying the above bandwidth management model to DOCSIS, weights for individ-

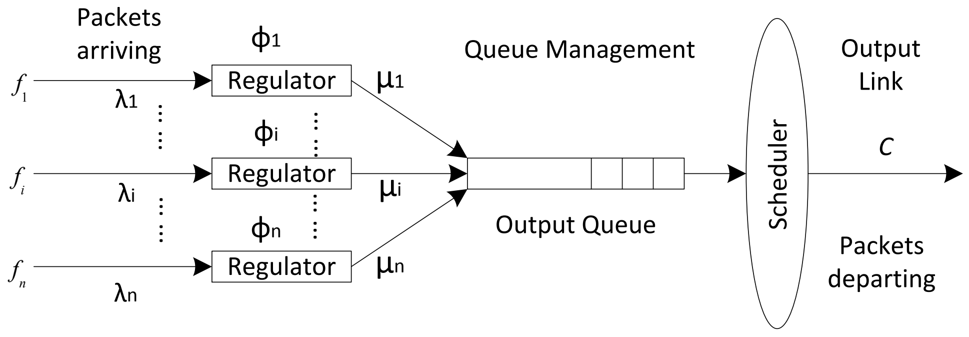


Figure 3.2: Current DOCSIS Downstream Bandwidth Management Approach

ual flows can be set to match the different levels of service subscribers paid for. Given the current practice of the service tier definition being in terms of individual imposed maximum subscriber service rates, the weights can be simply set to be in proportion to the service rates.

The reference bandwidth management model is inherently complex due to its use of one queue per flow. To lower the complexity, current cable systems often reduce the model to the use of a single queue and use regulators to enforce the individual maximum subscriber service rates. This results in a bandwidth management approach as illustrated in Figure 3.2.

In the figure, each flow f_i is individually regulated to not exceed a maximum service rate ϕ_i. Network operators often identify a discrete set of available maximum service rates for users to choose from. Let $\Phi = \{\varphi_j, 1 \leq j \leq p\}$ be the set of service rates, where p represents the number of the service tiers offered. Each flow f_i is then regulated to have a departure rate $\mu_i \leq \phi_i \in \Phi$. The DOCSIS standard specifies a token bucket is to be used to provide rate shaping and limiting. In this dissertation, we assume the term service rate is the maximum sustained traffic rate described in the standard.

With this approach, all packets irrespective of which flows they are from, after they are eligible for transmission following the regulator process, are managed using a single aggregate queue with a simple FCFS scheduler. The buffer management technique often used for the aggregate queue is the simplest DT scheme. More sophisticated active queue management schemes including the delay-based AQMs may or may not be used. The support for service tiering depends on the use of the regulators which impose a max subscriber service rate for each flow individually. Due to the use of FCFS scheduler, this management approach does not support flow weights and therefore does not guarantee the bandwidth allocated to f_i to be r_i.

We call this service model the *conventional service tiering model*, based on our current understanding of standard practice for cable networks. In this service model, the network utilizes service rates for rate limiting and shaping with the resulting regulated traffic to be aggregated and managed by a single queue scheduled FCFS. The model does not use flow weights and therefore does not provide weighted allocation. The bandwidth allocated to any flow is subject to its max service rate.

3.1.2 Approximate Fair Bandwidth Management Model

Due to the complications associated with the use of per-flow queues in the reference bandwidth management model, we present an alternative bandwidth management model that intends to approximate weighted fair allocation.

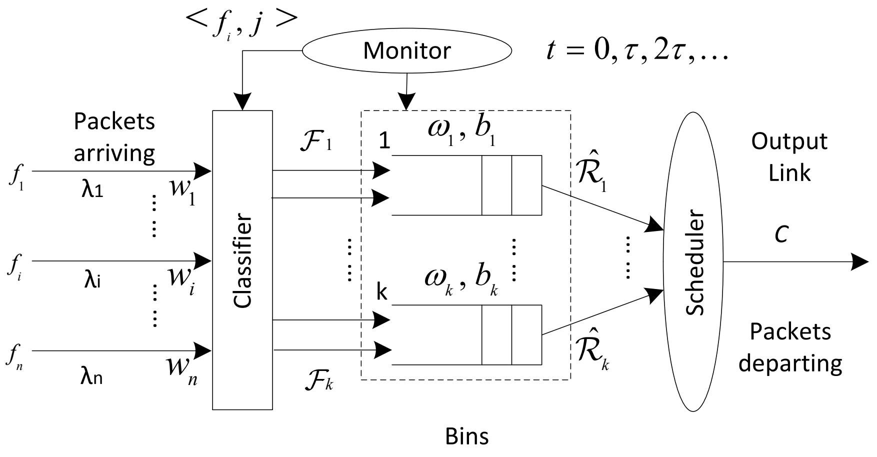


Figure 3.3: Approximate Fair Scheduling Using Adaptive Bandwidth Binning

Figure 3.3 illustrates the idea of the approximate fair bandwidth management model. The model only uses a fixed number of k bins where k is irrespective of n and it is expected to be a small number. Let τ be a time interval called reclassification interval. At time $t = 0, \tau, 2\tau, 3\tau, \ldots,$ f_i is (re-)mapped to bin j by a mapping $<f_i, j>$. The classifier in the model generates such mapping $M_t = \{<f_i, m_i^t>, (1 \le i \le n) \wedge (m_i^t \in \{1, 2, .., k\})\}$ for all flows. At each interval time t, the set of flows mapped to bin j is $\mathcal{F}_j^t = \{f_i, <f_i, j> \in M_t\}$.

Note at each reclassification interval time t, M_t can change. This may result in the changes of $\mathcal{F}_j^t$ $(1 \le j \le k)$ due to flows being remapped (or reclassified). Which bin j flow f_i is mapped to at t is controlled by a quantization method based on a novel approach we refer to as adaptive bandwidth binning (ABB). Let b_j be the bandwidth consumption cap for bin j $(1 \le j \le k)$. Without losing generality, let $b_j < b_{j+1}$ so that bins are arranged in increasing order in terms of bandwidth consumption caps. Let $b_0 = 0$ and $b_k = C$. The range of bandwidth consumption level for bin j is then between b_{j-1} and b_j. Let $\hat{r}_i^t$ be the moving average consumption rate of f_i, calculated using the flow consumption rate samples from the past intervals, at time t. Flow f_i is mapped to bin j if $b_{j-1} < \dfrac{\hat{r}_i^t}{w_i} \le b_j$ and the mapping generated by the classifier for f_i at time t is $<f_i, j>$. $\dfrac{\hat{r}_i^t}{w_i}$ is said to be the normalized flow consumption rate of f_i at time t. In this way, flows of the similar normalized consumptions are classified into the same bin. When $\hat{r}_i^t$ changes over time as often the case in the real system, f_i is likely to be remapped to a different bin at next reclassification interval time according to the adaptive bandwidth binning scheme. This results in the changes of the mapping M_t periodically. Flows are remapped and the system adapts to the changing flow consumption level. Exactly how b_j is assigned to bin j and how packet scheduling is carried out are given in the algorithms in Chapter 5.

The flows in the approximate model are scheduled with a two-level packet scheduling framework. Flows in the same $\mathcal{F}_j^t$ of bin j are scheduled by an intra-bin scheduler. Each bin uses a single aggregate queue and therefore the intra-bin scheduler in use is FCFS. The bins $\{j, 1 \le j \le k\}$ are scheduled with an inter-bin scheduler (also called outer scheduler). The outer scheduler can be any weighted fair queueing scheduler. The weight for each bin j is given by $\omega_j^t = \sum_{f_i} w_i$ where w_i is the weight of $f_i \in \mathcal{F}_j^t$. When $\mathcal{F}_j^t$ changes at t, ω_j^t also changes.

Several mechanisms play a role in the model to approximate fair scheduling over a time span that covers multiple reclassification intervals. The first mechanism is bandwidth binning described in the previous paragraphs. The intuition behind the idea is that flows that consume similar levels of bandwidth in the recent past are likely to continue to exhibit similar behavior in the next scheduling interval. If this proves to be not true, or if the flows within a bin do not compete fairly, the flows consuming more than the fair share of the flows in the bin will likely be mapped to a different bin in the future. This adaptive control ensures that a flow is serviced by the set of bins such that it receives approximately its fair share of available bandwidth over time scales of multiple scheduling intervals. The final necessary mechanism is the outer scheduler that ensures each active bin j receive a fair share in proportion to ω_j^t. Let $\hat{\mathcal{R}}_j^t$ be service rate bin j receives at t. Assume every flow in $\mathcal{F}_j^t$ of bin j is backlogged and the intra-bin scheduler is able to provide weighted fair allocation for flows in the same bin, the service rate flow f_i in $\mathcal{F}_j^t$ receives is:

$$r_i^t = \frac{w_i}{\omega_j}\hat{\mathcal{R}}_j^t$$

Assume all flows in the system are backlogged. Let W be the sum of all flow weights, and we have:

$$\hat{\mathcal{R}}_j^t = \frac{\omega_j}{W}C$$

Then we have:

$$r_i^t = \frac{w_i}{\omega_j}\hat{\mathcal{R}}_j^t = \frac{w_i}{W}C$$

This is the expected allocation for f_i under the reference bandwidth management model.

Note the above analysis is *only* intended to show how bandwidth is allocated to the flows through the bins. It is not correct without those unrealistic assumptions. In a real system, flows are not always backlogged. The service rates for the bins and the flows can change at any time depending the workload at the time. The intra-bin FCFS scheduling cannot provide weighted fair allocation. These factors, along with protocol effects that can include unforeseeable TCP dynamics, will lead to an allocation that only approximates fair scheduling over multiple reclassification intervals.

The approximate bandwidth management model lowers the complexity by using a small number of bins, each managed using FCFS, and removes the complications associated with the use of per-flow queues. The complexity of the outer scheduler based on DRR is $O(1)$.

The approximate model exposes a number of interesting research questions. For example, how will it meet the design goal as given earlier in the Introduction? What is the best operating region in terms of the system parameter choices? As given in Figure 2.1, the design tradeoff is relaxed fairness with lower complexity (as compared with approximations to FQ). The optimal number of bins is an interesting issue. It is likely that shorter reclassification interval produces better approximation but at the same time increases cost. A compromise for a reclassification interval to be on the order of seconds is appropriate as our simulation study indicates.

3.1.2.1 Application to DOCSIS

We now apply the approximate bandwidth management model to DOCSIS system. Each flow is first classified into a bin among the k bins. Flows that are classified to the same bin are aggregated to use a single aggregate queue and scheduled FCFS. The bins will be scheduled with weighted DRR scheduler as the number of bins to be scheduled is small and the scheduler is simple to implement. To support the periodical remapping of flows, a timer with the interval set to τ should be used to trigger the reclassification process based on adaptive bandwidth binning. To address the bufferbloat problem, the bins will be managed by CoDel.

The conventional service tiering model supports service tiers defined in terms of individually imposed max service rates through the regulator process. It does not use flow weights. The allocation is unweighted except each allocation is subject to the max service rates. During the time of congestion, it cannot provide better allocations for flows of higher tiers. This is apparently undesirable for high end users who paid more.

With the low complexity / cost approximate bandwidth management model, we now define a *new service tiering model* that allows us to add back the flow weights and offer a better tiered service.

In the context of DOCSIS downstream management, the flow weights are assigned in proportion to the tiered service quality levels a subscriber purchases. The conventional service tiering model can map its service tiers based on max service rates to a number of service quality levels without the explicitly specified service rates. Without losing generality, let $\min\{w_i, 1 \leq i \leq n\} = 1$. We assume:

- A flow f_i subscribed to basic level of service in term of bandwidth allocation is given a weight $w_i = 1$. The flow is a *Tier1* flow.
- A flow f_j subscribed to two times better level of service than the basic level of service is given a weight $w_j = 2$. The flow is a *Tier2* flow.
- A flow f_k subscribed to X times better level of service than the basic level of service is given a weight $w_k = X > 1$. The flow is a *TierX* flow.

For example, a backlogged Tier4 flow should obtain twice the allocation of a backlogged Tier2 flow and 1/2 of the allocation of a backlogged Tier8 flow.

Note this new tiering model has similarities and differences from the conventional tiering model as currently used by service providers. This new tiering model continues to support pricing-based economic model. However this new tiering model clearly specifies what level of services the flows should receive relative to each other no matter whether the system is during the period of congestion or not. Thus the bandwidth allocation under the new tiering model should become more predictable. For example, a higher tier flow should always be expected to receive more bandwidth than a lower tier flow during the time of congestion.

The new tiering model does not impose a hard-set max service rate for any flow and thus allow such flow to have access to a high percentage of the channel capacity when

Table 3.1: Different Allocations (Mbps) For Three Tiered Flows Under Different Tiering Models

Max Subscriber Service Rate / Tier	8 / Tier1	16 / Tier2	32 / Tier4
Allocation Per Conventional Tiering Model	8	13.5	13.5
Allocation Per New Tiering Model	5	10	20

the system is not loaded. We think this is important for D3.1 network to achieve its full potential. This noticeable difference moves the bandwidth management in DOCSIS cable systems towards a more work-conserving scheme. The new tiering model does not preclude the use of max service rates for the flows either. For example, when rate limiting is desired, a system wide max service rate ϕ (e.g., $\phi = C/2$ or $C/5$) may be enforced for all flows to protect the network from overly aggressive flows. We do not consider this option.

To illustrate the differences in the bandwidth allocation between the conventional and the new tiering models, we use the following example. Assume we have 3 flows that have sufficient demands to overload the system and are assigned to 3 tiers respectively. They share a total channel capacity of 35 Mbps. In the context of conventional tiering model, the 3 tiers are defined in terms of max subscriber service rates of 8, 16, and 32 Mbps respectively. The allocation is *unweighted but subject to the max service rate caps.* In the context of the new tiering model, the 3 tiers are defined in terms of relative service quality levels (Tier1, Tier2, and Tier4) and thus are given the weights of 1, 2, and 4 respectively. The allocation is *weighted without rate caps.*

Table 3.1 shows the different allocations of 35 Mbps among three tiered flows under the two different tiering models. As we can see, under the conventional tiering model, the two higher tiered flows are allocated the same bandwidth during the time of congestion. This is unfair to the subscriber who pays more to subscribe to the highest tier. The allocation under the new service tiering model better reflects the levels of service different subscribers purchase.

3.1.3 Defining and Assessing Fairness

In the context of bandwidth management, we assess only the throughput fairness in contrast to latency or jitter fairness, which is the focus of other work such as WF^{2}Q [7].

Defining fairness is challenging as the term has unique meaning in different contexts. As mentioned in the Background Chapter, the issue requires the time scales of interest to be identified. For packet scheduling based on approximations of fair queueing (e.g., WFQ [17], SCFQ [35]), fairness is assessed on packet transmission time scales. For example, in [86], a metric defined as the difference of services received by any two active flows during an arbitrary time interval is used to assess fairness of a scheduling discipline. For a scheduling discipline to be fair the metric must be a small constant that does not depend on the size of the time interval. In other words this metric states that fairness is defined by the worst case difference in the service received by any two active flows over any time interval.

In a scheduling environment that is based on single queue AQM such as AFD [76] and AFCD [94] that attempt to isolate the elephants from the mice, the definition of fairness is significantly relaxed. A measure such as Jain's Fairness Index (JFI) [44] computed based on hundreds of seconds are common.

JFI is a widely used assessment of how achieved resource allocation differs from the desired outcome. It is computed as follows. Suppose there are n competing flows. Let $x_i = {}^{T_i}/_{r_i}$ where T_i is the achieved throughput of the i-th flow and r_i is the expected outcome. The JFI is defined as:

$$\mathrm{JFI}(x) = \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n \sum_{i=1}^{n} x_i^2}, \quad x_i \geq 0 \tag{3.1}$$

Another similar fairness index is called min-max ratio, which is defined as:

$$\mathrm{MMR}(x) = \frac{\min_i\{x_i\}}{\max_i\{x_i\}}, \quad x_i \geq 0 \tag{3.2}$$

Both indexes range from 0 to 1, with ideal fairness represented by a value of 1.

We assume that *max-min fair* throughput allocation criterion is the desired fairness objective. It has been shown that packet scheduling algorithms that implement close approximations of fair queueing, such as DRR packet scheduling [86], can achieve max-min fairness. In recent work [37], we have shown that DRR packet scheduling also achieves max-min fair allocation in downstream D3.0 bonded channel environments when no channel is assigned to more than one bonding group at a time. In our work, we assume fairness is based on very large time scales of multiple seconds or minutes. Such large time scale is reasonable and acceptable in practice.

For our simulation study, we primarily use JFI and MMR to quantify fairness in achieved throughput among competing flows. The standard deviation (Stddev) and coefficient of variation (CoV) are also occasionally used. Their definitions are those of standard mathematical definitions which we do not intend to repeat.

3.2 Simulation Model

The research is simulation based. We carry out all our studies in a simulated DOCSIS 3.x cable environment on the ns-2 platform [1]. In prior work, we have developed an ns-2 based simulation model for DOCSIS [59].

3.2.1 Simulated Network

Figure 3.4 illustrates the simulation network model used in our studies. We model a single DOCSIS MAC domain in which one CMTS interacts with a number of CMs. In the D3.0 case, the model assumes channels that offer downstream and upstream physical layer data rates of 42 and 30 Mbps respectively. For D3.1, we assume the physical layer data rates of 1 Gbps and 100 Mbps for downstream and upstream channels respectively. Each CM represents a subscriber who, in practice, might have multiple TCP/IP devices interacting with a variety of Internet services. However, in the experiments described below, each CM hosts a single IP data flow. Various link speeds and delays in the figure can be configured to model Internet path diversity.

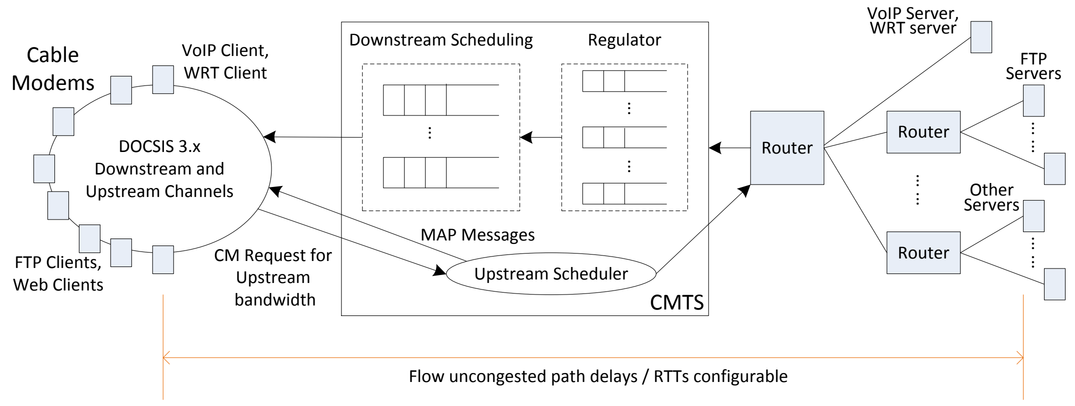


Figure 3.4: Simulation Network Model

The scheduling and queue management algorithms are located in the Downstream Scheduling component of Figure 3.4. Exactly how many scheduler queues to be used are determined by individual scheduler and queue management disciplines. For example, for DRR, there is a dedicated queue for each flow. But for CoDel with FCFS scheduler, all flows are served through a single shared FIFO queue.

The simulation network model also includes a regulator to support the conventional service tiering model based on a maximum sustained service rate (or simply as the service rate). The regulator can be either enabled or disabled.

Packets for a service flow from the Internet can be regulated at the regulator to a maximum service rate through the use of a token bucket (Figure 3.5). Tokens are placed in a token bucket with fixed capacity at a constant rate. For example, for a flow rate to be controlled at r bits/second, tokens will be added to the bucket at r tokens/second. When the token bucket is full, extra tokens will spill over and disappear. Packets from a flow first arrive at a packet queue (token bucket queue). Packets arriving at a full queue will be discarded. When sent, a packet of n bits will consume an equal amount of n tokens. The system only sends out packets when enough tokens are available in the bucket. With appropriate settings the mechanism limits the rate of the flow going into the DOCSIS network to be at around the rate of tokens entering the token bucket.

Our implementation of the token bucket for DOCSIS flows supports the following

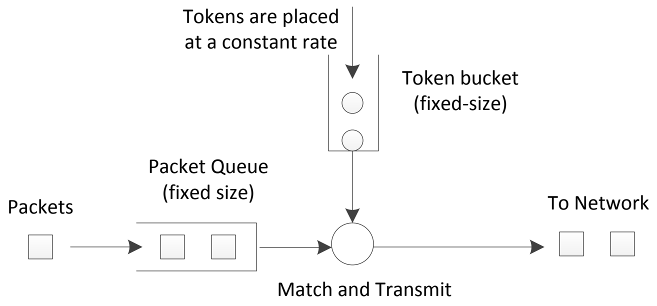


Figure 3.5: Operation of Token Bucket

three parameters:

Token Rate This gives the constant rate at which tokens are placed into the token bucket. It is the target service rate of a flow.

Token Bucket Size This is the capacity of the token bucket (in bits).

Token Bucket Queue Size This is the size of the token bucket queue in number of packets. It is the maximum number of packets a token bucket can buffer.

All of these parameters can be individually configured through simulation scripts.

3.2.2 Traffic Models and Performance Metrics

Our simulation experiments involve workloads consisting of a number of traffic types such as FTP, HAS, web, exponential on/off, and CBR traffic. Other than HAS, the traffic models we use are those provided in the ns2.

Besides some common performance metrics such as throughput and packet delay, we also use a number of performance metrics introduced below. Other specific metrics, only used in relevant context, will be covered where they are used.

3.2.2.1 HAS traffic model for ns2

As described in Sandvine's most recent Internet traffic report [83], Internet video streaming, commonly known as HTTP-based Adaptive Streaming or HAS, consumes more than 50% of the downstream bandwidth delivered to fixed access end points. HAS is an application layer protocol based on HTTP that utilizes TCP at transport layer.

The interaction between a HAS client (i.e., the player) and server has been established in recent academic research [3, 2, 39, 45]. In a HAS system, video and audio content is encoded and made available at servers located either at the content provider's facilities or distributed to locations in the Internet by a content delivery network provider. Multiple representations of the content are created that reflect a range of possible encoded bitrates corresponding to different levels of video quality and subsequently different levels of bandwidth consumption. While the range of bitrates is continually being extended, the literature suggests that a bitrate range of 0.50 Mbps to 5.0 Mbps is reasonable [3, 40, 16]. The client requests portions of the content in chunks referred to as *segments*. A segment size between 2 and 10 seconds is reasonable [40, 58]. The client maintains a playback buffer that serves to compensate for the jitter in the traffic arrival process. The literature suggests that a playback buffer capacity ranging from 60 to 240 seconds is reasonable [40, 58].

The HAS adaptation algorithm, which determines the video quality for the next segment requested by the client, is an active research topic. Initial ideas explored *capacity-based* approaches where the client monitors the rate of arriving video traffic and selects the quality level based on a prediction of the available TCP bandwidth [3, 52, 55, 90, 2, 53]. If the client predicts that available bandwidth might decrease over the next segment time interval, it requests a lower quality segment. If the prediction suggests that conditions are improving the client will 'switch-up' and request a higher quality segment. Other research, however, suggests that it is difficult for HAS to reliably predict available TCP bandwidth [2, 15, 68, 53]. In addition to variability of available bandwidth due to competing traffic in the network or from changing wireless channel conditions, rate-limited applications can induce pathological behaviors due to complex dynamics between the application and TCP's congestion control algorithm [39]. The use of *buffer-based* adaptation has been suggested to avoid the issues [40].

Evaluating a particular HAS design is challenging due to the many factors that impact an end user's perceived quality [19, 18, 5]. While the issue of assessing HAS quality of experience is intense study, the literature does suggest that the following measures are useful for evaluating HAS [19, 69, 2, 45, 48]:

- *videoPlaybackRate*: Attributes of the stream such as the resolution, pixel density, frame rate as well as end device capabilities all determine the base quality of the video stream. This measure represents the average bitrate of the stream based on the rate (in Mbps) at which data is dequeued from the playback buffer. If the video player stalls, samples are not recorded.

- *adaptationRate*: It has been shown that frequent adaptations and sudden changes to video quality are distracting [13, 69]. This metric counts the number of adaptations that occur during the simulation time. The measure is normalized to represent the rate of adaptations per hour.

- *rebufferTime*: Buffer stalls have a significant impact on perceived quality [19, 48]. Based on the ratio of total time the player is stalled to the stream duration, this measure represents the percentage of time that the player is stalled.

- *applicationBias*: The system should provide fair allocation. We focus primarily on the fairness allocated to groups of flows. This measure computes the percentage above or below the expected max-min fair allocation.

We have developed a HAS traffic model for ns2. The HAS client issues an HTTP request to the HAS server for the next segment of video content. The client specifies the video quality of the segment. The HAS server receives the request and simply sends the amount of data based on the requested segment size and quality using the same TCP connection. Data that arrive at the client are placed in the client's playback buffer. When the session starts, the client requests back-to-back segments to fill the playback buffer to a configured level (2 segments by default). Once this threshold is reached, the client video player starts and consumes one segment of video data at a time.

When the video player requires a new segment and the playback buffer holds less than one segment of video data, the player moves to a stall state where it remains until a configured number of segments arrive (2 segments by default). The client adaptation algorithm is based on a previously published capacity-based adaptation algorithm [55]. Each time the client issues a new segment request, it computes the ratio (μ as defined in equation 3.3) of the time required to play a segment (referred to as the media segment duration or MSD) to the download time of the previous segment (referred to as the segment fetch time or SFT).

$$\mu = \frac{\text{MSD}}{\text{SFT}} \tag{3.3}$$

The algorithm switches to the next higher quality level if

$$\mu > 1 + \varepsilon \tag{3.4}$$

where ε is defined as

$$\varepsilon = \max\left\{\frac{b_{r_{i+1}} - b_{r_i}}{b_{r_i}}, \forall i = 0, 1, \ldots, n-1\right\}. \tag{3.5}$$

The b_{r_i} represents the encoded bitrate of the i'th representation where b_{r_n} represents the highest bitrate.

The algorithm assumes there is insufficient capacity for the next segment at the current quality level and switches to a lower quality rendition if

$$\mu < \gamma_d. \tag{3.6}$$

where γ_d is a threshold that determines the sensitivity of the algorithm to observed network congestion. We used the recommended value of 0.67. The adaptation will potentially switch down multiple quality levels to match the next request to an estimate of available bandwidth. The new bitrate is the highest bitrate for which

$$b_{r_i} < \mu b_c \tag{3.7}$$

where b_c represents the bitrate of the current segment.

The client supports a configured maximum playback buffer size. The client will not issue a new segment request once the buffer is full. As pointed out in recent work, this algorithm can be improved through more careful selection of γ_d and by computing μ based on multiple SFT samples [56]. However, for the research presented in this paper, we chose to use a simple published adaptation algorithm. The objective of the HAS experiment is to explore the impacts of different buffer management strategies on HAS performance.

3.2.2.2 VoIP performance metric

To assess the impact that high bandwidth applications might have on low bandwidth, latency sensitive flows, we establish a VoIP performance monitor between a server node and a cable modem. A 'talker' located on the VoIP server node (in Figure 3.4) sends a stream of UDP traffic in a manner similar to a G.711 session to a 'listener' located on the CM labeled VoIP client. We estimate the call quality (referred to as the R-value) using the technique described in [12]. The R-value ranges from 0 to 100. Toll quality requires an R-value of 70 or higher. The model assumes that two 10 ms 'chunks' of digitized voice data are sent in an IP packet (of size 238 bytes) every 0.020 seconds.

3.2.2.3 Web performance traffic generator and metric

The web traffic model is based on well understood behaviors of web browsing users [6, 14]. The ns2 simulation package provides extensive support for web traffic models. The model includes a set of web servers and web browsing users. A variable number of CMs are configured as web clients. Based on empirically derived distributions, these CMs periodically issue HTTP GET requests to randomly selected servers. The model is parameterized by user think times and web object sizes derived from heavy-tailed Pareto distributions.

We add an additional designated client and server application flow whose response times are captured to obtain the Web Response Time (WRT) metric. The designated client issues an HTTP GET and the server replies by transferring a configured amount of data (referred to as the WRT object size) for each iteration. The time between iterations is also configurable. The client computes the response time (referred to as a web response time or WRT) associated with the request. The WRT client repeatedly downloads the WRT object from the WRT server. Each iteration results in a WRT sample. At the end of the simulation we compute the average WRT statistic.

Chapter 4

Evaluation of Single-Queue Management Approaches

For the first phase of our study, we focus on evaluating the performance of modern delay-based AQM schemes in downstream DOCSIS 3.0 cable environment that employs a single queue bandwidth management approach. With the potential large deployment of AQM imminent for DOCSIS 3.0, our analysis provides timely feedback to the community concerning how delay-based AQM can manage bandwidth allocation fairness and application performance in realistic, single or bonded channel downstream DOCSIS 3.0 cable network scenarios.

We set to address the following questions that are key to the broadband network community concerning the ability of delay-based AQM in managing bandwidth for fairness and application performance:

1. How effectively do CoDel and PIE support fairness and application performance in realistic cable network scenarios?

2. Are there undesirable side effects when the AQM interacts with tiered service levels?

3. How effectively do the schemes isolate responsive traffic from unresponsive flows?

This chapter contains significant portion of the text as published in [38] and is organized as follows. In Section 4.1, we introduce the experimental setup and experiment definitions used for our analysis. In the next three sections, each section addresses one of the three questions posed above. First, we evaluate how effectively each AQM maintains throughput fairness and application performance in scenarios that do not involve service rate limits. Next, we analyze the interaction of service rate management with AQM. Finally, we evaluate the ability of AQM to protect responsive flows from unresponsive ones.

4.1 Experimental Setup and Experiment Definitions

We analyze several scenarios using varying numbers of downstream application flows including FTP, VoIP, web, and HAS video streams using the simulation network model given in Figure 3.4. The simulated client sides of these applications run on nodes attached to cable modems. The simulated servers run on nodes located outside the cable network. The cable network end point of each simulated flow is attached to a unique cable modem.

To model Internet path diversity, we set the propagation delays of the application server access links to different values. The range of uncongested path round trip times varies from 30 ms to 130 ms. This captures the inherent variability that subscriber flows might experience with respect to the physical distance between communication endpoints.

In the simulations that are presented, all data packets are 1500 bytes with the exception of VoIP packets which are 238 bytes. The bandwidth delay product of a round trip path consisting of heterogeneous links is properly computed as the bit rate of the bottleneck link multiplied by the path RTT. Assuming a data rate of 38 Mbps, fixed packet size of 1500 bytes and RTT of 130 ms, the bandwidth-delay product is 411 packets.

Maximum buffer capacities at the simulated routers and the CMTS are set large enough to ensure that, in the absence of AQM, all TCP senders can always have the full bandwidth delay product of unacknowledged packets in the pipeline. The maximum buffer capacity at the CMTS, referred to as the *QUEUE_CAPACITY* is set to 2048 packets. The maximum buffer capacity at all other routers is set to twice the *QUEUE_CAPACITY*. The *QUEUE_CAPACITY* is sufficiently large that it is no factor in packet drops for any schedulers employing AQM. For all but the HAS workloads the ns2 TCP simulation was configured as follows:

- TCP: TCP/Sack (ns2 TCP agent: Agent/TCP/Sack1).

- Maximum window/cwnd: 10000 segments.
- Initial window: 2 segments
- Delayed ACK mean interval: 100ms
- Maximum segment size: 1460 bytes
- Number of segment arrivals required to trigger an ACK: 2

For the HAS experiment, the ns2 Agent/TCP/FullTCP/Sack was used for both FTP and HAS. The TCP configuration used the parameters shown above.

In all simulations, we verified that the upstream channel was never congested. Upstream traffic was limited to ping replies, TCP acknowledgments, and HTTP GET requests.

The behavior of the AQM schedulers is obviously strongly dependent on their configuration parameters. The ARED drop algorithm is dependent on the target queue level, which in our simulation is derived from the *QUEUE_CAPACITY*. CoDel and PIE depend on the target delay.

The round trip latency experienced by packets under deficit round robin and drop tail scheduling is strongly dependent on the *QUEUE_CAPACITY*. The packet drop rate is also dependent, to a lesser degree, on *QUEUE_CAPACITY*. We employ the recommended default settings for all schedulers except as noted.

We map the three motivating questions posed in the introduction of the Chapter to three sets of experiments:

Set 1 We examine fairness and application performance resulting from the use of the AQM schemes under investigation.

Set 2 We explore implications of the use of different service tiers.

Set 3 We examine the ability of the schemes to deal with unresponsive flows.

Table 4.1 identifies five experiments that are presented in this paper. Set 1 consists of three different workloads while sets 2 and 3 each consist of a variation on the BASE workload of set 1. To better evaluate the effectiveness of the recently proposed schemes, CoDel and PIE, we compare their performance with three non-AQM techniques and one older approach to AQM.

Table 4.1: Experiment Definition

Set	Experiment	Summary
1	**BASE**	Varies the number of competing downstream FTP flows along with a low bitrate VoIP flow for different simulation runs; no service rate limit.
	HAS	Same as **BASE** except adding five HAS flows
	WEB	Runs 5 FTP flows with varied number of web flows ranging from 10–80
2	**TIER**	Same as **BASE** except: 1) Setting a service rate of 6 Mbps for all flows (referred to as Tier-1 flows); 2) Adding one additional competing FTP flow with a service rate of 12 Mbps. We refer to this flow as a Tier-2 flow.
3	**UDP**	Same as **BASE** except adding a 12 Mbps downstream UDP flow (starts at 500 seconds, stops at 1500 seconds). No service rate limit.

We compare modern delay-based AQMs (CoDel and PIE) against a number of queue managers including DT and ARED. DRR using per-flow queues are also included for reference purpose.

The *DT* queue manager employs a single queue with FCFS scheduling discipline that drops packets arriving to a full queue. The DT queue capacity is set to $QUEUE\_CAPACITY$ (2048) full size packets.

DT20 is also a FCFS drop-tail queue manager but configured in a way to be more comparable to CoDel and PIE. Its queue capacity is constrained to 65 full size packets which corresponds to a maximum queuing delay of approximately 20 ms, the target queuing delay for our CoDel implementation. The DT20 queue manager is used in the BASE experiment to assess the benefits that accrue from the additional complexity of CoDel and PIE.

The *DRR* queue manager employs one queue per flow and deficit round robin scheduling. Capacity of each queue is set to one-tenth of the capacity of the DT queue. Thus the total queue size is comparable. DRR is known to provide max-min fair allocation of channel capacity. Thus, results pertaining to throughput sharing and fairness are not of particular interest. Interest in packet loss rate and impact on latency sensitive traffic motivates the inclusion of DRR results.

The *ARED* queue manager is a simple extension to RED that adapts *maxp* such that the average queue level tracks a target queue level. We used the recommended target of $(minth + maxth)/2$. Configuration parameters are based on recommendations from [30]. The *minth* and *maxth* thresholds are the *QUEUE_CAPACITY* divided by 20 and by 2 respectively. The *control_interval* parameter is 0.50 seconds. The initial setting of *maxp* is 0.10.

The original CoDel description recommends a *target_delay* parameter set to 0.005 seconds [73]. We determined that a setting of 0.020 seconds provided more predictable results in our scenarios, especially in scenarios that involved a small number of high speed TCP flows. The *interval* parameter is set to 0.100 seconds. The original CoDel algorithm proactively drops packets at dequeue time. Due to potential implementation issues of head dropping raised by DOCSIS community, we implemented a tail drop variant of CoDel where packets are proactively dropped at enqueue time. Our implementation is based on an ns2 tail-drop variant of CoDel contributed by CableLabs. PIE is another delay-based AQM and is described in [77]. The D3.1 specification [9] provides recommended algorithm configuration settings for a CM running PIE. We use the default settings described in [77] for the algorithm run on CMTS. We set the *target_delay* parameter to 0.020 seconds and the T_{update} (frequency of update) parameter to 0.030 seconds. The *max_burst* parameter is 0.100 seconds. The internal burst control parameters, α and β, are set to 1.25 and 0.125 respectively.

4.2 Throughput Fairness and Application Performance

In this section we describe the results of the three experiments shown in Table 4.1 as set 1. The objective of the **BASE** experiment is to evaluate the effectiveness of the AQM schemes with respect to bandwidth utilization, throughput fairness and isolation of low bandwidth flows having low latency requirements.

Table 4.2: Experiment BASE TCP Throughput: Mean / Stddev / Sum (Mbps)

#FTPs	1	3	5	7	9	11
DRR	37.4 / 0.00 / 37.4	12.5 / 0.00 / 37.4	7.5 / 0.00 / 37.4	5.3 / 0.00 / 37.4	4.2 / 0.00 / 37.4	3.4 / 0.00 / 37.4
DT	37.4 / 0.00 / 37.4	12.5 / 1.86 / 37.4	7.5 / 0.86 / 37.4	5.3 / 0.69 / 37.4	4.2 / 0.86 / 37.4	3.4 / 0.69 / 37.4
DT20	36.3 / 0.00 / 36.3	12.3 / 1.60 / 37.0	7.4 / 1.24 / 37.1	5.3 / 1.39 / 37.1	4.1 / 1.62 / 37.0	3.4 / 1.62 / 36.9
ARED	37.4 / 0.00 / 37.4	12.5 / 1.37 / 37.4	7.5 / 0.99 / 37.4	5.3 / 0.86 / 37.4	4.2 / 0.86 / 37.4	3.4 / 0.81 / 37.4
CoDel	36.8 / 0.00 / 36.8	12.5 / 1.61 / 37.4	7.5 / 1.52 / 37.4	5.3 / 1.45 / 37.4	4.2 / 1.21 / 37.4	3.4 / 1.04 / 37.4
PIE	36.2 / 0.00 / 36.2	12.4 / 1.95 / 37.3	7.5 / 1.58 / 37.4	5.3 / 1.47 / 37.4	4.2 / 1.35 / 37.4	3.4 / 1.25 / 37.4

Table 4.3: Experiment BASE TCP Loss Rate (Percentage): Mean / Stddev

#FTPs	1	3	5	7	9	11
DRR	0.004 / 0.000	0.010 / 0.001	0.014 / 0.002	0.017 / 0.001	0.021 / 0.001	0.025 / 0.004
DT	0.017 / 0.000	0.016 / 0.017	0.020 / 0.025	0.021 / 0.042	0.016 / 0.022	0.015 / 0.032
DT20	0.007 / 0.000	0.050 / 0.005	0.097 / 0.020	0.141 / 0.029	0.189 / 0.020	0.245 / 0.018
ARED	0.002 / 0.000	0.013 / 0.003	0.022 / 0.001	0.035 / 0.004	0.048 / 0.004	0.063 / 0.005
CoDel	0.005 / 0.000	0.024 / 0.001	0.049 / 0.002	0.075 / 0.004	0.098 / 0.005	0.121 / 0.012
PIE	0.006 / 0.000	0.028 / 0.001	0.055 / 0.002	0.081 / 0.004	0.111 / 0.007	0.137 / 0.004

4.2.1 BASE simulation results

Experiment **BASE** employs a varying number FTP flows ranging from one to eleven along with a single VoIP flow. The flows compete for a total available bandwidth of 38 Mbps. The simulation time for each simulation is 2000 seconds. The VoIP performance metric flow starts at time 0.0 seconds. Each simulated FTP flow starts at a random time within the range of 0.0 to 2.0 seconds and has a unique uncongested path RTT. The path RTTs increase with the number of flows. For $i \in \{0, 1, .., 10\}$, the path RTT of flow$_i$ is $30+10i$ ms. Therefore, the average uncongested RTTs for the 6 load levels with $\{1, 3, .., 9, 11\}$ FTP flows are 30, 40, 50, 60, 70 and 80 ms.

Table 4.4: Experiment BASE TCP RTT: Mean / Stddev

#FTPs	1	3	5	7	9	11
DRR	0.073 / 0.000	0.182 / 0.006	0.291 / 0.011	0.399 / 0.016	0.508 / 0.020	0.617 / 0.025
DT	0.527 / 0.000	0.571 / 0.009	0.599 / 0.017	0.614 / 0.025	0.627 / 0.027	0.630 / 0.033
DT20	0.039 / 0.000	0.051 / 0.008	0.061 / 0.014	0.071 / 0.020	0.080 / 0.026	0.090 / 0.032
ARED	0.075 / 0.000	0.094 / 0.008	0.107 / 0.014	0.119 / 0.020	0.130 / 0.026	0.140 / 0.031
CoDel	0.041 / 0.000	0.057 / 0.008	0.068 / 0.014	0.079 / 0.020	0.089 / 0.026	0.100 / 0.031
PIE	0.039 / 0.000	0.054 / 0.008	0.066 / 0.014	0.077 / 0.020	0.088 / 0.026	0.098 / 0.031

4.2.1.1 Bandwidth Utilization

Table 4.2 summarizes the throughput achieved by the FTP flows. The total throughput was 37.4 Mbps in all but eight of the thirty-six simulations run. Throughput is marginally less for CoDel and PIE when the number of flows is small because the limited amount of buffering can cause the TCP pipeline to stall and, when combined with a small number of flows, loss of throughput results. Aggregate throughput for DT20 never reaches 37.4, demonstrating that the more complex queue management algorithms of CoDel and PIE do provide value. When a sufficient number of flows are active, all queue management mechanisms except DT20 produce the same aggregate bandwidth utilization.

The standard deviation reflects the magnitude of the variation in the throughput achieved by the competing flows. Since DRR is known to be max-min fair, the absence of variation is to be expected. For the other queue management techniques, the variation is proportional to the number of flows and inversely proportional to queue capacity. A more detailed analysis of the factors underlying the variation in observed throughput will be presented in the discussion of fairness.

Table 4.3 shows the mean and standard deviation of the loss rate experienced by the FTP flows for each simulation. The lost rates are expressed as a percentage and peak at only 0.245% (Approximately 25 packets out of every 10000 are lost.) We conclude that in all cases TCP is adapting very well to the available bandwidth and buffer space. Not surprisingly, marginal increases in loss rate are again proportional to the number of flows and inversely proportional to queue capacity. When the number of flows exceeds one, DT20 produces significantly higher loss rates than CoDel and PIE. As pointed out in [49], it is possible to tune the parameters of RED (or ARED in our case) to match the behavior of CoDel and PIE, as long as the channel's bit rate stays constant. However, in the presence of adaptive modulation and dynamic channel bonding in networks such as wireless or DOCSIS 3.1, ARED, as currently defined, would not be able to maintain a target queue delay.

Table 4.4 shows the average RTT experienced by the TCP packets of the FTP flows.

Table 4.5: Experiment BASE 9 FTP Flow 5 Run Average / 95% Confidence Interval

Scheme	RTT (seconds)	Loss Rate	Throughput (Mbps)
DRR	0.5077 / (0.5067, 0.5087)	0.0210 / (0.0201, 0.0218)	4.158 / (4.157, 4.158)
DT	0.6255 / (0.6229, 0.6281)	0.0206 / (0.0182, 0.0229)	4.158 / (4.157, 4.158)
DT20	0.0802 / (0.0801, 0.0802)	0.1904 / (0.1862, 0.1945)	4.111 / (4.110, 4.112)
ARED	0.1296 / (0.1295, 0.1297)	0.0484 / (0.0477, 0.0491)	4.157 / (4.157, 4.158)
CoDel	0.0895 / (0.0894, 0.0895)	0.0986 / (0.0977, 0.0995)	4.157 / (4.156, 4.158)
PIE	0.0876 / (0.0875, 0.0877)	0.1088 / (0.1079, 0.1097)	4.156 / (4.155, 4.157)

Table 4.6: Experiment BASE DT 5 Run Average / 95% Confidence Interval

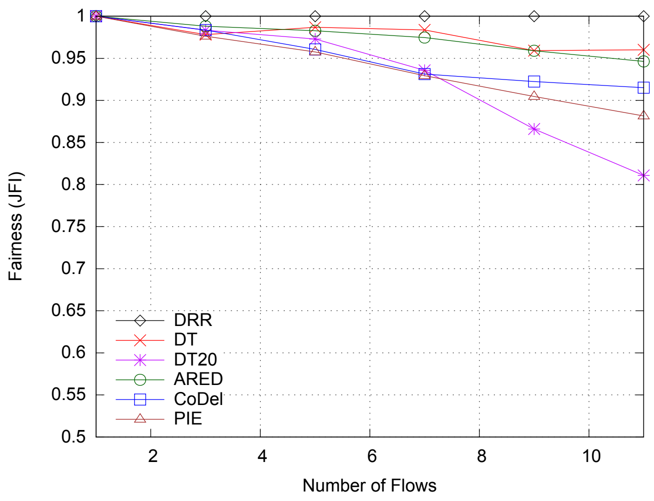
#FTPs	RTT (seconds)	Loss Rate	Throughput (Mbps)
1	0.5052 / (0.4802, 0.5302)	0.0172 / (0.0171, 0.0173)	37.397 / (37.379, 37.415)
3	0.5810 / (0.5682, 0.5938)	0.0178 / (0.0160, 0.0197)	12.473 / (12.470, 12.476)
5	0.5977 / (0.5904, 0.6049)	0.0190 / (0.0155, 0.0225)	7.484 / (7.482, 7.485)
7	0.6116 / (0.6036, 0.6196)	0.0231 / (0.0141, 0.0321)	5.346 / (5.345, 5.347)
9	0.6234 / (0.6174, 0.6294)	0.0195 / (0.0174, 0.0216)	4.158 / (4.157, 4.158)
11	0.6312 / (0.6285, 0.6340)	0.0224 / (0.0139, 0.0308)	3.402 / (3.402, 3.402)

The reported RTT is the sum of the inherent path delay and the queuing delay. For the 11 flow case, the mean inherent path delay is 80 ms. Therefore, mean queuing delay can be inferred by subtracting 80 ms from the values shown in the 11 flow column. We observe that PIE, CoDel, and DT20 maintain RTT values consistent with the target of 20 ms. ARED's queue latency ranges from 45 ms to 60 ms as the number of competing flows increase.

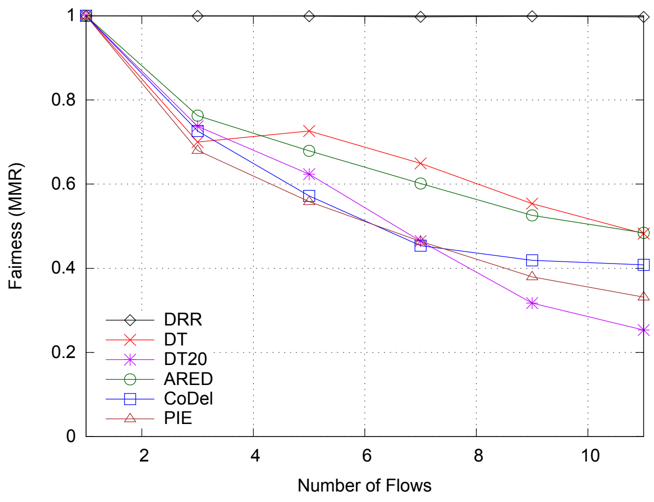
The DT results show a constantly high RTT regardless of the number of competing flows. Since the maximum allowed TCP window is configured to be larger than the queue capacity, the queue length is limited only by the $QUEUE\_CAPACITY$.

The DRR results reflect individual queue capacities that are 1/10 of the $QUEUE\_CAPACITY$. Therefore, when there are n flows with full queues, DRR queuing delay is the sum of the inherent path delay and $n \times QUEUE\_CAPACITY$ / 10.

We briefly consider the statistical accuracy of the results associated with the BASE experiment. We ran five independent replications with nine active TCP flows. The 95% confidence intervals for the average throughput, loss rate, and RTT are shown in Table 4.5. The widths of the confidence intervals were no more than 0.002 Mbps for throughput for



(a) Jain's Fairness Index



(b) Min-Max Ratio

Figure 4.1: Experiment BASE throughput fairness results.

all queue management schemes and no more than 0.2 ms in RTT for all three AQMs.

Given the synchronization and lockout issues that are associated with DT, we ran the DT simulation 5 times with different random seeds and involving different numbers of FTP flows. The results are given in Table 4.6.

Table 4.7: Throughput of 11 Individual FTP Flows

Scheme	Throughput (Mbps)											Mean/Stddev
Max-min fair	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	3.45	
DRR	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4 / 0.0
DT	3.8	2.5	3.9	2.8	3.3	2.5	3.0	4.0	3.4	3.8	4.4	3.4 / 0.6
DT20	6.8	5.6	4.6	3.9	3.2	2.8	2.4	2.2	1.9	1.8	1.7	3.4 / 1.7
ARED	5.0	4.6	4.0	3.8	3.4	3.0	3.0	2.8	2.8	2.6	2.4	3.4 / 0.8
CoDel	5.9	4.8	4.1	3.7	3.3	3.1	2.8	2.6	2.5	2.4	2.2	3.4 / 1.1
PIE	6.0	5.0	4.2	3.8	3.5	3.1	2.8	2.6	2.3	2.1	2.0	3.4 / 1.3
Uncongested RTT (ms)	30	40	50	60	70	80	90	100	110	120	130	

4.2.1.2 Throughput fairness in the BASE experiment

Figure 4.1 shows the throughput fairness results. These results are consistent with the standard deviation values previously discussed. The JFI results confirm that DRR is approximately max-min fair. The figure also shows that at high loads DT20 exhibits the highest level of unfair allocation. CoDel and PIE are significantly fairer but still worse than the others. Furthermore, it is apparent that unfairness is increasing with the number of flows for all queue management schemes other than DRR.

We ran additional ARED, CoDel and PIE simulations extending the number of competing flows to 50 and we observed the fairness continue to deteriorate. The JFI results for ARED, CoDel, and PIE with 50 competing flows were 0.75, 0.60, and 0.62 respectively.

Nevertheless, the underlying cause of the unfairness is not the increase in the number of flows *per se*. Instead, it is primarily a function of the disparity in uncongested RTTs. Table 4.7 shows the throughput of each of the individual FTP flows for an 11 flow experiment in one **BASE** simulation. The uncongested path RTTs for individual flows are given in the bottom row. It can be seen in this table that the achieved throughput for an individual flow under AQM management is inversely proportional (in a non-linear way) to its path RTT.

This is in agreement with the model proposed by Mathis [61]. TCP/RTT unfairness is in large measure to the fact that achieved throughput depends on factor of 1/ RTT. The result is that TCP flows having shorter RTT paths can starve longer path flows of buffer

Table 4.8: JFI of BASE with Same Path RTT

#FTPs	1	3	5	7	9	11
DRR	1.00	1.00	1.00	1.00	1.00	1.00
DT	1.00	0.96	0.97	0.95	0.99	0.98
DT20	1.00	1.00	1.00	1.00	1.00	1.00
ARED	1.00	1.00	1.00	1.00	1.00	1.00
CoDel	1.00	0.99	1.00	1.00	1.00	1.00
PIE	1.00	1.00	1.00	1.00	1.00	1.00

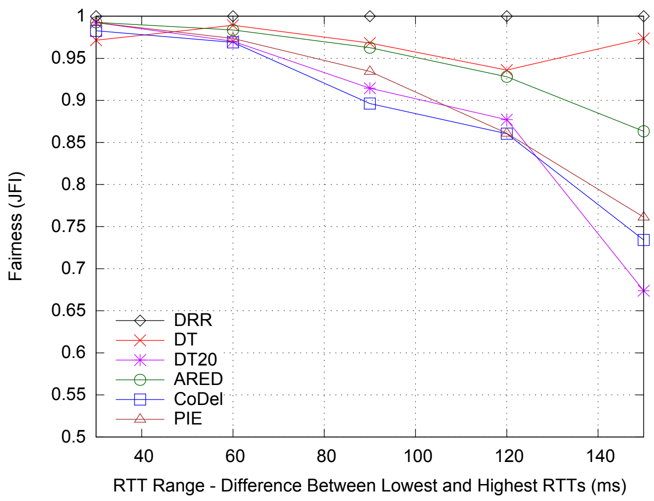
Table 4.9: MMR of BASE with Same Path RTT

#FTPs	1	3	5	7	9	11
DRR	1.00	1.00	1.00	1.00	1.00	1.00
DT	1.00	0.66	0.61	0.50	0.75	0.63
DT20	1.00	0.93	0.96	0.89	0.92	0.92
ARED	1.00	0.86	0.95	0.91	0.88	0.92
CoDel	1.00	0.83	0.95	0.94	0.95	0.93
PIE	1.00	0.92	0.92	0.96	0.92	0.92

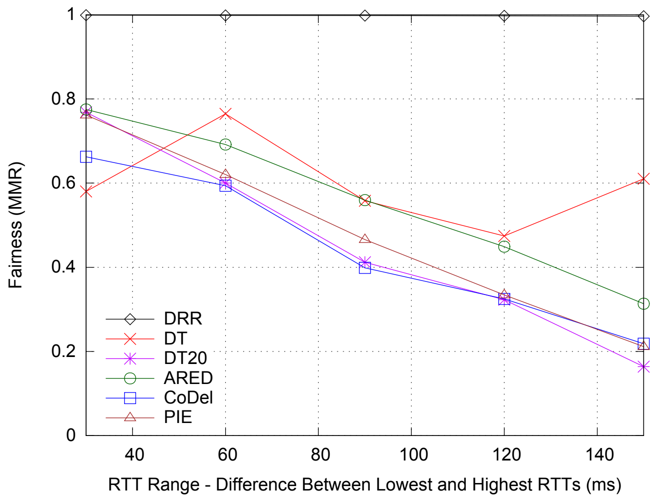
resource at a bottleneck. Each DRR flow has a dedicated amount of buffer resource, and this completely eliminates buffer starvation of one flow by another. The DT flows have a large *QUEUE_CAPACITY* shared buffer resource which mitigates the effect. Because of the parameters chosen, the ARED flows also had more effective buffer capacity (as evidenced by their longer RTTs) than did CoDel and PIE and therefore less unfairness due to buffer contention.

To confirm the conjecture that buffer contention was the root cause of the unfairness, we ran additional simulations (these results are not shown) with the *target_delay* parameter for CoDel and PIE set to a value commensurate with observed ARED queuing delays. The fairness results are consistent with those of DT and ARED.

To further explore the effect of path diversity on fairness, we ran a modified BASE simulation with all FTP flow paths set to the 80 ms mean RTT of the original BASE simulation. The fairness results are given in Table 4.8 and Table 4.9. Except for DT, the fairness results for all schemes improve across all workloads. In addition, the allocation is not shown to be less fair as the workload increases when RTT does not vary.



(a) Jain's Fairness Index



(b) Min-Max Ratio

Figure 4.2: BASE fairness results for 7 FTP flows with varied RTT ranges.

We then ran another modified BASE simulation where we kept the number of FTP flows fixed at 7 and varied the RTT range. The mean RTT was fixed at 100 ms and individual RTTs differed by *step* where $step \in \{5, 10, 15, 20, 25\}$. For example, when $step = 5$, the individual RTTs are $\{100, 95, 105, 90, 110, 85, 115\}$. Therefore, the total RTT range for each experiment is $6 \times step$.

The results as shown in Figure 4.2 clearly indicate that the AQMs under investigation become less fair as the range of RTTs widens. We conclude path RTT diversity has an

Table 4.10: Experiment BASE VoIP Isolation Performance (Mean Latency / Mean Loss Rate (Percentage) / R-Value)

#FTPs	1	3	5	7	9	11
DRR	0.045 / 0.000 / 93.1	0.046 / 0.000 / 93.1	0.046 / 0.000 / 93.1	0.046 / 0.000 / 93.1	0.046 / 0.000 / 93.1	0.047 / 0.000 / 93.1
DT	0.541 / 0.000 / 41.2	0.577 / 0.012 / 36.4	0.594 / 0.011 / 34.1	0.599 / 0.017 / 33.4	0.602 / 0.019 / 33.0	0.596 / 0.002 / 33.9
DT20	0.054 / 0.000 / 92.9	0.056 / 0.092 / 92.4	0.056 / 0.098 / 92.4	0.056 / 0.064 / 92.6	0.056 / 0.060 / 92.6	0.056 / 0.048 / 92.7
ARED	0.089 / 0.005 / 92.0	0.100 / 0.016 / 91.7	0.103 / 0.028 / 91.6	0.105 / 0.032 / 91.5	0.106 / 0.057 / 91.4	0.106 / 0.062 / 91.4
CoDel	0.056 / 0.001 / 92.9	0.062 / 0.059 / 92.4	0.064 / 0.080 / 92.3	0.065 / 0.110 / 92.2	0.065 / 0.163 / 91.9	0.066 / 0.231 / 91.6
PIE	0.054 / 0.008 / 92.9	0.060 / 0.029 / 92.6	0.062 / 0.054 / 92.5	0.063 / 0.071 / 92.4	0.064 / 0.131 / 92.1	0.064 / 0.160 / 92.0

adverse impact on throughput fairness under AQM.

4.2.1.3 Flow isolation

Having addressed the impact of AQM on overall throughput and fairness we now turn to flow isolation. We do this by analyzing the behavior of the simulated simplex VoIP stream that is sent in parallel with the FTP transfers of the BASE experiment. The stream is sent from a source outside the cable network to a sink attached to a simulated cable modem, and it is implemented as a simulated UDP flow in which two 10 ms 'chunks' of digitized voice data are sent every 20 ms in an IP packet of size 238 bytes.

The results are summarized in Table 4.10. The latency is average one-way latency experienced by the simulated VoIP packets. The mean loss rate experienced by the VoIP flow is of the same order of magnitude as those of the FTP flows which are shown in table Table 4.3. As would be expected, the VoIP loss rates under AQM are generally larger than the higher bandwidth FTP flows. It is also the case that the disparity between FTP and VoIP loss rates is consistently larger for CoDel than it is for PIE. Furthermore, DT20 yields lower loss rates and higher R-values than either CoDel or PIE across the board. The reasons for these disparities are yet to be fully understood.

The R-value for DT drops quickly from 41 to 33 as the number of competing FTP flows increases from 1 to 11. The main factor contributing to the poor R-values is the large latency experienced by the VoIP flow with DT managed queues. DT20, followed by CoDel and PIE, provides the lowest VoIP packet delays.

In additional simulations we found that the R-value remains reasonably high. When the number of competing FTP flows was increased from 50 to 90, the R-values varied from about 88 to 69 for ARED, CoDel, and PIE with slight differences.

4.2.2 HAS simulation results

We evaluate HAS video performance under different AQM schemes when the video streams are subject to competing greedy FTP flows. Experiment **HAS** involves five HAS flows along with a varied number of FTP flows for different simulation runs. As previously noted, TCP throughput is inversely proportional to path RTT. To make the HAS and FTP allocation results more comparable, the mean uncongested path RTTs of the HAS flows and of the FTP flows are set to 80 ms in all experiments. The five HAS flows have path RTT's of {80, 70, 90, 60, and 100 ms}. The path RTT of the first FTP flow is also 80 ms, and when additional flows are added, the path RTTs follow a similar pattern as that of the HAS flows.

For the results presented in this paper, we used the following configuration settings:

- Playback buffer capacity: 240 seconds.
- Segment size: 4 seconds.
- Set of bitrate representations (in Mbps): {0.5, 1.0, 2.0, 3.0, 4.0, 5.0}
- TCP specific configuration: we use the ns2 simulator's TCP/FullTCP/Sack model with similar settings to those used in the other experiments described in this paper.

During the startup phase, the client makes back-to-back requests until the buffer threshold (2 segments) is reached. The client then enters a steady state where it periodically (every segment time) issues a new segment request. As an example, consider an uncongested scenario involving a single HAS flow and no FTP flows. Once the HAS client reaches steady state, it requests a maximum quality segment (which is 2.5 Mbytes) every 4 seconds which corresponds to the maximum bit rate of 5.0 Mbps that HAS is configured to consume. Now suppose that the new segment arrives 1.07 seconds after the request is sent. Then ε is

$4/1.07 = 3.74$ and thus the estimated available capacity is 5.0×3.74 or 18.7 Mbps. This capacity estimate underestimates the actual available capacity (38 Mbps) by more than a factor of two. The estimate is low in part because the TCP stack resets the TCP congestion window after the application has been idle for one retransmission timeout period (which happens each on/off cycle). This is a known issue with rate-limited applications [39, 68, 24].

Table 4.11: Experiment HAS Throughput Efficiency

#FTPs	DRR	DT	ARED	CoDel	PIE
1	88%	76%	88%	83%	84%
3	96%	96%	96%	94%	94%
5	96%	96%	96%	95%	95%
7	96%	96%	96%	95%	95%
9	96%	96%	96%	95%	96%
11	96%	96%	96%	95%	96%

Table 4.12: FTP and HAS Average Throughput and Allocation Bias

(a) FTP Flow Average Throughput

#FTPs	DRR	DT	ARED	CoDel	PIE
1	8.59 (0.5%)	8.60 (78.2%)	8.59 (2.8%)	8.17 (23.8%)	7.89 (12.4%)
3	6.22 (36.6%)	7.07 (55.3%)	5.90 (29.9%)	5.48 (23.4%)	5.13 (15.4%)
5	4.36 (19.8%)	5.40 (48.1%)	4.32 (18.7%)	4.10 (13.4%)	3.98 (10.2%)
7	3.37 (11.0%)	4.34 (42.9%)	3.46 (14.0%)	3.34 (10.7%)	3.26 (7.8%)
9	2.82 (8.4%)	3.34 (28.4%)	2.88 (10.7%)	2.79 (8.1%)	2.77 (6.8%)
11	2.37 (4.2%)	2.89 (26.9%)	2.46 (8.1%)	2.43 (7.5%)	2.45 (7.6%)

(b) HAS Flow Average Throughput

#FTPs	DRR	DT	ARED	CoDel	PIE
1	4.99 (-0.2%)	4.07 (-15.6%)	4.95 (-0.9%)	4.69 (-6.3%)	4.83 (-3.5%)
3	3.55 (-22.0%)	3.04 (-33.2%)	3.73 (-17.9%)	3.82 (-14.0%)	4.03 (-9.3%)
5	2.92 (-19.8%)	1.89 (-48.1%)	2.96 (-18.7%)	3.13 (-13.4%)	3.24 (-10.2%)
7	2.57 (-15.4%)	1.21 (-60.0%)	2.44 (-19.6%)	2.57 (-14.9%)	2.69 (-10.9%)
9	2.21 (-15.0%)	1.27 (-51.2%)	2.10 (-19.3%)	2.21 (-14.6%)	2.28 (-12.3%)
11	2.07 (-9.2%)	0.93 (-59.1%)	1.87 (-17.9%)	1.88 (-16.6%)	1.89 (-16.8%)

Table 4.11 shows the percentage utilization of the downstream channel for each

experiment. The results show that in all cases except for simulations involving 5 HAS flows and 1 FTP flow, the channel utilization is at least 94%. CoDel and PIE exhibited a utilization 1-2% lower than that of ARED and DRR.

It is challenging to create meaningful throughput fairness comparisons between the HAS and FTP workloads for several reasons. First, the maximum application layer demand of each HAS application is self-limited to 5.0 Mbps while each FTP application has unbounded application layer demand. Second, when congestion occurs HAS responds by further reducing its application layer demand while FTP does not. Furthermore, when few FTPs are active, the applications can interact with AQM in such a way that considerably less than full utilization of the 38 Mbps downstream channel is actually achieved. Finally, there are unwanted dynamics between HAS and TCP control algorithms. Nevertheless, comparing the average FTP flow allocation and the average HAS flow application to what one would expect from a max-min fair allocation does provide insight in these dynamics.

The average throughput for FTP and HAS applications as well as the *applicationBias* is shown in Table 4.12. For all buffer management schemes, the throughput achieved by both FTP and HAS decreased as the number of competing flows increased.

The *applicationBias* metric is the percentage above or below the max-min fair allocation of the bandwidth that was actually consumed in the experiment (as opposed to the nominal channel bandwidth of 38 Mbps.) The expected max-min fair allocation for a HAS flow is based on a demand of 5 Mbps, which is the highest HAS encoded video bitrate. Using the low congestion scenario (1 FTP flow and 5 HAS flows) as an example, Table 4.11 indicates that the consumed bandwidth under DRR is 88.3% of the 38 Mbps. The max-min fair allocation of the 33.5 Mbps actually consumed under DRR is 5 Mbps for HAS and 8.55 Mbps for FTP. Consumed bandwidth under CoDel was significantly lower and max-min fair allocation is 5 Mbps for HAS and 6.6 Mbps for FTP. For the CoDel results the actual FTP throughput was 8.17 Mbps and the average HAS throughput was 4.69 Mbps. The HAS *applicationBias* is $(4.69 - 5.00)/5.00$ or -6.3%. For the scenario involving 5 FTP and 5 HAS flows, under CoDel, the bandwidth actually consumed is 36.1 Mbps. So the max-min fair allocation is 3.6 Mbps for all flows. In this case, for CoDel, the HAS *applicationBias* is $(3.1 - 3.6)/3.6$ or -13%.

In all cases the allocation favored FTP. The results indicate that only DRR consistently improved the HAS allocation outcome as the network became congested. For PIE and CoDel, the allocation outcome actually became worse (i.e., *applicationBias* experienced by HAS flows grew further negative) as the load increased.

Table 4.13: HAS Performance Metrics (videoPlaybackRate / adaptationRate)

#FTPs	1	5	11
DRR	4.7 / 9	2.8 / 5	2.0 / 3
DT	3.7 / 32	1.9 / 56	0.9 / 53
ARED	4.7 / 35	2.9 / 91	1.8 / 150
CODEL	4.4 / 50	3.0 / 165	1.8 / 216
PIE	4.6 / 44	3.1 / 101	1.8 / 154

Table 4.13 indicates the HAS performance metrics. The *rebufferTime* results (not shown) was 0 in all simulations. The results show that AQM significantly improves HAS performance compared to DT based on the *videoPlaybackRate* metric. Among the three AQMs, the *videoPlaybackRate* results were very similar. The results suggest that CoDel leads to higher HAS *adaptationRate* than the other AQMs. We have observed that PIE is less tolerant of bursts and consequently during times of heavy congestion does maintain the target queue latency generally more reliably than CoDel. We conjecture that differences in the AQM's burst tolerance behavior contribute to the observed difference with the *adaptationRate* metric. However, it is likely that changing AQM configuration parameters would alter the behavior. Therefore, further study is required before we can draw any conclusions.

The absence of rebuffering events is largely due to relatively static workloads (a small number of established TCP flows) and to a large playback buffer capacity. In additional simulations not included in the paper, we run the scenarios using a version of HAS that does not adapt. Instead, the client always requests highest quality segment (we refer to this algorithm as *max-always*). The *videoPlaybackRate* in all cases is 5.0 Mbps which represents the highest quality. In this scenario, all buffer management schemes exhibit frequent stalls. The CoDel and PIE *rebufferTime* results were similar with the percentage of time spent stalled ranging from 0% (with 5 HAS and 1 FTP flow) to 53% of the time (with five HAS and 11 FTP flows).

The work in [39] shows that HAS is prone to a 'spiral down' effect where it is possible for a HAS flow to be 'beaten down' by competing greedy TCP application flows. In this situation, the HAS flow is likely to remain at the lowest bitrate quality level to oscillate across the lowest quality levels. In our results, the *applicationBias* metric shows evidence that aspects of this effect are present. Primarily due to relatively well behaved dynamics, the system does not suffer the full spiral-down effects observed in [39].

4.2.3 WEB simulation results

The experiment **WEB** explores the impact of the different queue management schemes in scenarios that involve a mix of downloads of web objects of varying size and FTP traffic. Five greedy FTP flows are run in parallel with a varied number of simulated web flows. For five FTPs, the uncongested RTTs are 30, 40, 50, 60, 70 (as in the BASE). The number of web flows ranges from 10 to 80, and for the web flows, the uncongested RTTs cycle in the same pattern as: 30, 40, 50, 60, 70, 30, 40, 50, 60, 70, ...

The parameters of the web traffic model, derived from [26], include user and session attributes designed to reproduce the variability inherent in web traffic. The web sessions start randomly within the first second of the simulation. The *inter-page time* is set to a Pareto distribution with mean and shape equal to 5 and 2.0 respectively. The *objects per page* is set to Pareto distribution with mean and shape equal to 4 and 1.2 respectively. The *inter-object time* is set to Pareto distribution with mean and shape equal to 0.5 and 1.5 respectively. The *object size* in KB is set to Pareto distribution with mean and shape equal to 20 and 1.5. Both of these sets of parameters produce distributions with infinite variance. Therefore, it is difficult to obtain repeatable estimates of statistics such as mean response time with simulations of practical duration.

It should be noted that web models such as the one from ns2 no longer properly characterize the traffic generated by browsing commercial web sites. The objects at such sites typically contain many embedded links to different advertising sites that are not co-located with the primary page. The objects loaded via the embedded links commonly contain a mix of text, images, audio, video, and possibly even additional embedded links. Modeling traffic of this type is well beyond the scope of this paper. The ns2 web model is designed to model browsing of a non-commercial document collection in which the user may download citations, abstracts, or full text of scientific papers of various sizes. While the web traffic models may have changed over time, the inherent on/off, heavy tailed nature of the aggregate web traffic should still hold.

Therefore, as described in Section 3.2.2.3, the WRT metric reflects the response time experienced by an additional designated web client in requesting and receiving a web object of a particular size. By default, the WRT object size is 20 KB, and the think time before the next request is issued is 10 seconds. The range of uncongested path RTTs for both the web browsing flows and the competing FTP flows is between 30 and 70 ms. The designated flow for which the WRT metric is computed has an uncongested path RTT of 50 ms which is the mean of the RTTs of the competing traffic.

Figure 4.3 shows the average throughput achieved by FTP flows as the number of web browsing flows increases from 10 to 80. Although not shown, the mean bandwidth demand of a single web flow is 0.13 Mbps and the total observed bandwidth consumption of the web flows ranges from 1.3 to 10.3 Mbps for all schedulers except DT. In a max-min fair allocation of the available 38 Mbps, each web flow should receive its entire demand and the 5 FTP flows should divide the remainder. It can be observed in Figure 4.3 that this is approximately the case for all schedulers except DT. For example, when there are 80 web flows, the web flows consume an average of 0.129 Mbps and the 5 FTP flows consume approximately 5.2 Mbps each. This indicates that the 36.3 Mbps of consumed bandwidth is fairly divided between the web and FTP traffic.

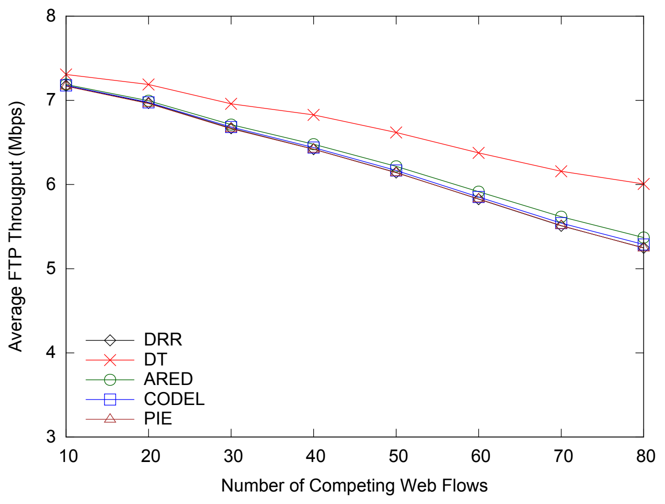


Figure 4.3: WEB Simulation Average FTP Throughput

Table 4.14: Web Response Time / Standard Deviation (seconds)

#Web Flows	10	20	30	40	50	60	70	80
DRR	0.173 / 0.062	0.172 / 0.060	0.175 / 0.059	0.175 / 0.060	0.178 / 0.060	0.178 / 0.061	0.181 / 0.062	0.178 / 0.062
DT	2.414 / 0.622	2.416 / 0.626	0.737 / 0.407	2.439 / 0.596	1.116 / 0.872	0.662 / 0.266	1.023 / 0.780	0.905 / 0.702
ARED	0.213 / 0.079	0.209 / 0.079	0.206 / 0.070	0.208 / 0.081	0.208 / 0.097	0.209 / 0.093	0.210 / 0.097	0.208 / 0.095
CoDel	0.173 / 0.076	0.174 / 0.098	0.171 / 0.059	0.174 / 0.077	0.174 / 0.085	0.173 / 0.077	0.173 / 0.075	0.176 / 0.091
PIE	0.173 / 0.096	0.173 / 0.097	0.172 / 0.095	0.174 / 0.116	0.179 / 0.117	0.176 / 0.103	0.171 / 0.103	0.177 / 0.111

The DT allocation of slightly more than 6 Mbps to the FTP flows demonstrates the well known result that with unmanaged queues greedy flows prevent less greedy ones from obtaining their fair shares.

Table 4.14 provides the WRT results for the experiment. As shown, AQM schemes are able to support a range of low bandwidth web flows with consistently low web response time. TCP slow-start requires multiple RTTs before the 20 KBytes of data is successfully received by the WRT client. As the number of web flows increase, there is significant increase in WRT only with the DT queue manager.

In a separate WEB experiment, we explore how the different queue management techniques might impact the web response times as the WRT object size increases. The scenario involves 5 FTP flows that compete with 50 web flows based on the web traffic model. Unlike the previous WRT experiment, we vary the WRT web object size from 32 KBytes to 2048 KBytes. The WRT results are shown in Figure 4.4. As expected, the web

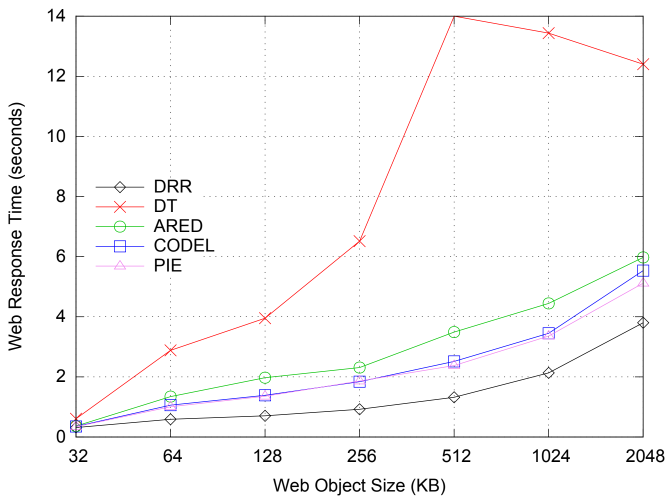


Figure 4.4: WEB Simulation WRT Results for Varied Object Sizes

response time increases along with the web object size. The best mean response time was obtained by DRR, and the worst by DT. DT response was also quite sensitive to stochastic effects. In one test, five independent replications of the DT simulation produced mean response times of 6.8, 8.9, 8.2, 6.7, and 6.2 seconds. In contrast, the mean response times produced by ARED, CoDel, and PIE had small variation.

4.3 Implications of Service Rate Management

The objective of the **TIER** experiments is to explore the impact of the different queue management schemes on subscribers having different maximum service rates. A varying number of Tier-1 FTP flows, each having a service rate of 6 Mbps, compete with a single Tier-2 FTP flow having a service rate of 12 Mbps on a single downstream channel. The uncongested path RTT of the Tier-2 flow is set to 80 ms which is also the mean uncongested RTT of the Tier-1 flows. Individual RTTs of adjacent Tier-1 flows vary according to the pattern {80, 70, 90, 60, 100, . . . }.

Figure 4.5 illustrates the observed behavior of nine Tier-1 flows competing with the single Tier-2 flow under CoDel scheduling. The Tier-1 FTP flows run for the entire simulation time of 2000 seconds, but the Tier-2 flow is active only from time 500 to 1500

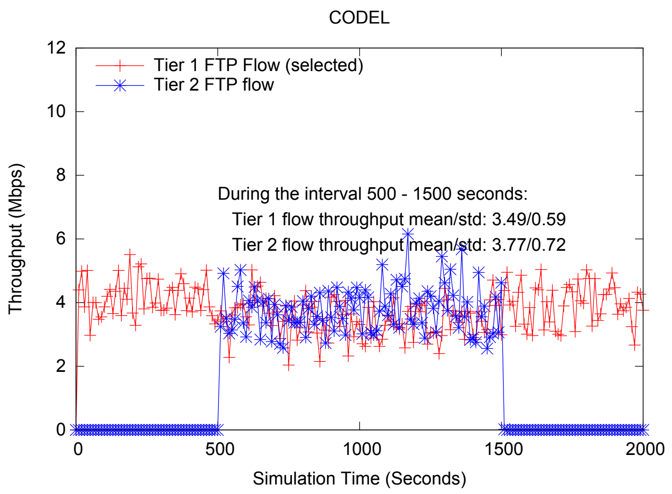


Figure 4.5: TIER simulation results with CoDel (1 Tier-2 flow competing with 9 Tier-1 flows)

seconds. Each data point plotted represents measured throughput during an interval of ten seconds. The Tier-1 flow with 80 ms uncongested RTT is shown in red, and the Tier-2 flow in blue. During the time period when the Tier-2 flow is active, the throughput of the selected Tier-1 flow and Tier-2 flow are 3.49 Mbps and 3.77 Mbps respectively, and the Tier-2 flow displays higher variability. The weighted max-min fair allocation is 3.45 Mbps for the Tier-1 flows and 6.9 Mbps for the Tier-2 flow. The unweighted max-min fair allocation is 3.8 Mbps for all flows. These results illustrate that single queue AQM will not support differentiated service that intends to allocate more bandwidth for flows at higher tiers. Whether or not this outcome is desirable, depends on one's perspective. From the perspective of the Tier-2 customer, this is clearly an undesirable outcome and conversely so for the Tier-1 customer.

The second **TIER** experiment is designed to show the allocation to the different tiers achieved by the different queue management schemes. A varying number of Tier-1 flows compete with a single Tier-2 flow for the entire simulation.

Table 4.15 summarizes the average allocation achieved by all Tier-1 flows and the allocation achieved by the Tier-2 flow. Simulations involving 1 and 3 competing Tier-1 flows are not shown as these reflect uncongested conditions, and the expected tiered allocations

Table 4.15: TIER Flow Average Throughput Allocation (Mbps)

# Tier-1 FTP Flows	5		7		9		11	
Tier (Mbps)	6	12	6	12	6	12	6	12
Max-min fair	6.0	8.0	4.75	4.75	3.8	3.8	3.17	3.17
Weighted max-min fair	5.4	10.8	4.2	8.4	3.45	6.9	2.9	5.8
DRR	5.9	7.8	4.7	4.7	3.7	3.7	3.1	3.1
DT	5.9	8.1	4.5	6.0	3.5	5.5	3.1	3.3
ARED	5.7	8.7	4.7	4.8	3.7	3.8	3.1	3.0
CoDel	5.8	7.9	4.6	5.4	3.7	3.7	3.1	2.9
PIE	5.7	8.4	4.6	4.8	3.7	3.7	3.1	2.8

are achieved in all cases.

As expected, DRR provides an unweighted max-min fair allocation. DT consistently allocates more bandwidth to the Tier-2 flow. The AQMs support tiered allocation in a load dependent way. As the load increases, the *average* flow bandwidth allocation by various AQMs for both tiers converges approximately to *unweighted* max-min fair allocation. In the case of 11 Tier-1 flows, all AQMs allocate less bandwidth than its max-min fair share to the Tier-2 flow. This anomaly is due to the non-linear nature of the RTT penalty. Recall that the Tier-1 throughput is the average of flows having uncongested RTTs ranging from 30 to 130 ms while the Tier-2 throughput is that of a single flow.

Table 4.16: TIER Flow Average Throughput Allocation (Mbps) with 4-channel Bonding Group

# Tier-1 FTP Flows	5		7		9		11	
Tier (Mbps)	24	48	24	48	24	48	24	48
Max-min fair	24	32	19	19	15.2	15.2	12.7	12.7
DRR	23.7	31.7	18.7	18.8	15.0	15.0	12.5	12.5
DT	23.1	34.6	18.0	24.2	14.3	20.9	12.0	17.5
ARED	23.2	34.0	18.4	21.4	15.2	13.5	12.5	12.5
CoDel	23.1	31.3	18.5	20.0	15.1	14.1	12.6	11.5
PIE	22.9	33.1	18.5	20.2	15.1	14.3	12.6	10.9

DOCSIS 3.0 supports channel bonding to provide fat data pipes. A bonding group as deployed in the field often involves 4 to 8 downstream channels bonded together. Such fat pipes can support higher service rates and tiers. We run **TIER** with a 4-channel bonding

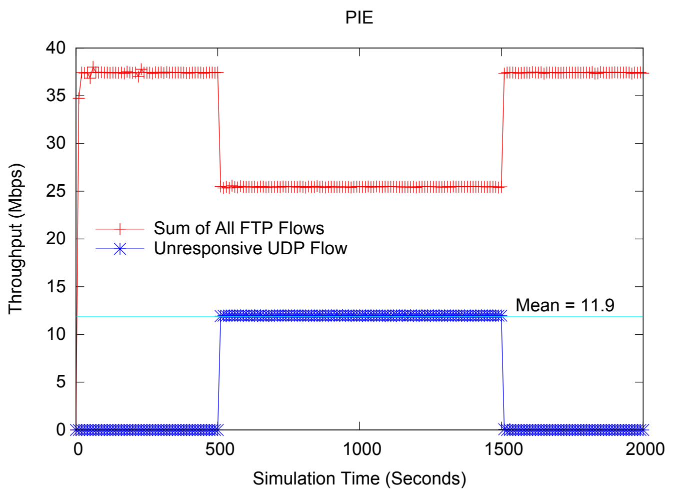


Figure 4.6: Experiment UDP Results with PIE

group. We set the Tier-1 service rate to 24 Mbps and the Tier-2 service rate to 48 Mbps. To adjust for the increased bandwidth-delay product, we increase the $QUEUE\_CAPACITY$ and TCP window size by a factor of 4. Again for the uncongested scenarios (i.e., those involving 1 and 3 competing Tier-1 flows), our simulation results show each flow receives a throughput equal to its service rate. For congested scenarios, Table 4.16 shows the results of this simulation. The results show that our previous conclusions for the single channel case still hold for the channel bonding case.

4.4 Management of Unresponsive Flows

To evaluate how effectively the AQM schemes under investigation manage unresponsive flows, experiment **UDP** runs a varying number of FTP flows along with an unresponsive UDP CBR flow. Figure 4.6 illustrates the results using PIE queue management in a **UDP** simulation in which 9 FTP flows compete with an unresponsive UDP CBR flow for a total available bandwidth of 38 Mbps. The UDP flow was configured to send 1500 byte packets at a constant rate of 12 Mbps during the time period 500 to 1500 seconds of the 2000 second simulation. No service rate limitations were imposed. As shown, the unresponsive flow was allocated almost its total demand of 12 Mbps by PIE and the FTP flows were allocated the remaining bandwidth. For all other schemes, only DRR correctly allocates equal share (about 3.7 Mbps) to all flows during the time period when the UDP flow was active. The results for DT, ARED, and CoDel were very similar to the PIE result shown in Figure 4.6.

We conclude that single queue management schemes including the AQM schemes evaluated here are not effective in preventing unresponsive flows from consuming more bandwidth than their fair shares during the time of congestion.

4.5 Summary of the Results

The focus of the first phase has been upon the effects of downstream queue management in a DOCSIS cable environment. We have considered workloads consisting of FTP, HAS, web browsing, and unresponsive UDP traffic. Our analysis also considered scenarios with and without imposition of service rate limits. We explored fairness and application performance obtained with four single queue packet scheduling disciplines: FCFS drop tail (DT), Adaptive RED (ARED), CoDel, and PIE. We also included results obtained with multi-queue DRR as a reference point for our analysis.

The clearest result is the undesirability of the use of drop tail queue management in conjunction with a large unmanaged buffer pool. The effects of the resulting bufferbloat included poor flow isolation, very high latency, and unpredictable allocation of bandwidth to competing flows.

Our ARED results differ from CoDel and PIE in large part due to our choice of parameter settings. For all AQMs, we attempted to use recommended settings. Nevertheless, it is possible to configure ARED to behave in a manner similar to CoDel and PIE when channel capacity is known *a priori* and not subject to changes due to factors such as adaptive modulation or dynamic changes to channel bonding groups.

Below we summarize the conclusions of this phase by addressing the three motivating questions raised at the beginning of the chapter:

1. *How effective are CoDel and PIE in realistic cable network scenarios?*

The schemes are quite effective at maintaining the target queue delay. This in turn provides isolation between responsive flows in the scenarios we explored. Both CoDel and PIE exhibited similar results in all of the experiments, and even as the level of congestion became very high, both were able to provide strong flow isolation.

All schemes except DRR are shown to not allocate bandwidth fairly in the presence of heterogeneous RTTs. However, the principal cause of the unfairness is TCP's well known bias toward short-RTT connections. Nevertheless, delay-based AQMs such as CoDel and PIE are particularly sensitive to the RTT induced unfairness issue when compared to other queue management schemes that allow sustained large queues. When a buffer bloated queuing point causes 100's of milliseconds of packet delay, the fairness outcome of competing TCP flows that might have uncongested path delays of a fraction of this are less sensitive to TCP RTT-based unfairness. When delay-based AQM removes 'bufferbloated' queues, differences in uncongested path delays have more of an impact on TCP performance. Nevertheless, it should be noted that throughput fairness is not a primary objective of delay-based AQMs.

2. *Are there undesirable side effects when AQM interacts with tiered service levels?*

   We observed a range of behaviors among the AQM schemes. When different service tiers compete for bandwidth during periods of the congestion, the impact upon tiered service is dependent upon the severity of the congestion. When the system is not loaded or slightly loaded, the AQMs do allocate more bandwidth to the higher service tier flows. As the load continues to increase, we see that the allocations for different tier flows tend to converge. As congestion increases, a higher tier flow can be more aggressively targeted by an AQM scheme because the regulator allows its packets to reach the queue at a higher rate than those of the lower tier.

3. *How effectively do the schemes isolate responsive traffic from unresponsive flows?*

   An inherent problem with the single queue schemes is that they cannot effectively manage unresponsive flows. This is a huge challenge network operators have to address.

Our results of this phase suggest that there is a trade-off involved with single queue management mechanisms between the goals of managing fairness and managing application performance. Current practice tends to favor application performance over fairness. This may not be acceptable in emerging converged broadcast networks where a larger percentage of subscribers view content from video-on-demand providers located in the Internet. For example, our results indicate that content providers that physically locate content closer to the subscriber might be allocated a larger share of access network resources (during periods of congestion) compared to a content provider that does not locate content at the access network. While this result is common to all buffer management methods, delay based AQM exacerbates this issue in certain circumstances primarily because bufferbloat effects served to mask the issue. Moving forward, we address some of these problems with a multiple queue solution that can also meet the constraints of CMTS vendors.

Chapter 5

Approximate Fair Scheduling Using Adaptive Bandwidth Binning

In this chapter, we present the design, implementation, and performance evaluation of a low complexity approach to managing bandwidth that addresses the combined problem of fairness and bufferbloat for DOCSIS. We present detailed performance study of the scheduler in both D3.0 and D3.1 environments. The chapter is organized as follows. In Section 5.1, we present our adaptive bandwidth binning scheme that implements the approximate bandwidth management model introduced in Chapter 3. In Section 5.2, we introduce the experimental setup and experiment definitions used for the analysis of this phase. The last two sections provide our results and analysis targeted for single tier environment and multi-tier environment respectively.

5.1 Adaptive Bandwidth Binning and Implementation

As described earlier, adaptive bandwidth binning (ABB) quantizes flows based on their observed bandwidth consumption normalized by flow weights. Flows classified to the same bin will be scheduled based on FCFS. Each bin is implemented as a single aggregate queue which is managed by CoDel. Bins are then scheduled by weighted DRR. The weight of each bin is set to the sum of weights of the flows classified into this bin.

We maintain each flow's bandwidth consumption using the standard exponential moving average formula. The current level of consumption for each flow is calculated with the flow's consumption rate samples obtained from each past reclassification interval. The α parameter for the formula is set to 0.4.

The scheme uses k bins for the scheduling of n flows. The system design parameters are:

- Number of bins (k): k should be a small fixed number that is usually much smaller than the number (n) of flows
- reclassification interval (τ): this is the time interval of periodical flow reclassification. We recommend it is set at time scale of seconds

Parameters for CoDel such as target delay and control interval are those given in [73, 72].

We justify the scheme as follows. A CMTS already monitors service flows [64]. So the bandwidth consumption aspect of the scheme does not add extra complexity. Tiering is naturally handled by having control decisions based on recent consumption normalized by flow weights. Binning is used to avoid per-flow queueing. The approach has low complexity.

5.1.1 Data Structures

Our implementation is for ns-2 and follows the structure of SFQ-CoDel [70]. It uses a few data structures: bins and flows, described in Listing 5.1.

Listing 5.1: Data Structures

```
// bin structure
struct bindesc {
  queue *q_;             // underlying FIFO queue
  int index;             // bin ID (array index)
  double bi;             // Value of bi (consumption threshold)
  double weight;         // bin weight

  // Dynamic state used by CoDel algorithm
  double first_above_time_; // when we went (or will go) continuously above
    target for interval
  double drop_next_;    // time to drop next packet (or when  dropped last)
  int count_;           // how many drops we've done since the last time we
    entered dropping state.
  int dropping_;        // = 1 if in dropping state.
  int deficit_;         // for rounding on bytes

  int newflag;          // newly active bin
  int on_sched_;
  bindesc* prev;
  bindesc* next;
};

// flow state
struct flowstat {
  double bw;  // bandwidth consumed
  int binid;  // id of bin the flow is classified to
  int weight; // flow weight
  int alloced;  // internal flag
};

// main data structure
struct ABBCoDelAQM {
  bindesc bin_[k];      // k bins
  bindesc* binsched_;   // active bin list
  flowstat flow[n];     // array of n flows

  // Static state (user supplied parameters)
  double target_;       // target queue size (in time, same units as clock)
  double interval_;     // width of moving time window over which to compute min

  // Dynamic state used by CoDel algorithm
  double first_above_time_; // when we went (or will go) continuously above
```

```
    target for interval
  double drop_next_;     // time to drop next packet (or when we dropped last)
  int count_;            // how many drops we've done since the last time
                         // we entered dropping state.
  int dropping_;         // = 1 if in dropping state.

  int maxpacket_;    // largest packet we've seen so far
  int mtu_max_;

  int curlen_;       // the total occupancy of all bins in packets
  int maxbinid_;     // id of bin with the most pkts
  int quantum_;      // for rounding by bytes
};
```

The main data structure *ABBCoDelAQM* contains an array of a fixed number of k bins (sub-queues) used to queue incoming packets from n different flows. As in (S)FQ-CoDel, there is a global limit on the number of packets the bins can hold but not a limit per bin. In other words, bins share a common buffer of certain size.

All bins are managed by CoDel separately. Hence the state variables for the CoDel state machines for the individual bins are given in the *bindesc* structure. But all CoDel state machines maintain a common CoDel delay target and use the same CoDel control interval given in the main data structure. Note the packet queueing latency is measured in CoDel by the time a packet enters one of the bins until the time the packet leaves that bin. So even though the packet is in a bin that is not currently being scheduled, it continues to accrue its queueing latency. When the packet is scheduled, this queueing latency is used to compare with the common delay target.

As defined in the structure, a dynamically adjusted weight variable is associated with each bin. A few other state variables are useful for scheduling. For example, the deficit and the number bytes queued. Each bin $i\,(1 \leq i \leq k)$ uses a single value b_i that defines the bandwidth range of the bin. Without losing generality, let $b_i < b_{i+1}$ so that bins are arranged in increasing order in terms of consumption levels. Let $b_0 = 0$ and $b_k = C$, where C is the channel capacity. The range of bandwidth consumption level for bin_i is then $(b_{i-1}, b_i]$.

The main data structure $ABBCoDelAQM$ also contains a varying number of n flows, which is defined by *flowstat* structure. The structure contains the information that tracks the bandwidth consumed by the flow, the ID of the bin the flow is classified to, and the static flow weight. The ID of the bin associated with a flow may change from one control interval to another when the flow is remapped.

5.1.2 Operations

The ABB scheme asynchronously performs two operations: *enqueue* and *dequeue*. The enqueue operation handles the task of inserting a newly arrived packet into one of the bins (sub-queues). The dequeue operation deals with picking (scheduling) which packet for service when channel is available for transmission. The code for the two operations is almost identical to that of SFQ-CoDel except weights are accounted for inter-bin WDRR scheduling. The scheme also periodically performs a third operation that *optimizes* the system, *reclassifies* flows into bins, and adjust bin weights to maintain fairness. The period that controls the third operation is given by the reclassification interval.

5.1.2.1 The enqueue operation

The pseudocode for *enqueue*, which is almost identical to that of SFQ-CoDel except weights are considered, is given in Listing 5.2:

Listing 5.2: Enqueue routine

```
// on arrival of packet 'pkt'
Enqueue (pkt):
  if no more buffer space left:
    maxbinid_ = FindLongestBin()
    tail drop a packet from bin_[maxbinid_]
  i = ExtractFlow(pkt)          // flow ID
  bid = flow[i].binid
  enqueue(bin_[bid], pkt)

  // add new active bin to the schedule
  if binsched_ == NULL:
    // new active bin is the only active one
    binsched_ = bin_[bid]
    bin_[bid].newflag = 1
    bin_[bid].on_sched_ = 1
    bin_[bid].deficit_ = quantum_ * bin_[bid].weight
  else if bin_[bid].on_sched_ == 0:
    // if bin was not on the schedule,
    InsertBin(binsched_, bin_[bid]) // insert new bin to schedule before old
    bins but after other new bins
    bin_[bid].newflag = 1
    bin_[bid].on_sched_ = 1
    bin_[bid].deficit_ = quantum_ * bin_[bid].weight
```

The *enqueue* operation takes $O(1)$ time because, given only a limited number bins in use, finding longest bin in terms of most packets queued and inserting newly active bin to the schedule both take constant time. Finding flow ID by hashing packet header [87] and adding a packet to the tail of a queue also take constant time.

5.1.2.2 The dequeue operation

The pseudo code for *dequeue*, almost identical to that of SFQ-CoDel except bin weight is accounted for, is given in Listing 5.3:

Listing 5.3: Enqueue routine

```
// when channel becomes idle or is free
Dequeue():
  do:
    // find a ready bin
    b = binsched_
    while b is empty:
      nb = b.next
      remove b     // b is taken off from binsched_
      b = nb
    while b.deficit_ <= 0:
      b.deficit_ += quantum_ * b.weight
      b = b.next
      while b is empty:
        b = b.next
    binsched_ = b

    b.newflag = 0

    // set up to do CoDel for bin b
    first_above_time_ = b.first_above_time_
    drop_next_ = b.drop_next_
    count_ = b.count_
    dropping_ = b.dropping_

    pkt = ... // run CoDel on bin b to dequeue a packet

  while pkt == NULL

  // save CoDel state for bin b
  b.count_ = count_
  b.first_above_time_ = first_above_time_
  b.drop_next_ = drop_next_
  b.dropping_ = dropping_

  b.deficit_ -= pkt.size
```

The do-while loop in the dequeue code takes O(1) time to finish due to the limited number of bins in use. The do-while is guaranteed to terminate because the *dodequeue*, one of the routines used by CoDel, safeguards against dropping all packets to leave the channel idle. Thus, the *dequeue* routine overall takes $O(1)$ time to finish.

5.1.2.3 The Optimize operation

The pseudo code for the optimizing routine is given in Listing 5.4. When a timer set with the reclassification interval τ is triggered periodically, the *Optimize* routine is run to reclassify flows into different bins should the consumption levels of the flows change. The routine also assigns consumption range for each bin and adjusts bin weights according to the weights of the flows classified to the bins.

Listing 5.4: Optimize routine

```
// Let $C$ be the channel capacity
// Let $k$ be the number of bins
// Let $totalWeight$ be the sum of the weights of all flows

Optimize():
  // remove inactive flows from threshold calc
  for each flow f:
    if f.bw is less than (such as 50 kbps):
      f.alloced = 1
    else:
      f.alloced = 0

  // calc bin bandwidth consumption threshold
  for i = 1 .. k-1:
    lw = 0.0     // sum of weights of unalloc'ed flows
    lbw = C      // bandwidth left for alloc
    for each flow f:
      if f.alloced:
        lbw -= f.bw
      else:
        lw += f.weight
    bin_[i].bi = lbw / lw
    for each flow f:
      if f.bw / f.weight <= bin_[i].bi:
        f.alloced = 1
bin_[k].bi = C

// calc bin weights and reclassify flows
for i = 1 .. k:
  bin_[i].weight = 0
for each flow f:
  for i = 1 .. k:
    if f.bw / f.weight <= bin_[i].bi
      f.binid = i
      bin_[i].weight += f.weight
      break
```

In the pseudocode, to calculate the bandwidth consumption level thresholds for the bins, we use a method that is loosely based on finding the max-min fair allocation for aggregate groups of flows. We call this method the max-min fair *breakpoint* guided approach, which we will describe in detail below. With this approach, a series of max-min fair breakpoints are calculated in the increasing order and the $b_i, (1 \leq i \leq k)$ are set accordingly to those increasing max-min fair breakpoints. In the pseudocode, to avoid skewing the calculation for the first max-min fair breakpoint, as a preprocessing step, we also try to remove any inactive (or low consumption) flows from the calculation. The suggested threshold for low consumption is 50 or 100 kbps, which should cover low consumption flows such as VoIP.

We use a simple (and unrealistic) example to illustrate the approach and what max-min fair breakpoints are. Based on known individual flow demands in terms of bandwidth request, the process we often use to calculate the max-min fair share bandwidth generates a series of estimated max-min fair share values which we call max-min fair breakpoints. For example, for a capacity of 40 Mbps and six flows having demands of 4, 6, 7, 8, 9, and 10 Mbps respectively, the first max-min fair breakpoint is $40/6 = 6.7$. With this breakpoint, the demands of the first two flows are completely satisfied with excess of 2.7 and 0.7 respectively, which must be evenly distributed among the four remaining flows. This leads to the calculation of next max-min fair breakpoint. In this case, the remaining capacity is 30 Mbps for the rest of 4 flows. The next breakpoint is thus $30/4 = 7.5$ and the demand of the third flow is satisfied with excess of 0.5. Again this leads to the third max-min fair breakpoint calculation $23/3 = 7.7$, which is below either demand of the rest of the 3 flows. So the rest of the 3 flows can only receive maximum 7.7 Mbps bandwidth without excess. The calculation thus stops with a max-min fair share of 7.7.

Since in a real system there is no way to know the flow demands *a priori* or measure the demands, our algorithm takes the actual achieved throughput of the flows at the moment in lieu of the flow demands for the max-min fair breakpoint calculation. This is not a problem as the goal is to separate flows based on their consumption levels. The breakpoints calculated this way turned out to naturally separate flows into a few consumption levels.

Using a data sample extracted from one of the simulations, we now show how the bin thresholds are set based on the breakpoints calculated at that time. For this example, we ran 6 flows sharing a channel capacity of 38 Mbps. The bandwidth consumption was 4.0, 6.0, 6.4, 7.4, 6.4, and 6.9 for the 6 flows respectively. The total consumption is 37.1 Mbps, which should always be lower than the channel capacity. This is where the calculation based on actual consumption differs from the calculation based on known flow demands, whose total can well exceed the channel capacity. Based on the actual consumption data, the first max-min fair breakpoint is $38/6 = 6.3$. With this breakpoint, the first two flows are said to be "allocated" with excess. This leads to the calculation of our next max-min fair breakpoint of $28/4 = 7$. With the new breakpoint, the rest of the flows except the fourth flow are said to be "allocated" again with excess. So the calculation continues with the third breakpoint being $(28 - 6.4 - 6.4 - 6.9)/1 = 10.3$. With this breakpoint, all flows are said to be "allocated" and the calculation ends. Based on these breakpoints, if using 3 bins, the bin thresholds will be set to 6.3, 7, and 38 respectively. The threshold for the last bin is always set to the capacity irrespective of if we have a max-min fair breakpoint for the last bin. This is to ensure that any flows that are not classified to other bins will be classified to the last bin. In the Optimize routine, to deal with inactive or very low bandwidth flows that may skew the calculation for the first breakpoint resulting in the need of more bins, we explicitly exclude these flows from the calculation.

With this approach, flows that consume more bandwidth will be isolated to higher bins from flows consuming less bandwidth. The *Optimize* routine further adjusts the bin weights depending on how many flows (and their weights) are classified to each bin. In the above example where flows have equal weight of 1, the weights for the 3 bins are set to 2, 3, and 1 respectively. The bin weights support the use of a weighted inter-bin DRR scheduler to ensure weighted fair allocation among the sets of flows classified to different bins. With the proper bin weight settings, flows in different bins will likely be forced to equalize their bandwidth consumption during next interval to maintain fairness. For example, if the set of 2 flows in bin 1 together consume 1/3 of the total bandwidth and the set of 3 flows in bin 2 consume one half, the fourth flow classified into bin 3 cannot consume more than 1/6 of the total bandwidth, reducing its consumption from 7.4 to 6.3.

Although the operation runs with a complexity of $O(n)$, where n is the number of flows, it runs fairly infrequently only for each reclassification interval, which is at the time scale of seconds. Since the work needed for scheduling is defined as number of steps needed per packet [87], this operation does not affect the amount of work needed, which remains at $O(1)$.

5.1.3 An Illustrative Example

In this example, we run six UDP CBR flows 1-6 sourced at 4, 6, 7, 9, 11, 13 Mbps respectively sharing a channel capacity of 38 Mbps. The total demand of the flows is 50 Mbps. The expected max-min fair allocation is thus $\{4, 6, 7, 7, 7, 7\}$.

For the ABB scheme to approximate fair allocation, in general, flows 1-3 with lower demand (and certainly lower consumption) should be classified to lower bins to be guaranteed their fair share whereas flows 4-6 with higher demand (and possibly higher consumption) should be classified to higher bins so that they will not be able to unfairly grab a lion share of the bandwidth. Due to the possible fluctuation in consumption over time, it is possible for middle ranged flows such as flows 4 and 5 to jump from one bin to another. But overall such flows should stay in lower bin for more time if it consumes relatively less bandwidth or vice versa.

Figure 5.1 shows a snapshot (between 500 and 532 seconds) of the flows residing in different bins over time under ABB scheme with 2 bins. Each sub-plot is for one flow. The smooth curves in the figure show the flow consumption rate changes over time. The square waves show a flow being moved from one bin to another. For example, at 501 second, flows 1-3 were classified into bin 1. Flows 4-6 were classified to bin 2. Therefore the weights for the bins were set to 3 and 3 respectively. At second 503, flow 3 consumed 6.6 Mbps and was reclassified to bin 2. But flow 4 was reclassified to bin 1 since its consumption at the moment dropped to 6.1 Mbps.

As clearly seen in this case, the ABB scheme was able to correctly classify flows into proper bins. The two flows that consumed below the estimated fair share were always classified into bin 1 to be guaranteed its share. The demand of flow #3 was right around the estimated fair share. It is shown the flow mostly resided in bin 1 and occasionally went to bin 2. Flow #4 spent more of its time in bin 2 than in bin 1 because of its slightly higher consumption than flow #3. Flows #5 and #6 always resided in bin 2 due to their higher demands and consumptions.

At the end of 1000 second simulation, the throughput of the flows under the approach with different number of bins are given in Table 5.1, where ABB with k bins are specified as ABB-k. We also calculated the per-flow bin switch rates as the ratio of total number bin switches over the total maximum possible bin switches. The bin switch rates of the scheme using 2–5 bins are 0.23, 0.43, 0.42, and 0.42 respectively.

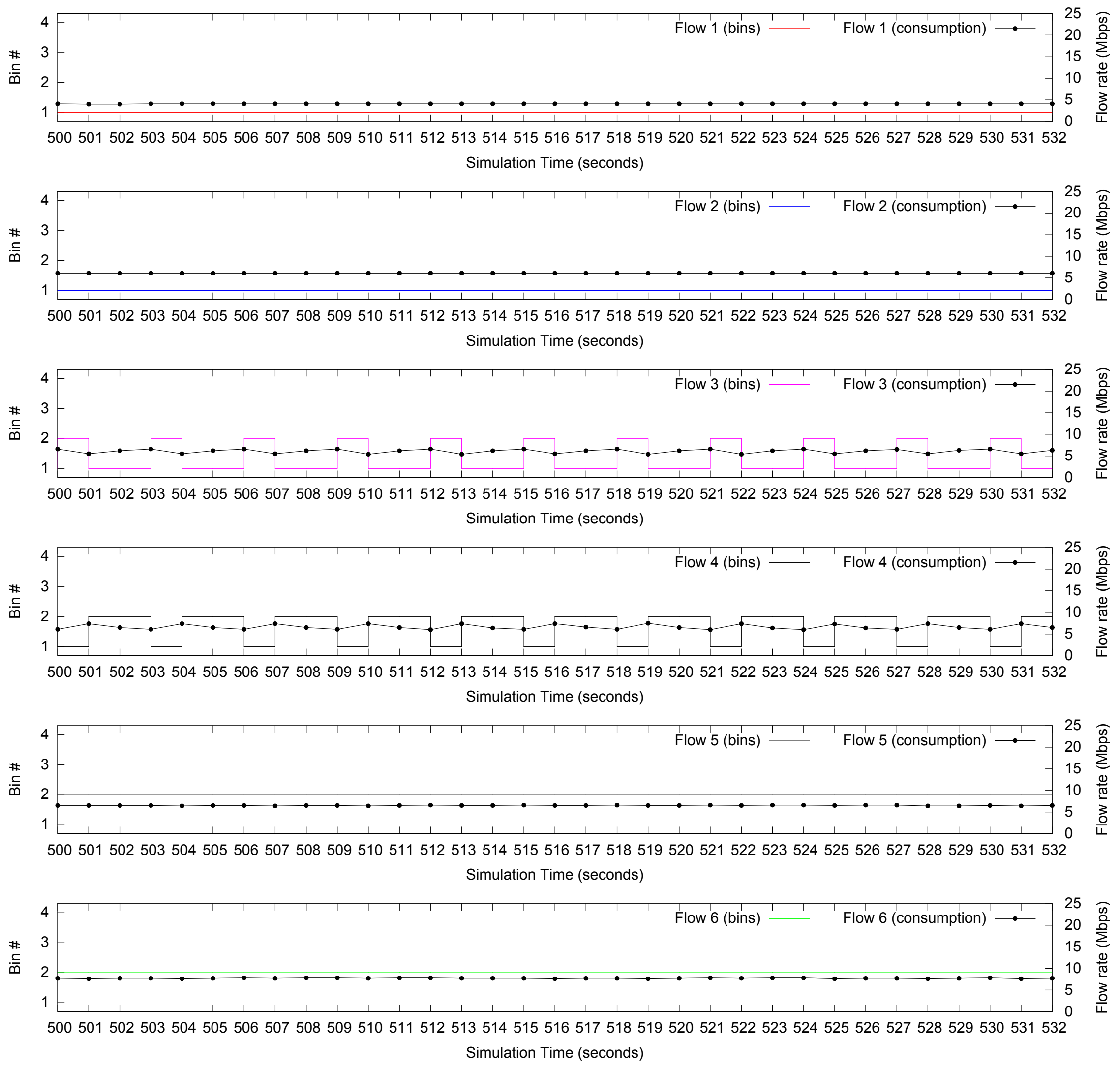


Figure 5.1: Example simulation showing how the ABB scheme works (UDP case)

Table 5.1: UDP Flow Throughput (Mbps)

Flow	#1	#2	#3	#4	#5	#6	JFI
CBR Rate	4	6	7	9	11	13	
Max-min fair	4	6	7	7	7	7	
ABB-1	2.9	4.3	5.0	6.5	7.9	9.4	0.937
ABB-2	3.9	5.9	5.9	6.5	6.4	7.5	0.995
ABB-3	3.8	5.8	5.9	6.5	6.9	7.2	0.996
ABB-4	3.8	5.8	5.9	6.5	6.9	7.2	0.996
ABB-5	3.8	5.8	5.9	6.5	6.9	7.2	0.996

5.1.4 Addressing the Packet Reordering Issue

For the most part, packets from a flow classified to a bin will be queued and sent in chronicle order under the ABB scheme. However, since periodically flows are reclassified, some flows unavoidably will be moved to new bins and unsent old packets from such flows may be left queued in their previous bins. Therefore old and new packets from the same flow in different bins can be sent out of order. The number of packet left behind should be limited in proportion to the target delay of CoDel *if* the queue length is well under control. Then under the same assumption, if the target delay is 20 ms and the reclassification interval is set to 1 second, the impact of potential packet reordering is estimated to be no more than $0.02/1 = 2\%$. Although the potential impact does not look very significant, the issue likely has some negative effect over the TCP performance.

The same packet reordering problem exists with other schedulers such as SFQ when periodical perturbation of the hash function is performed. The problem was not raised and addressed in [63]. However the SFQ code in Linux kernel rehashes the old packets whenever perturbation occurs. The rehashing involves moving packet from one bin to another and inserting packets in chronicle order. This can be quite costly. The FQ-CoDel code in the Linux kernel, on the other hand, does not do perturbation and therefore avoids the need of rehashing. This of course comes at the expense of possibly less throughput for flows that collide. The odds of such collision are discussed in [36] and potential collisions are accepted.

We address the out-of-order issue with a low complexity solution that completely eliminates the need of moving packets around. Figure 5.2 shows the basic idea of the solution using three bins as an example. First packets are scheduled normally from the bins (Figure 5.2(a)). At each flow reclassification, if any flow switches bins, all bins are 'plugged' (Figure 5.2(b)). The 'plugs' serve the purpose of dividing the already-queued old packets from the incoming new packets that continue to be queued normally. The old packets from all bins will continue to be scheduled normally until one of the plugs reaches the head of its bin. From this time on, any bin that has its plug at its head is temporarily suspended

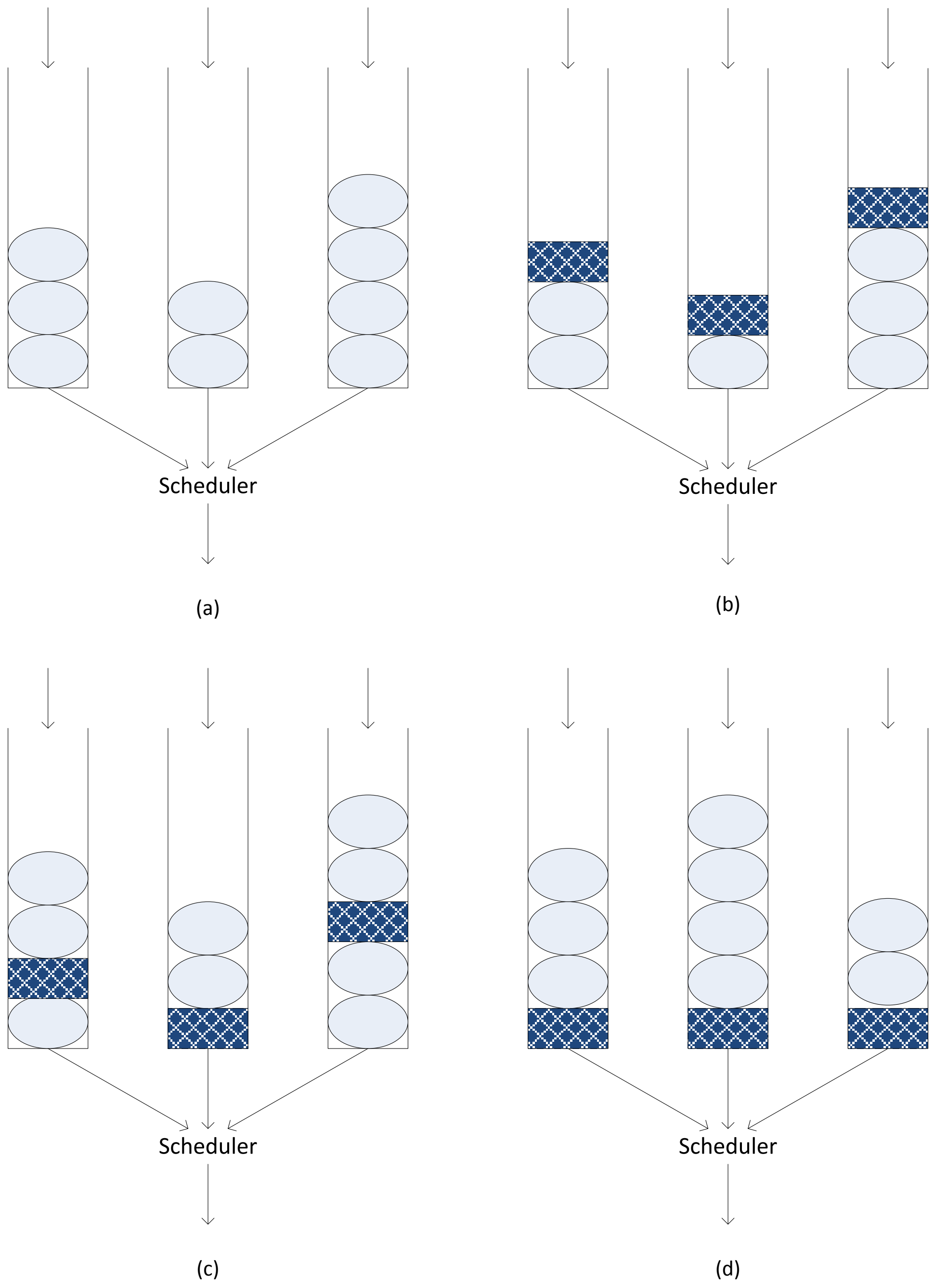


Figure 5.2: Addressing packet reordering issue by bin plugging.

from scheduling (Figure 5.2(c)). In other words, the plug stops the new packets on the bin from being sent ahead of any old packets that are yet to be sent. This ensures no packets will be sent out of order. The temporary scheduling suspension of a bin is until the plugs of all bins reach their bin heads (Figure 5.2(d)). At this time, all plugs are removed and the bins return to normality to be scheduled normally again.

Between the time the first plug hits its bin head and the bin has at least one packet after the plug and the time all plugs hit their bin heads, there is a period of time during which normal bin scheduling is temporarily disrupted. We call such period of time *disruption* time. For example, in the above illustrative example with six UDP flows, the total disruption time was 84.7 seconds through 1000 second simulation for the approach using 3 bins. This is equivalent to about 8.5% overall disruption. With unresponsive UDP flows, CoDel's burst tolerance can lead to higher queue level and imbalanced bins. We will see, with TCP flows, the disruption is consistent with our earlier estimation and well under control.

5.2 Experimental Setup and Experiment Definition

The experimental setup for this phase is similar to that of the first phase as given in Section 4.1. We carry out all our studies in a simulated DOCSIS 3.x cable environment. The simulation network model was illustrated in Figure 3.4. For the downstream scheduling component in the figure, we compare a number of scheduling schemes including single-queue CoDel and SFQ-CoDel to our ABB scheme. We use DRR-CoDel as a reference. DRR-CoDel is a scheme that combines DRR scheduler with CoDel managing each of the DRR's per-flow queues. The DRR scheduler in use is weighted unless otherwise specified. The single-queue CoDel is adapted from an open source tail-drop variant of CoDel for DOCSIS. SFQ-CoDel is adapted from [70] to work in our DOCSIS simulation framework. As indicated earlier, SFQ-CoDel is an implementation of FQ-CoDel [36] for ns2. ABB is properly implemented in our simulation framework. The pseudo code was described in previous section.

Our analysis covers several scenarios using varying numbers of downstream application flows including FTP, VoIP, exponential on/off, web, and HAS video streams. The simulated client sides of these applications run on nodes attached to cable modems. The simulated servers run on nodes located outside the cable network. The cable network end point of each simulated flow is attached to a unique cable modem.

To model Internet path diversity, the propagation delays of the application server access links are experimental parameters. In each set of experiments, we maintain consistent path RTTs (either using same RTTs for all flows or maintaining same average RTTs among classes of flows). The maximum buffer size setting can have significant impact on results for queue managers other than delay-based AQMs. But for delay-based AQM, the max buffer size setting is less significant as long as the buffer size is large enough to accommodate occasional bursts. In the following we set the max buffer size at CMTS for delay-based AQM to allow maximum 150 ms queueing delay should the queue reach its full capacity.

In the simulations presented, all data packets are 1500 bytes with the exception of VoIP packets which are 238 bytes. The bandwidth delay product of a round trip path consisting of heterogeneous links is properly computed as the bit rate of the bottleneck link multiplied by the path RTT. Assuming fixed packet size of 1500 bytes and RTT of 100 ms, the bandwidth product is 8333 packets and 350 packets for D3.1 data rate of 1 Gbps and D3.0 data rate of 42 Mbps respectively. We set the max buffer size to be 32768 for 1 Gbps channel and 4096 for 42 Mbps channel at all routers. Maximum buffer capacities at the simulated routers are set so large to ensure that, in the absence of AQM, all TCP senders can always have the full bandwidth delay product of unacknowledged packets in the pipeline. The maximum buffer capacity at CMTS is set to 11875 packets for 1 Gbps D3.1 channel or 457 packets for 42 (38) Mbps D3.0 channel. The settings are equivalent to a 150 ms max standing queue.

The ns2 TCP simulation was configured as follows:

- TCP: TCP/Sack (ns2 TCP agent: Agent/TCP/Sack1).
- Maximum window/cwnd: 10000 segments (for D3.0 channel) or 65536 segments (for D3.1 channel).

- Initial window: 2 segments
- Delayed ACK mean interval: 100 ms
- Maximum segment size: 1460 bytes
- Number of segment arrivals required to trigger an ACK: 2

For workload involving HAS, the ns2 Agent/TCP/FullTCP/Sack was used for HAS. The rest of the TCP configuration used the same parameters shown above.

We decompose our analysis into two components. One set of experiments, shown in Table 5.2, is designed for each component. First, we set our analysis in a single-tier environment with the current D3.0 channel speed. The focus is to answer how well the scheme is able to address TCP RTT unfairness and to provide flow isolation and good latency property. We also explore the dynamic behavior of the scheme adapting to changing workload and the possible impact of such adaption. Second, we set the analysis in a multi-tier environment with 1 Gbps D3.1 channel speed. We focus on answering how well the scheme is able to provide weighted fairness based on flow tiers and to provide flow isolation with respect to unresponsive UDP flows. We also explore how well the scheme is able to operate with more realistic traffic workload involving FTP, HAS, and web flows.

Table 5.2: Experiment Definition

Set	**Experiment**	**Summary**
Single Tier	**BASE**	For different simulation runs, varies the number of competing downstream FTP flows along with a low bitrate VoIP flow. Used to assess performance over different workloads.
	ME	Runs a mix of small mice and large elephant flows to assess the stability of the approach.
Multi Tier	**TIER**	Similar to **BASE** except that adding a number of Tier2 and Tier4 flows for different runs. Used to assess performance in a tiered environment.
	UDP	Runs 11 FTP flows per tier at Tier1, Tier2, and Tier4. Each tier also runs one UDP/CBR flow sourced at 50 Mbps. Used to assess UDP flow isolation performance.
	APP	Runs 10 FTP, 30 web, and 60 HAS flows with different configurations to assess application performance.

For CoDel, we configure the target delay to be 0.020 seconds. The original CoDel recommends a target delay of 0.005 seconds. We determined that a setting of 0.020 seconds provided more predictable results in our scenarios, especially in scenarios that involved a small number of high speed TCP flows. The interval parameter is set to 0.100 seconds. Following the recommendation, SFQ-CoDel is configured to use 1024 bins as in the original code. For ABB, we ran the experiments with a small number of bins (2, 3, and 4 in most cases). The reclassification interval is set to 1 second.

We present the results from our simulation studies in the next two sections. Each section corresponds to each set of experiments in Table 5.2.

5.3 Results and Analysis for Single Tier Environment

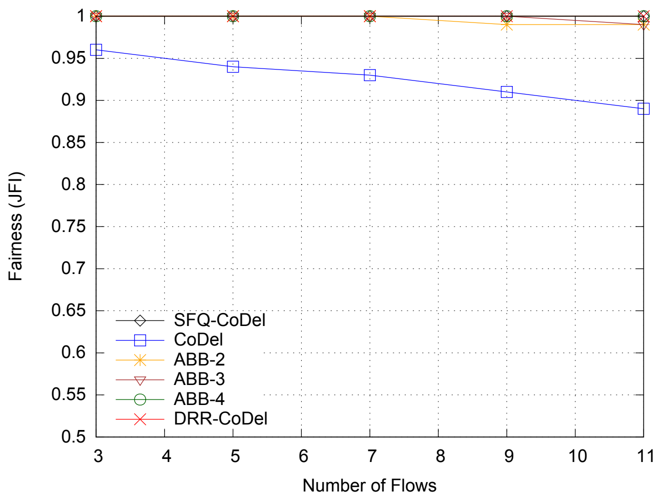
In this section we present the results of the single tier set of experiments shown in Table 5.2. The objective of this set of experiments is to evaluate the effectiveness of the ABB scheme in providing fairness and delay performance.

The ABB results are obtained from using 2, 3, and 4 bins respectively. They are referred to as ABB-2, ABB-3, and ABB-4 below. Unless otherwise specified, the reclassification interval is set to 1 second. For this set of simulations, the simulation time is set to 2000 seconds unless otherwise specified.

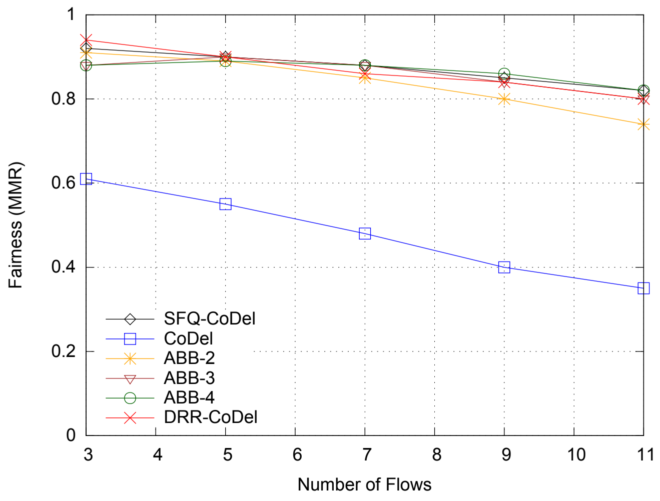
5.3.1 BASE Simulation Results

Experiment **BASE** employs a varying number of FTP flows ranging from three to eleven along with a single low bit rate VoIP flow. The flows compete for a total available bandwidth of 38 Mbps. The VoIP performance metric flow starts at time 0.0 seconds. Each simulated FTP flow starts at a random time within the range of 0.0 to 2.0 seconds and has a unique uncongested path RTT. All simulation runs use the same average uncongested RTTs of 80 ms. But different FTP flows are set to different uncongested path RTTs according to this pattern $\{80, 70, 90, 60, 100, \dots\}$.

Figure 5.3 shows the throughput fairness results from five simulation runs with different number of FTP flows. In general, the fairness from all schemes in comparison is



(a) Jain's Fairness Index

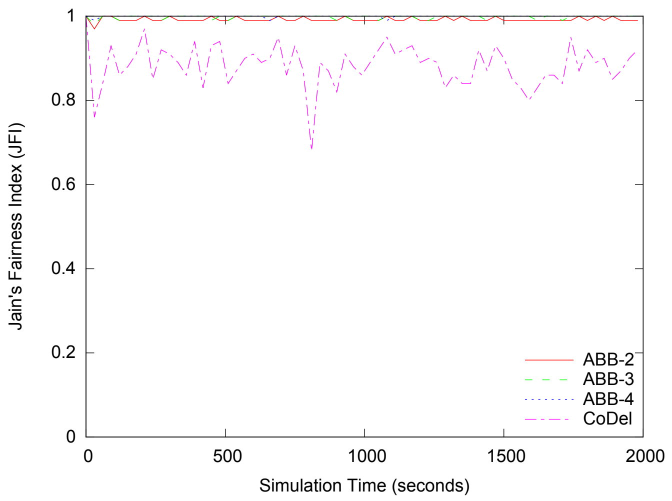


(b) Min-Max Ratio

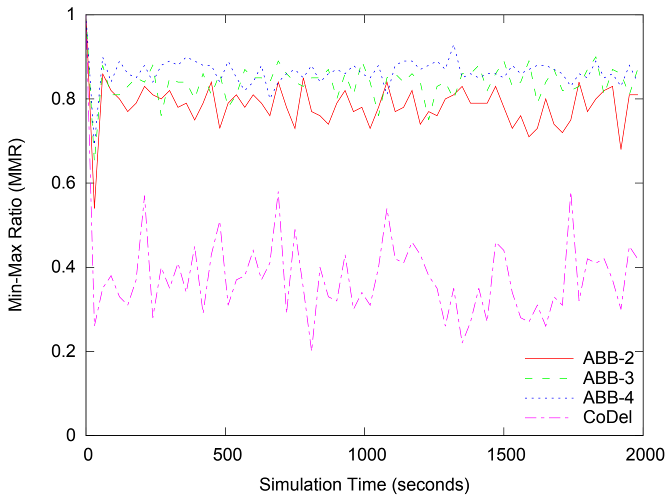
Figure 5.3: Experiment BASE throughput fairness results.

shown to get worse as the number of flows increases. As shown in our prior work [38], this is primarily a function of the disparity in the uncongested path RTTs. It is known as TCP RTT unfairness.

DRR-CoDel fairness results show that DRR-CoDel is able to provide fairness close to max-min fairness. With results not shown, DRR with bloated buffer is max-min fair. The results show SFQ-CoDel has almost identical fairness results to that of DRR-CoDel. This is not surprising as in this case the number of bins SFQ-CoDel is configured to use far



(a) Jain's Fairness Index



(b) Min-Max Ratio

Figure 5.4: Experiment BASE throughput fairness results over time.

exceeds the number of flows in all runs such that the chance of collision is slim. But with more flows running, the fairness of SFQ-CoDel can be quite different. The single-queue CoDel is shown to be much less fair especially with wider RTT disparities. Overall the long term fairness of the ABB scheme is very close to that of SFQ-CoDel and DRR-CoDel. Occasionally we see that ABB scheme with 3 and 4 bins provides slightly better long term fairness than DRR-CoDel. With the use of more bins, ABB scheme in general is able to better address the RTT unfairness issue.

Table 5.3: Experiment BASE TCP Throughput: Mean / CoV / Sum (Mbps)

#FTPs	3	5	7	9	11
DRR-CoDel	12.1 / 0.03 / 36.4	7.4 / 0.04 / 37.0	5.3 / 0.05 / 37.2	4.1 / 0.06 / 37.3	3.4 / 0.07 / 37.4
SFQ-CoDel	12.2 / 0.03 / 36.5	7.4 / 0.03 / 37.1	5.3 / 0.04 / 37.2	4.1 / 0.05 / 37.3	3.4 / 0.06 / 37.4
ABB-2	12.1 / 0.04 / 36.3	7.4 / 0.04 / 36.9	5.3 / 0.06 / 37.3	4.2 / 0.07 / 37.4	3.4 / 0.09 / 37.4
ABB-3	12.1 / 0.05 / 36.4	7.4 / 0.04 / 37.0	5.3 / 0.04 / 37.2	4.2 / 0.06 / 37.4	3.4 / 0.07 / 37.4
ABB-4	12.1 / 0.05 / 36.4	7.4 / 0.04 / 37.0	5.3 / 0.04 / 37.2	4.2 / 0.05 / 37.4	3.4 / 0.06 / 37.4
CoDel	12.1 / 0.19 / 36.3	7.4 / 0.24 / 37.2	5.3 / 0.28 / 37.4	4.2 / 0.31 / 37.4	3.4 / 0.36 / 37.4

Figure 5.4 shows the throughput fairness results over time for the run of nine FTP flows under the ABB schemes configured with 2, 3, and 4 bins. We also show the fairness results from CoDel for comparison purpose. The throughput samples were taken every 30 seconds. As shown, the fairness over time is close to that of long term fairness in Figure 5.3 with slight variations. For other runs of different number of FTP flows, the results not shown are similar. The results also suggest that the ABB scheme is able to provide reasonable fairness over a time scale that is an order of magnitude higher than the reclassification interval.

Table 5.3 summarizes the throughput results by the FTP flows under different schemes. With results not shown, DRR with bloated buffer achieves the best total throughput of 37.4 Mbps in all runs. When compared to DRR with bloated buffer, the total throughput achieved by all schemes under investigation is less in the runs of fewer flows. Once the number of flows increases, total throughput of various schemes gets close to that of DRR with bloated buffer. When compared to the DRR-CoDel, the total throughput results from the ABB scheme with 3 and 4 bins are shown to be the same or occasionally higher. This suggests that the ABB scheme provides good utilization of the channel capacity. The coefficient of variation (CoV) values are consistent with Figure 5.3.

Table 5.4 shows the average RTT experienced by the TCP/FTP packets. The average RTT includes the inherent path delay plus the queueing delay. Subtracting the inherent average uncongested path RTT of 80 ms, the values meet the target delay of 20 ms set for CoDel. As the number of flows increases, the RTT variations increase. This reflects our varied settings of the uncongested RTTs for different flows. Again, the ABB schemes

Table 5.4: Experiment BASE TCP RTT: Mean / Stddev

#FTPs	3	5	7	9	11
DRR-CoDel	0.087 / 0.008	0.088 / 0.013	0.088 / 0.019	0.089 / 0.024	0.091 / 0.030
SFQ-CoDel	0.088 / 0.008	0.088 / 0.013	0.087 / 0.019	0.088 / 0.024	0.089 / 0.029
ABB-2	0.087 / 0.007	0.088 / 0.013	0.090 / 0.018	0.092 / 0.023	0.094 / 0.028
ABB-3	0.087 / 0.007	0.087 / 0.013	0.088 / 0.018	0.090 / 0.023	0.092 / 0.029
ABB-4	0.087 / 0.007	0.087 / 0.013	0.088 / 0.018	0.089 / 0.024	0.091 / 0.029
CoDel	0.092 / 0.008	0.095 / 0.014	0.098 / 0.020	0.100 / 0.026	0.101 / 0.032

Table 5.5: Experiment BASE VoIP Isolation Performance
(Mean Latency / Mean Loss Rate (Percentage) / R-Value)

#FTPs	3	5	7	9	11
DRR-CoDel	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.039 / 0.000 / 93.3
SFQ-CoDel	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3
ABB-2	0.040 / 0.001 / 93.2	0.040 / 0.001 / 93.2	0.041 / 0.000 / 93.2	0.043 / 0.001 / 93.2	0.045 / 0.005 / 93.1
ABB-3	0.040 / 0.000 / 93.2	0.040 / 0.000 / 93.2	0.040 / 0.000 / 93.2	0.041 / 0.001 / 93.2	0.043 / 0.000 / 93.2
ABB-4	0.040 / 0.000 / 93.2	0.040 / 0.000 / 93.2	0.040 / 0.001 / 93.2	0.041 / 0.000 / 93.2	0.043 / 0.004 / 93.1
CoDel	0.050 / 0.055 / 92.8	0.053 / 0.111 / 92.4	0.055 / 0.138 / 92.3	0.057 / 0.157 / 92.1	0.059 / 0.200 / 91.9

are not shown to diverge from providing good delay performance.

In the BASE experiment, a simulated simplex VoIP stream is used to assess the capability of the schemes in isolating low bit rate flows from greedy ones. Table 5.5 provides the results. The latency is one-way average latency experienced by the VoIP packets with uncongested path RTT of 80 ms, the average uncongested RTT of the FTP flows running in parallel.

All schemes are shown to provide similarly good R-value. The ABB scheme is able to provide comparably close latency performance with extremely low loss rates to that of DRR-CoDel and SFQ-CoDel. The clearest result is single-queue CoDel has longer latency and higher loss rates. This suggests multi-queue AQMs can provide better flow isolation capability than single queue CoDel.

Table 5.6 provides results of this experiment on the dynamic behavior of the ABB scheme. The results provide per-flow bin switches and the rate of such switching. The rate of the switches is calculated by the actual number of switches observed divided by the max number of potential switches. For example, given 3 flows, 2000 second total simulation

Table 5.6: ABB Dynamic Behavior for BASE

#FTPs	3	5	7	9	11
	Number of Bin Switches (rate)				
ABB-2	1213 (0.20)	2668 (0.27)	4059 (0.29)	5718 (0.32)	7266 (0.33)
ABB-3	1543 (0.26)	4338 (0.43)	6772 (0.48)	8393 (0.47)	10268 (0.47)
ABB-4	1514 (0.25)	4381 (0.44)	7280 (0.52)	10055 (0.56)	12045 (0.55)
	Disruption Time (ratio)				
ABB-2	2.7 (0.001)	7.2 (0.004)	10.9 (0.005)	12.5 (0.006)	12.3 (0.006)
ABB-3	4.1 (0.002)	8.0 (0.004)	11.3 (0.006)	13.2 (0.007)	16.1 (0.008)
ABB-4	3.8 (0.002)	8.2 (0.004)	11.3 (0.006)	13.8 (0.007)	16.8 (0.008)

time, and one second reclassification interval, the maximum number of switches is 3×2000. So the rate of the switches is $1213/6000 = 0.20$. The results also provide the disruption time and ratio from our solution addressing packet reordering issue due to bin switches. The ratio is calculated by the actual disruption time divided by the total simulation time. As the number of flows increases, the rate of per-flow bin switches goes up reflecting the wider RTT gaps among the flows and the scheme's responsiveness to the workload changes. The disruption time and ratio remain to be very low showing that our solution to resolve packet reordering issue works well as workload increases.

We caution that the rate of flow bin switches is not an indicator of system instability. The ability of the scheme in approximating long term fairness is built upon the idea of having flows switch bins by responding to the changes of flow consumption levels. So a reasonably high switching rate can be a healthy indicator that the system is operating properly when the system is loaded. This is especially true in the case of greedy TCP flows. Due to TCP being responsive, TCP flows are likely operating around their fair share, moving back and forth among a few bins. In this case, the legitimate concern is how such switches can cause disruption in terms of the above disruption time and ratio measurements. The above results show in the case of TCP flows the system gives rise to very little disruption.

Table 5.7: Experiment ME1 Flow Throughput (Mbps)

Scheme	FTP (mean / CoV / sum)	On-off (mean / CoV / sum)	Total
DRR-CoDel	4.12 / 0.02 / 16.47	0.53 / 0.08 / 19.15	35.63
SFQ-CoDel	4.24 / 0.03 / 16.95	0.51 / 0.09 / 18.45	35.41
ABB-2	4.31 / 0.06 / 17.26	0.52 / 0.07 / 18.60	35.86
ABB-3	4.28 / 0.04 / 17.12	0.50 / 0.11 / 17.91	35.03
ABB-4	4.17 / 0.03 / 16.68	0.51 / 0.07 / 18.35	35.04
CoDel	4.60 / 0.27 / 18.40	0.51 / 0.10 / 18.53	36.93

5.3.2 ME Simulation Results

The set of ME experiments is used to evaluate the throughput fairness among long-lived greedy "elephant" flows in the presence of large number of short-lived on-off "mice" traffic flows under the ABB scheme. We are also interested in the dynamic behavior of the scheme with this type of traffic load.

The first variant, called ME1, runs 36 exponential on/off TCP flows with an average rate of 500 kbps. The overall demand is thus 18 Mbps. The on/off time ratio is 1/3 with on time drawn between 2 and 4 seconds randomly in uniform distribution. ME1 also runs 4 TCP/FTP flows sourced from 4 separate FTP servers. For a 38 Mbps channel capacity, the expected max-min fair allocation for the FTP flows is 5 Mbps. All flows have their uncongested path RTTs set to the same 80 ms. We also set the FTP server link speeds to 5, 10, 20, and 1000 Mbps to see if the diverse server links that exist in the Internet may have any impact over the bandwidth allocations.

Table 5.7 provides the throughput results of ME1. When compared to schemes such as DRR-CoDel and SFQ-CoDel, the FTP throughput CoV values for ABB-2, ABB-3, and ABB-4 are all similarly small. This suggests that the ABB scheme is able to maintain similarly good fairness under this type of traffic load. It does not appear that the FTP server access link speeds have significant impact over the bandwidth allocation as long as the access link speeds are above the fair bandwidth share of a flow. CoDel is shown to be much less fair with a much larger CoV. The total throughput under ABB is also comparable to that of DRR-CoDel and SFQ-CoDel. This shows that the ABB scheme is able to provide

Table 5.8: Experiment ME1 Flow Average RTTs (second)

Scheme	FTP (mean / stddev)	On-off (mean / stddev)	Overall Mean
DRR-CoDel	0.094 / 0.013	0.084 / 0.000	0.085 / 0.005
SFQ-CoDel	0.092 / 0.011	0.083 / 0.000	0.084 / 0.005
ABB-2	0.097 / 0.011	0.086 / 0.001	0.087 / 0.005
ABB-3	0.095 / 0.010	0.083 / 0.000	0.084 / 0.005
ABB-4	0.094 / 0.010	0.083 / 0.000	0.084 / 0.005
CoDel	0.096 / 0.005	0.094 / 0.000	0.094 / 0.002

Table 5.9: ABB Dynamics with Experiment ME1

Scheme	Number of Bin Switches (rate)	Disruption Time (ratio)
ABB-2	5045 (0.06)	8.2 (0.004)
ABB-3	6160 (0.08)	6.2 (0.003)
ABB-4	7224 (0.09)	6.5 (0.003)

reasonable channel utilization. We noticed that the average throughput for the exponential on-off flow is slightly above the targeted average rate of 500 kbps. In a separate experiment with just three exponential on-off flows, we observed similar result. This is likely attributed to the ns2 implementation of the exponential on-off traffic generator. For example, the use of integer division in calculating mean packet inter-arrival time interval in our case results in a shorter interval value. This should cause the actual generated traffic slightly above the targeted average rate.

The RTT information for ME1 is given in Table 5.8. Similar to that of DRR-CoDel and SFQ-CoDel, the average RTT for the on-off mice flows under ABB is lower than the average RTT for the FTP flows. On the contrary, single queue CoDel does not provide a lower average RTT for the mice flows. This indicates that the ABB scheme, like DRR-CoDel and SFQ-CoDel, is able to provide better flow isolation than CoDel.

The dynamic behavior of ABB in terms of flow bin switches and disruption time in ME1 is shown in Table 5.9. The low switching rate is due to the scheme being able to isolate the mice flows on the first bin for majority of the time without much switching. The low disruption time and ratio show that the bin switches in this scenario causes little disruption and the scheme behaves well.

Table 5.10: Experiment ME2 Flow Throughput (Mbps)

Scheme	FTP (mean / CoV / sum)	On-off (mean / CoV / sum)	Total
DRR-CoDel	4.54 / 0.02 / 18.17	0.10 / 0.04 / 18.47	36.65
SFQ-CoDel	4.56 / 0.01 / 18.25	0.10 / 0.04 / 18.48	36.73
ABB-2	4.69 / 0.06 / 18.76	0.10 / 0.04 / 18.43	37.19
ABB-3	4.63 / 0.01 / 18.54	0.10 / 0.04 / 18.48	37.02
ABB-4	4.63 / 0.01 / 18.51	0.10 / 0.04 / 18.50	37.01
CoDel	4.67 / 0.23 / 18.68	0.10 / 0.04 / 18.41	37.09

Table 5.11: Experiment ME2 Flow Average RTTs (second)

Scheme	FTP (mean / stddev)	On-off (mean / stddev)	Overall Mean
DRR-CoDel	0.100 / 0.025	0.092 / 0.000	0.092 / 0.004
SFQ-CoDel	0.096 / 0.014	0.090 / 0.000	0.090 / 0.002
ABB-2	0.102 / 0.014	0.090 / 0.000	0.091 / 0.003
ABB-3	0.097 / 0.011	0.090 / 0.000	0.090 / 0.002
ABB-4	0.096 / 0.010	0.090 / 0.000	0.090 / 0.002
CoDel	0.097 / 0.005	0.102 / 0.000	0.102 / 0.001

The second variant, called ME2, is similar to ME1 except it runs 180 exponential on/off TCP flows with an average rate of 100 kbps. The on/off time intervals are set to 2 and 3 seconds respectively. The results for ME2 are similar to that of ME1. The detailed data are given in Table 5.10 and Table 5.11 and Table 5.12.

The third variant, called ME3, runs 90 low rate exponential on/off TCP flows with an average rate of 100 kbps and 90 middle rate exponential on/off TCP flows with an average rate of randomly drawn between 300 – 600 kbps. The 4 FTP flows are set the same as the other two variants. Due to the large number of middle rate flows involved along with the 4 greedy FTP flows, ME3 is a highly congested scenario.

The throughput and RTT results are given in Table 5.13 and Table 5.14 for ME3.

Table 5.12: ABB Dynamics with Experiment ME2

Scheme	Number of Bin Switches (rate)	Disruption Time (ratio)
ABB-2	42 (0.00)	0.1 (0.000)
ABB-3	2244 (0.01)	6.1 (0.003)
ABB-4	2846 (0.01)	6.5 (0.003)

Table 5.13: Experiment ME3 Flow Throughput (Mbps)

Scheme	FTP (mean / CoV / sum)	Low On-off (mean / CoV / sum)	Mid On-off (mean / CoV / sum)	Total
DRR-CoDel	0.31 / 0.00 / 1.24	0.10 / 0.04 / 9.24	0.30 / 0.01 / 27.07	37.55
SFQ-CoDel	0.32 / 0.01 / 1.29	0.10 / 0.05 / 9.23	0.30 / 0.16 / 27.02	37.54
ABB-2	0.30 / 0.02 / 1.21	0.10 / 0.04 / 9.23	0.30 / 0.01 / 27.02	37.45
ABB-3	0.30 / 0.03 / 1.21	0.10 / 0.04 / 9.22	0.30 / 0.01 / 26.95	37.38
ABB-4	0.30 / 0.01 / 1.21	0.10 / 0.05 / 9.21	0.30 / 0.01 / 26.93	37.36
CoDel	0.18 / 0.57 / 0.72	0.10 / 0.04 / 9.23	0.31 / 0.03 / 27.56	37.51

Table 5.14: Experiment ME3 Flow Average RTTs (second)

Scheme	FTP (mean / stddev)	Low On-off (mean / stddev)	Mid On-off (mean / stddev)	Overall Mean
DRR-CoDel	0.145 / 0.001	0.127 / 0.001	0.153 / 0.003	0.140 / 0.013
SFQ-CoDel	0.123 / 0.001	0.101 / 0.012	0.124 / 0.004	0.113 / 0.015
ABB-2	0.117 / 0.002	0.108 / 0.000	0.115 / 0.001	0.112 / 0.004
ABB-3	0.120 / 0.003	0.107 / 0.001	0.118 / 0.001	0.113 / 0.006
ABB-4	0.119 / 0.003	0.107 / 0.000	0.116 / 0.001	0.112 / 0.005
CoDel	0.129 / 0.011	0.120 / 0.001	0.112 / 0.001	0.117 / 0.005

For the throughput results, we can see that the ABB scheme is fair similarly to DRR-CoDel and SFQ-CoDel. Both FTP flows and middle rate on/off flows obtain similar average throughput. On the other hand, CoDel is not shown to be fair as it does not provide good isolation between different classes of flows. It allows middle rate on/off flows to obtain higher average throughput than FTP flows as it targets high bandwidth FTP flows with more packet losses when all flows share a single queue. The RTT results are similar to that of ME1 and ME2. Greedy FTP flows have higher RTTs and mice flows have lower RTTs.

It is expected in the highly congested scenario the ABB scheme will exhibit a high degree of dynamics in terms of bin switches. The results given in Table 5.15 confirm the fact. However, the overall disruption these switches caused remains to be low.

5.4 Results and Analysis for Multi Tier Environment

We report the results for the set of experiments from Table 5.2 designed for the multi-tier environment in this section. The channel capacity is set to 1 Gbps and the available

Table 5.15: ABB Dynamics with Experiment ME3

Scheme	Number of Bin Switches (rate)	Disruption Time (ratio)
ABB-2	73880 (0.20)	22.5 (0.011)
ABB-3	106564 (0.29)	38.9 (0.019)
ABB-4	117082 (0.32)	40.3 (0.020)

bandwidth is about 948 Mbps. The multi-tier set of experiments is designed to evaluate the effectiveness of the ABB scheme in providing *weighted* fairness. Unless otherwise specified, the reclassification interval of ABB is set to 1 second. For this set of experiments, all simulation times are set to 1000 seconds unless otherwise specified.

5.4.1 TIER Simulation Results

The TIER experiment is similar to the BASE experiment in purpose but involving the tiered service quality levels. All experiments involve TCP/FTP flows in three tiers (Tier1, Tier2, and Tier4 with a weight of 1, 2, and 4 respectively). A single VoIP flow at Tier1 is also included. We ran two variants of the experiments.

The first variant of the TIER experiment is called TierG. It runs a varying number (11 – 55) of FTP flows at Tier1. For both Tier2 and Tier4, a single FTP flow is used for all runs. All flows are set to have the same RTT of 80 ms.

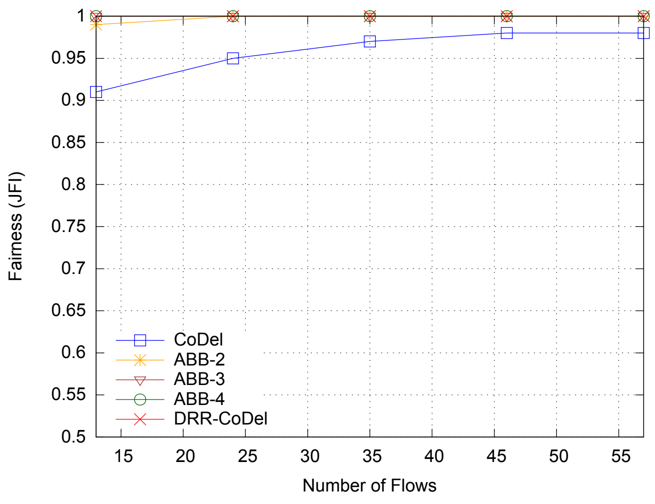
Figure 5.5 shows the weighted throughput fairness results from five simulation runs with different number of FTP flows. As shown, the ABB scheme especially for the configurations with 3 and 4 bins is shown to be close to DRR-CoDel in terms of approximating weighted fairness. It is expected that neither SFQ-CoDel (not shown) or CoDel is able to provide weighted fairness. The average tiered flow throughput is given in Table 5.16. The ABB scheme is able to provide high utilization of the channel capacity as the total throughput of the ABB scheme is comparable to others. The average RTT information is given in Table 5.17. The ABB scheme is shown to maintain similar queueing delay for all flow tiers. The VoIP performance for the single VoIP flow is given in Table 5.18. Again the ABB scheme is able to provide similar performance to that of DRR-CoDel.

Table 5.16: Experiment TierG Flow Throughput Mean / CoV / Sum (Mbps)

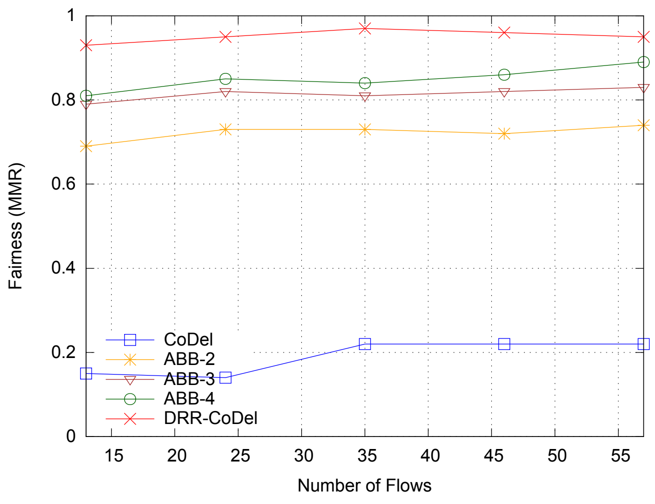
#FTPs (T1)	11				22				33			
Tier/Sum	T1	T2	T4	Sum	T1	T2	T4	Sum	T1	T2	T4	Sum
DRR-CoDel	55.6 / 0.0	111.0	210.1	933.0	33.6 / 0.0	66.2	129.3	934.5	24.1 / 0.0	47.4	92.7	933.8
ABB-2	59.3 / 0.1	102.3	179.3	933.6	34.9 / 0.0	56.1	110.0	934.3	24.7 / 0.0	41.6	76.3	934.6
ABB-3	57.8 / 0.0	107.4	190.4	933.6	34.3 / 0.0	61.8	119.2	934.6	24.4 / 0.0	45.1	83.3	934.6
ABB-4	57.9 / 0.0	108.9	186.7	932.9	34.0 / 0.0	64.1	122.4	934.5	24.3 / 0.0	45.0	86.6	934.6
CoDel	68.8 / 0.1	85.1	91.1	933.3	39.1 / 0.1	34.3	39.0	934.2	26.7 / 0.1	25.0	27.4	934.2
#FTPs (T1)	44				55							
Tier/Sum	T1	T2	T4	Sum	T1	T2	T4	Sum				
DRR-CoDel	18.7 / 0.0	36.9	73.4	934.3	15.4 / 0.0	30.0	59.3	934.4				
ABB-2	19.1 / 0.0	32.6	61.0	934.8	15.6 / 0.0	27.8	48.5	934.9				
ABB-3	19.0 / 0.0	34.8	64.6	933.4	15.5 / 0.0	28.6	53.6	934.9				
ABB-4	18.9 / 0.0	35.0	66.9	934.8	15.5 / 0.0	28.8	55.5	934.8				
CoDel	20.3 / 0.1	20.8	20.0	934.4	16.4 / 0.1	16.6	15.9	934.3				

Table 5.17: Experiment TierG Flow RTT Mean / Stddev

#FTPs (T1)	11			22			33		
Tier	T1	T2	T4	T1	T2	T4	T1	T2	T4
DRR-CoDel	0.088 / 0.000	0.089	0.090	0.091 / 0.000	0.091	0.091	0.092 / 0.000	0.093	0.092
ABB-2	0.092 / 0.001	0.089	0.088	0.094 / 0.001	0.088	0.087	0.094 / 0.001	0.088	0.086
ABB-3	0.089 / 0.001	0.087	0.088	0.090 / 0.000	0.087	0.087	0.091 / 0.001	0.088	0.087
ABB-4	0.088 / 0.000	0.087	0.088	0.089 / 0.000	0.087	0.087	0.090 / 0.001	0.088	0.088
CoDel	0.099 / 0.000	0.099	0.099	0.101 / 0.000	0.100	0.100	0.101 / 0.000	0.101	0.101
#FTPs (T1)	44			55					
Tier	T1	T2	T4	T1	T2	T4			
DRR-CoDel	0.094 / 0.000	0.094	0.094	0.095 / 0.001	0.096	0.095			
ABB-2	0.095 / 0.001	0.088	0.086	0.095 / 0.001	0.090	0.085			
ABB-3	0.092 / 0.003	0.088	0.086	0.092 / 0.001	0.088	0.087			
ABB-4	0.091 / 0.000	0.088	0.088	0.091 / 0.000	0.089	0.088			
CoDel	0.102 / 0.000	0.102	0.102	0.102 / 0.000	0.102	0.102			



(a) Jain's Fairness Index



(b) Min-Max Ratio

Figure 5.5: Experiment TierG weighted throughput fairness results.

The second variant of the TIER experiment is called TierM. This variant is structured differently from the first one. For different runs, it runs the same number of FTP flows but with a different mix of FTP flows in all three tiers. For the five runs, the mixes of numbers of flows in each tier are:

1. T1 / T2 / T4 = 13 / 11 / 11

2. T1 / T2 / T4 = 25 / 5 / 5

Table 5.18: Experiment TierG VoIP Isolation Performance (Mean Latency / Mean Loss Rate (Percentage) / R-Value)

#FTPs (T1)	11	22	33
DRR-CoDel	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3
ABB-2	0.039 / 0.000 / 93.3	0.039 / 0.004 / 93.2	0.039 / 0.008 / 93.2
ABB-3	0.039 / 0.000 / 93.3	0.039 / 0.006 / 93.2	0.039 / 0.002 / 93.3
ABB-4	0.038 / 0.000 / 93.3	0.039 / 0.008 / 93.2	0.039 / 0.004 / 93.3
CoDel	0.055 / 0.006 / 92.8	0.057 / 0.030 / 92.7	0.058 / 0.058 / 92.6
#FTPs (T1)	44	55	
DRR-CoDel	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	
ABB-2	0.039 / 0.006 / 93.2	0.039 / 0.002 / 93.2	
ABB-3	0.039 / 0.004 / 93.2	0.039 / 0.004 / 93.2	
ABB-4	0.039 / 0.002 / 93.3	0.039 / 0.008 / 93.2	
CoDel	0.058 / 0.104 / 92.3	0.058 / 0.138 / 92.2	

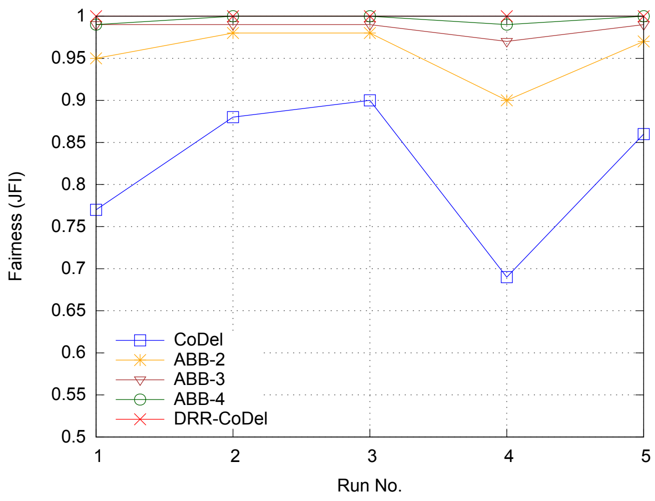
3. T1 / T2 / T4 = 5 / 25 / 5

4. T1 / T2 / T4 = 5 / 5 / 25

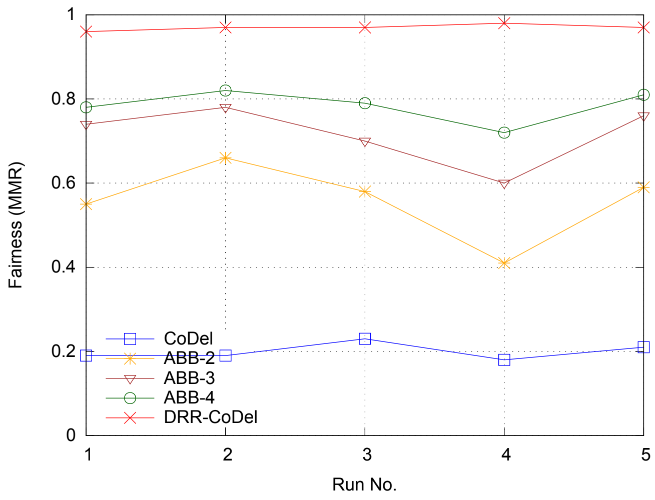
5. T1 / T2 / T4 = 20 / 10 / 5

These mixes represent a spectrum of workloads with some being more heavy than others. All flows are still set to have the same RTT of 80 ms and the simulation time remains to be 1000 seconds.

Figure 5.6 provides the weighted throughput fairness results from the five simulation runs of TierM. Again as shown, the ABB scheme especially for the configurations of 3 and 4 bins provides close approximation of weighted fairness to that of DRR-CoDel. The average tiered flow throughput for TierM is given in Table 5.19. The small CoV values suggest that the fairness within the same tier is great under this workload where all flows having the same RTT. Among the five runs, the worst approximation of the weighted fairness comes at the run #4, which has the most number of Tier4 FTP flows running. It represents the most imbalanced and congested scenario. From the throughput results, the ABB scheme again is able to provide high utilization of the channel capacity. The average RTT information is given in Table 5.20. The VoIP performance for the single VoIP flow is given in Table 5.21.



(a) Jain's Fairness Index



(b) Min-Max Ratio

Figure 5.6: Experiment TierM weighted throughput fairness results.

We can draw the similar conclusions from the RTT and VoIP results as those in the TierG.

Finally we provide the results of the ABB dynamic behavior for the two variants of the TIER experiment in Table 5.22 and Table 5.23. The disruption time remains to be low in all cases, showing the scheme works well with TCP flows.

Table 5.19: Experiment TierM Flow Throughput Mean / CoV / Sum (Mbps)

#FTPs	13	11	11	35	25	5	5	35
Tier/Sum	T1	T2	T4	Sum	T1	T2	T4	Sum
DRR-CoDel	12.0 / 0.0	23.7 / 0.0	47.0 / 0.0	934.1	17.1 / 0.0	33.9 / 0.0	67.2 / 0.0	934.4
ABB-2	16.9 / 0.0	23.7 / 0.0	41.3 / 0.0	934.6	19.7 / 0.0	31.6 / 0.0	56.7 / 0.0	934.8
ABB-3	14.2 / 0.0	23.9 / 0.0	44.3 / 0.0	934.5	18.6 / 0.0	32.8 / 0.0	61.3 / 0.0	934.7
ABB-4	13.6 / 0.0	24.0 / 0.0	45.0 / 0.0	934.5	18.1 / 0.0	33.5 / 0.0	62.9 / 0.0	934.6
CoDel	27.4 / 0.1	25.9 / 0.1	26.6 / 0.1	934.4	26.6 / 0.1	27.3 / 0.1	26.4 / 0.0	934.3
#FTPs	5	25	5	35	5	5	25	35
Tier/Sum	T1	T2	T4	Sum	T1	T2	T4	Sum
DRR-CoDel	12.6 / 0.0	25.0 / 0.0	49.3 / 0.0	934.2	8.2 / 0.0	16.4 / 0.0	32.5 / 0.0	934.4
ABB-2	17.7 / 0.0	25.2 / 0.0	43.0 / 0.0	934.5	16.4 / 0.1	19.7 / 0.0	30.2 / 0.0	934.7
ABB-3	15.5 / 0.0	25.1 / 0.0	45.8 / 0.0	934.6	12.2 / 0.0	17.8 / 0.0	31.4 / 0.0	934.6
ABB-4	14.3 / 0.0	25.1 / 0.0	46.9 / 0.0	934.5	10.6 / 0.0	17.2 / 0.0	31.8 / 0.0	934.5
CoDel	23.6 / 0.1	27.5 / 0.1	25.8 / 0.1	934.3	26.0 / 0.1	28.2 / 0.1	26.5 / 0.1	934.0
#FTPs	20	10	5	35				
Tier/Sum	T1	T2	T4	Sum				
DRR-CoDel	15.7 / 0.0	31.2 / 0.0	61.6 / 0.0	934.5				
ABB-2	19.2 / 0.0	28.8 / 0.0	52.6 / 0.0	934.7				
ABB-3	17.4 / 0.0	30.4 / 0.0	56.4 / 0.0	934.7				
ABB-4	16.9 / 0.0	30.7 / 0.0	57.9 / 0.0	934.7				
CoDel	26.7 / 0.1	26.3 / 0.1	27.6 / 0.0	934.3				

Table 5.20: Experiment TierM Flow RTT Mean / Stddev

#FTPs	13	11	11	25	5	5
Tier	T1	T2	T4	T1	T2	T4
DRR-CoDel	0.092 / 0.000	0.092 / 0.000	0.092 / 0.000	0.092 / 0.000	0.092 / 0.000	0.092 / 0.000
ABB-2	0.100 / 0.000	0.093 / 0.001	0.088 / 0.001	0.097 / 0.001	0.090 / 0.001	0.087 / 0.000
ABB-3	0.095 / 0.000	0.089 / 0.000	0.086 / 0.000	0.093 / 0.001	0.088 / 0.000	0.086 / 0.000
ABB-4	0.094 / 0.001	0.089 / 0.000	0.087 / 0.000	0.091 / 0.000	0.088 / 0.000	0.087 / 0.000
CoDel	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000
#FTPs	5	25	5	5	5	25
Tier	T1	T2	T4	T1	T2	T4
DRR-CoDel	0.092 / 0.000	0.093 / 0.000	0.093 / 0.000	0.091 / 0.000	0.092 / 0.000	0.093 / 0.000
ABB-2	0.100 / 0.000	0.094 / 0.001	0.088 / 0.000	0.101 / 0.000	0.098 / 0.000	0.090 / 0.001
ABB-3	0.096 / 0.001	0.090 / 0.001	0.086 / 0.000	0.099 / 0.000	0.092 / 0.000	0.087 / 0.000
ABB-4	0.095 / 0.000	0.090 / 0.000	0.088 / 0.000	0.097 / 0.000	0.091 / 0.000	0.088 / 0.000
CoDel	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000
#FTPs	20	10	5			
Tier	T1	T2	T4			
DRR-CoDel	0.092 / 0.000	0.093 / 0.000	0.093 / 0.000			
ABB-2	0.098 / 0.001	0.090 / 0.001	0.087 / 0.001			
ABB-3	0.093 / 0.001	0.088 / 0.001	0.086 / 0.000			
ABB-4	0.092 / 0.001	0.088 / 0.000	0.087 / 0.000			
CoDel	0.101 / 0.000	0.101 / 0.000	0.101 / 0.000			

Table 5.21: Experiment TierM VoIP Isolation Performance
(Mean Latency / Mean Loss Rate (Percentage) / R-Value)

#FTPs (T1:T2:T4)	13:11:11	25:5:5	5:25:5	5:5:25	20:10:5
DRR-CoDel	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3	0.038 / 0.000 / 93.3
ABB-2	0.039 / 0.004 / 93.2	0.039 / 0.004 / 93.2	0.039 / 0.004 / 93.2	0.039 / 0.004 / 93.2	0.039 / 0.002 / 93.3
ABB-3	0.039 / 0.002 / 93.3	0.039 / 0.006 / 93.2	0.039 / 0.002 / 93.3	0.038 / 0.010 / 93.2	0.039 / 0.006 / 93.2
ABB-4	0.039 / 0.006 / 93.2	0.039 / 0.004 / 93.3	0.039 / 0.004 / 93.3	0.038 / 0.000 / 93.3	0.039 / 0.008 / 93.2
CoDel	0.058 / 0.082 / 92.4	0.058 / 0.060 / 92.5	0.058 / 0.056 / 92.6	0.058 / 0.046 / 92.6	0.058 / 0.078 / 92.5

Table 5.22: ABB Dynamic Behavior for TierG

#FTPs	11	22	33	44	55
	Number of Bin Switches (rate)				
ABB-2	1855 (0.14)	4595 (0.19)	7152 (0.20)	9286 (0.20)	12385 (0.22)
ABB-3	2486 (0.19)	5554 (0.23)	9001 (0.26)	12564 (0.27)	16374 (0.29)
ABB-4	2590 (0.20)	6059 (0.25)	10263 (0.29)	14445 (0.31)	19216 (0.34)
	Disruption Time (ratio)				
ABB-2	6.9 (0.007)	10.0 (0.010)	10.9 (0.011)	11.1 (0.011)	11.1 (0.011)
ABB-3	6.0 (0.006)	7.2 (0.007)	8.1 (0.008)	9.0 (0.009)	9.0 (0.009)
ABB-4	5.3 (0.005)	6.8 (0.007)	7.8 (0.008)	8.4 (0.008)	8.7 (0.009)

Table 5.23: ABB Dynamic Behavior for TierM

#FTPs (T1:T2:T4)	13:11:11	25:5:5	5:25:5	5:5:25	20:10:5
	Number of Bin Switches (rate)				
ABB-2	5946 (0.17)	5971 (0.17)	6799 (0.19)	6854 (0.20)	6189 (0.18)
ABB-3	8428 (0.24)	8958 (0.26)	8707 (0.25)	8243 (0.24)	8719 (0.25)
ABB-4	10257 (0.29)	10514 (0.30)	10067 (0.29)	9734 (0.28)	10426 (0.30)
	Disruption Time (ratio)				
ABB-2	9.6 (0.010)	10.1 (0.010)	10.3 (0.010)	9.7 (0.010)	10.0 (0.010)
ABB-3	6.7 (0.007)	7.2 (0.007)	7.3 (0.007)	6.4 (0.006)	7.0 (0.007)
ABB-4	6.5 (0.007)	7.1 (0.007)	7.3 (0.007)	6.3 (0.006)	7.0 (0.007)

Table 5.24: Flow Throughput by Tiers for Experiment UDP

Scheme	FTP-T1	UDP-T1	FTP-T2	UDP-T2	FTP-T4	UDP-T4	Total
DRR-CoDel	11.15	12.64	22.12	25.07	43.68	49.28	933.3
ABB-2	13.19	49.32	20.64	49.32	37.51	49.32	932.7
ABB-3	10.97	48.84	20.35	48.84	39.91	49.33	930.6
ABB-4	10.69	37.94	20.99	39.04	41.66	49.29	933.0
ABB-5	11.02	32.49	21.35	38.51	41.52	49.28	933.1
CoDel	23.24	49.31	24.31	49.33	23.79	49.32	932.7

5.4.2 UDP Simulation Results

The **U**DP experiment is designed to assess the performance of the ABB scheme in isolating unresponsive high bandwidth UDP flows. Our prior work showed that single queue schemes fail to provide UDP isolation. We set to answer if the ABB scheme with its use of multiple queues is able to provide better protection from unresponsive UDP flows.

The experiment is set up with three flow tiers: Tier1, Tier2, and Tier4. Each tier runs 11 FTP flows. In addition, each tier also has a single high bandwidth UDP flow sourced at 50 Mbps. All flows have the same uncongested path RTT of 80 ms.

Table 5.24 provides both FTP and UDP flow throughput results for each tier. In the FTP case, the throughput is the average of that 11 flows at each tier. As expected, DRR-CoDel provides nearly optimal UDP isolation. CoDel does not provide UDP isolation at all.

When compared to single queue CoDel, ABB with both 2 and 3 bins also do not provide adequate UDP isolation but the allocations to the FTP flows in different tiers are approximately more weighted fair. It takes ABB with 4 bins to start providing some degree of UDP isolation. Note this scenario involves three UDP flow tiers. To provide good UDP flow isolation, ideally the three tiered high speed UDP flows each need to be isolated into its own bins. A minimum of 4 bins are needed with one extra bin for the TCP flows. Otherwise when a FTP flow gets into the same bin with a UDP flow, the UDP flow will “win” over the FTP flow in terms of bandwidth allocation. Considering TCP flow dynamics, an extra

Table 5.25: ABB Dynamics with Experiment UDP

Scheme	Number of Bin Switches (rate)	Disruption Time (ratio)
ABB-2	6413 (0.18)	8.6 (0.009)
ABB-3	6260 (0.17)	15.8 (0.016)
ABB-4	7956 (0.22)	42.7 (0.043)
ABB-5	9451 (0.26)	40.8 (0.041)

bin should help by serving as a buffer between the TCP flows and the UDP flows. We confirmed this by running ABB with 5 bins in this case. The throughput results for ABB with 5 bins are also given in the above table and show a huge improvement over ABB with 4 bins in providing UDP flow isolation.

In terms of ABB dynamics, the results are given in Table 5.25. With the number of TCP flows involved, the bin switching rate is not low and is consistent with that of TierG and TierM with similar number of TCP flows running. One noticeable difference between the disruption time of this experiment and others is the relatively higher disruption time ratios in the case of 4 and 5 bins. This is due to the large disparity in queue lengths among the bins. A bin with unresponsive high bandwidth UDP flow will likely build up its queue while a bin with only responsive TCP flows maintains short queue length under CoDel.

5.4.3 APP Simulation Results

The **A**PP experiment is designed to run more realistic traffic including FTP, HAS, and web. Our objective is to evaluate application performance in various workloads. We are particularly interested in knowing if the ABB scheme is able to provide benefit to higher tier application flows when system is loaded.

The first variant, called TAPPG, runs a total 100 flows. They are 30 web flows and 20 HAS flows at Tier1, 20 HAS flows at Tier2, 20 HAS flows at Tier4, and 10 FTP flows at Tier4. So there are 60 HAS flows running in total. For the Gbps channel speed, the HAS video bit rate representations are set to 1.5, 3.0, 6.0, 9.0, 12.0, and 15.0 Mbps. All flows are set to have the same uncongested path RTT of 80 ms. Table 5.26 provides the average

Table 5.26: Application Flow Throughput by Types and Tiers for Experiment TAPPG

Scheme	FTP	HAS-T1	HAS-T2	HAS-T4
DRR-CoDel	27.17	6.17	10.65	15.16
ABB-2	19.48	9.80	11.04	14.84
ABB-3	22.18	8.17	10.42	15.57
ABB-4	23.05	7.60	10.54	15.52
CoDel	21.54	11.38	11.43	11.59

FTP and HAS flow throughputs. For HAS, it gives average HAS flow throughput by tiers.

The throughput results show that CoDel, which does not support weighted allocation, gives no benefit to HAS users for subscribing to high tiers. Tier aware DRR-CoDel does allocate more bandwidth for higher tier HAS flows than lower tier HAS flows. The ABB scheme is able to approximate the allocations of DRR-CoDel, supporting the economic model behind tiered service quality levels.

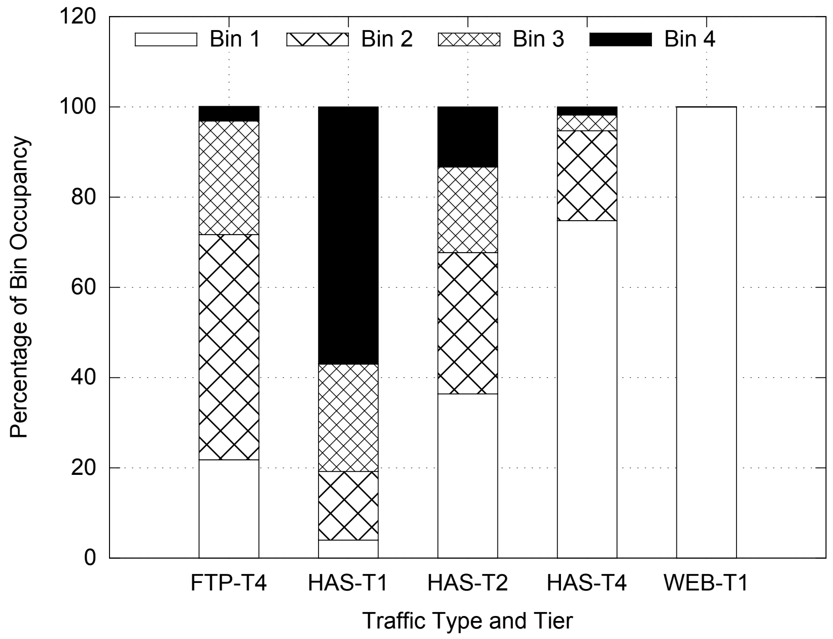


Figure 5.7: TAPPG - Bin occupancy in percentage of time for each tier and traffic type

To provide insight over how the ABB scheme allocates more throughput to higher tier HAS flows, we calculated the percentage of time for each tier and type of traffic spent in

Table 5.27: HAS Performance By Tiers for Experiment TAPPG: Video Play Rate (Mbps) / Average Adaptation Count Per Hour

Scheme	HAS-T1	HAS-T2	HAS-T4
DRR-CoDel	5.71 / 15.0	9.46 / 13.0	13.44 / 18.1
ABB-2	8.58 / 131.2	10.13 / 61.6	13.28 / 31.9
ABB-3	7.25 / 106.7	9.81 / 61.6	13.94 / 19.2
ABB-4	7.04 / 74.5	9.76 / 55.1	13.87 / 17.0
CoDel	10.23 / 131.9	10.54 / 110.7	10.46 / 116.3

each bin. Under ABB-4, the results are given in Figure 5.7. For example, Tier4 HAS flows spent 74.8% of time in bin 1, 19.9% of time in bin 2, 3.5% of time in bin 3, and 1.8% of time in bin 4. As we can see, due to their low demand for bandwidth, web flows were placed in bin 1 (the lowest bin in terms of bandwidth consumption) all the time. Among the three HAS tiers, the HAS flows of higher tier were placed in the lower bins for longer time to receive better bandwidth allocation. As the FTP flows of Tier4 consume more bandwidth than the HAS flows at the same tier, the FTP flows spends relatively less time in bin 1 so that the demand of HAS flows is guaranteed.

Table 5.27 provides the HAS performance results. Under CoDel, the video play rate and adaption count for HAS flows at different tiers are similar, defeating the purpose of subscribing to a higher tier. For DRR-CoDel that supports tiered scheduling, HAS flows at higher tier are able to achieve better HAS performance in terms of better play rate with low adaption count. The ABB scheme, similar to DRR-CoDel, offers better HAS video play rate and lower adaption count than CoDel. DRR-CoDel is seen to have lowest adaptation counts. This is expected for schemes that provide better short-term fairness (weighted or unweighted). As a result of that, flows under these schemes see less variations in their achieved throughput and therefore fewer adaptations.

It is of particular interest to know what the results would be for this scenario under the conventional tiering model based on max service rates. For this, we ran a second variant of the APP experiment called TAPPC. This variant runs the same set of flows with the same setup but the three tiers are the conventional service tiers with max service rates

Table 5.28: Application Flow Throughput by Types and Tiers for Experiment TAPPC

Tier Rate	150	12	50	150
Scheme	FTP	HAS		
DRR-CoDel	19.06	6.78	13.36	13.28
ABB-2	19.87	6.78	13.66	14.25
ABB-3	18.72	6.78	14.40	14.24
ABB-4	16.78	6.78	13.78	14.26
CoDel	26.77	6.72	12.55	12.92

Table 5.29: HAS Performance By Tiers for Experiment TAPPC:
Video Play Rate (Mbps) / Average Adaptation Count Per Hour

Tier Rate	12	50	150
DRR-CoDel	5.97 / 7.0	11.91 / 15.8	11.76 / 17.7
ABB-2	5.96 / 7.0	12.68 / 57.0	13.21 / 41.5
ABB-3	5.96 / 7.0	13.17 / 31.2	12.74 / 25.9
ABB-4	5.96 / 7.0	12.36 / 24.2	12.73 / 22.9
CoDel	5.96 / 7.5	11.42 / 91.8	11.73 / 82.9

set to 12, 50, and 150 Mbps as commonly used today. Table 5.28 and Table 5.29 provide the throughput and HAS performance results. With conventional tiering model, schedulers are unweighted. So the data for DRR-CoDel and ABB are the data with the schedulers configured to not use weights. The throughput and video play rate results for the middle tier show that the flows of the middle tier achieve similar performance to that of the flows at the highest tier. In some cases, the flows of the middle tier even achieve better performance than the flows at the highest tier, which totally defeats the purpose of subscribing to the best tier. Under single queue CoDel, the flows at the higher tiers exhibit the much higher degree of adaptations, which can have a negative impact over perceived video quality for higher tier HAS users. The clearest result from this experiment is the conventional tiering model is not good during the time of congestion and our new tiering model is better in maintaining service quality levels with respect to service tiers.

We also ran a third variant, called TAPPR, of the experiment with the three types of application traffic running in all three tiers simultaneously. Flows of each type are equally divided among the tiers. Each tier runs 3 FTP, 20 HAS, and 10 Web flows.

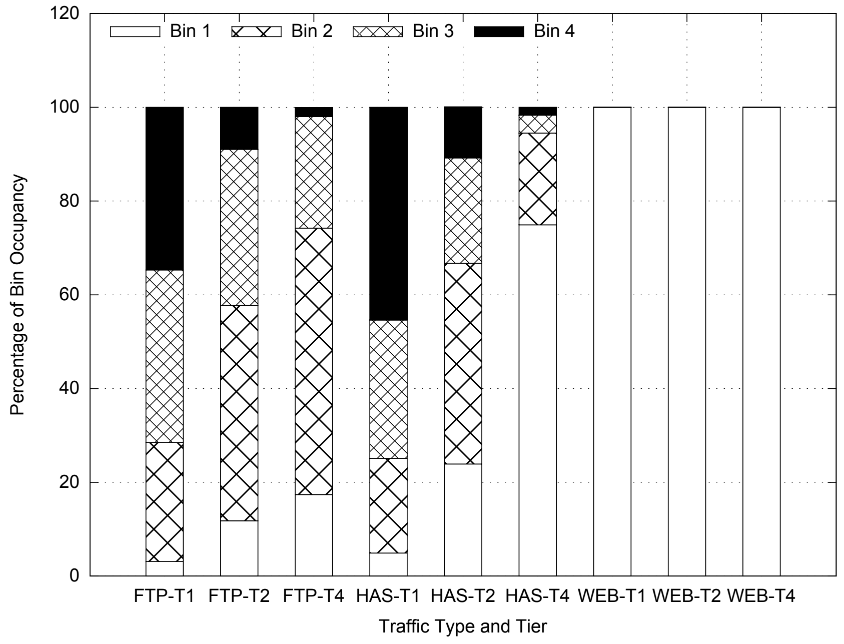


Figure 5.8: TAPPR - Bin occupancy in percentage of time for each tier and traffic type

Table 5.30 and Table 5.31 provide the application throughput and HAS performance results. The throughput results show that tier aware DRR-CoDel allocates more bandwidth for flows of higher tiers for both FTP and HAS flow classes. The throughput results also show that the ABB scheme is able to provide both FTP and HAS flows with the throughput consistent with their tiers approximating that of DRR-CoDel. The percentage of time each tier and type of traffic spend in each bin is given in Figure 5.8. Again we see that web flows irrespective of the tiers spend the whole time in bin 1 due to their low demand. Among the HAS or FTP flows of different tiers, higher tier flows were able to spend more time in lower bins to receive more bandwidth. Among the FTP and HAS flows of the same tier, FTP flows spend less time in lower bins due to their higher throughput demands. From the HAS application performance data for this experiment, it is not difficult to draw the same conclusion we drew before. Under ABB scheme, HAS flows of higher tier are able to achieve better HAS video play rate and lower adaption count than CoDel.

Finally we give the WRT results for all three APP experiment variants in Table 5.32.

Table 5.30: Application Flow Throughput by Types and Tiers for Experiment TAPPR

Scheme	FTP-T1	FTP-T2	FTP-T4	HAS-T1	HAS-T2	HAS-T4
DRR-CoDel	8.53	16.67	32.08	7.96	11.94	16.46
ABB-2	12.83	15.54	20.16	9.81	12.43	15.73
ABB-3	10.28	15.19	26.34	9.18	11.94	16.13
ABB-4	8.49	14.33	25.76	8.49	12.48	16.50
CoDel	24.63	22.80	23.13	11.74	11.66	11.46

Table 5.31: HAS Performance By Tiers for Experiment TAPPR:
Video Play Rate (Mbps) / Average Adaptation Count Per Hour

Scheme	HAS-T1	HAS-T2	HAS-T4
DRR-CoDel	7.29 / 18.2	10.54 / 15.2	14.73 / 18.0
ABB-2	8.70 / 115.8	11.39 / 56.5	14.12 / 19.2
ABB-3	8.21 / 95.1	10.94 / 25.1	14.45 / 19.6
ABB-4	7.65 / 73.0	11.16 / 43.7	14.74 / 19.1
CoDel	10.63 / 114.7	10.66 / 129.0	10.36 / 111.7

For the low bandwidth web flows, under all scenarios, we see reasonably good web performance in terms of web response time (WRT). It is worth noting that the web model we use is not a good representation of modern commercial web flows (see Section 4.2.3). So such conclusion is most likely not true for modern web flows.

Table 5.32: Web Performance for All APP Experiment Variants:
Web Response Time (Seconds) / Stddev

Experiment	TAPPG	TAPPC	TAPPR
DRR-CoDel	0.196 / 0.061	0.187 / 0.059	0.194 / 0.061
ABB-2	0.185 / 0.062	0.183 / 0.063	0.184 / 0.061
ABB-3	0.187 / 0.063	0.183 / 0.062	0.183 / 0.060
ABB-4	0.182 / 0.065	0.183 / 0.059	0.187 / 0.061
CoDel	0.195 / 0.060	0.200 / 0.061	0.198 / 0.060

Chapter 6

Conclusions and Future Work

In the research leading to this dissertation, we have studied the downstream bandwidth management in the context of emerging DOCSIS-based cable networks. The latest DOCSIS 3.1 (D3.1) standard is to provide unprecedented capacity at Gbps bandwidth for an access network. Once deployed, the whole landscape of broadband access will change with the number of the households having Gbps access increasing dramatically.

As cable networks are transitioning to D3.1 standard, the work that is of compelling interests to the broadband network community is apparently to understand if the existing bandwidth management strategy based on the current notion of service model is able to continue to function and provide reasonable performance. Such work defined the effort of the first phase of the research. Following the first phase, our second phase focused on addressing the combined problem of fairness and bufferbloat, which were properly identified in the first phase. While a solution based on fair queueing is possible, the community has been hesitant in adopting such approach due to its high complexity and cost. We proposed an approximate bandwidth management model to address the combined problem of fairness and bufferbloat with low complexity and low cost. The approximate bandwidth management model is based on quantization and binning techniques that have been the latest research trend in reducing scheduling complexity. Following the model, we further implemented a scheduling scheme based on adaptive bandwidth binning (ABB). Detailed comparative studies and results of the research were provided in the previous two chapters. In the following, we conclude with our results and remarks. We also end with a section for potential future work.

6.1 Concluding Remarks

For the first phase of the research, we assumed that the current standard practice of bandwidth management in DOCSIS is to use a single aggregate queue for all flows to be served FCFS after a regulator process used to support service tiering. The single queue is managed by a simple DT queue manager. We also assumed that delay-based AQMs or other AQMs can be used. Under these assumptions, we studied the effectiveness of these queue managers in managing fairness and application performance. The result is clear that DT is not desirable. Delay-based AQMs are effective in controlling queueing delays and provide strong isolations among responsive flows. However, single queue delay-based AQMs do not address TCP RTT unfairness issue nor do they protect responsive flows from unresponsive high bandwidth UDP flows.

For the second phase of the study, we addressed the combined problem of fairness and bufferbloat using an idea we call adaptive bandwidth binning (ABB). ABB is positioned in between state-aware AQM's designed to isolate mice from elephant flows and low complexity packet scheduling approximations to fair queueing. We apply the idea in the context of multi-tiered, emerging DOCSIS-based cable networks. We assume that the weighted max-min fairness is the desirable criterion to allocate more bandwidth for higher tiered users. The state-aware AQM will likely not provide sufficiently granular isolocation while a fair queueing approximation is generally viewed as overly complex.

ABB uses a fixed number of bins and periodically (re-)maps the flows into these bins based on flow consumption levels. Flows in the same bin share a queue managed by CoDel for low queueing latency and are scheduled in a FCFS manner for low complexity and cost. An outer scheduler schedules the bins. Due to the number of bins being limited, the scheduler can be sufficiently simple to implement. With the ABB scheme, bins are scheduled with a weighted DRR scheduler where each bin is given a weight according to the flows (along with their weights) classified into the bin. We addressed the service tiering by assuming flows subscribe to different service quality levels and the flow weights are assigned according to such quality levels. The bandwidth allocation is then to approximate weighted max-min fair allocation in the tiered scenario. We support the service tiering by normalizing flow consumption levels with flow weights.

Through simulation studies, we compared the ABB scheme with DRR-CoDel, SFQ-CoDel and single queue CoDel. DRR-CoDel is an representative of the reference approach and is included as a reference point. SFQ-CoDel is multi-queue based approach that cannot handle flow weights but works well in a single tier environment. Single queue based CoDel was also included for reference purpose.

We first analyzed the ABB scheme in the single tier environment where all flows are in the same tier with same weight. Our results and analysis indicated that the ABB scheme with its use of just a few bins is able to provide reasonable performance in terms of fairness and latency property close to that of DRR-CoDel and SFQ-CoDel. The results showed that the multi-queue based ABB scheme is able to ameliorate the TCP RTT unfairness issue that impacts single queue management schemes.

In a multi-tier environment, our results and analysis showed that the ABB scheme is able to provide weighted fairness close to that of DRR-CoDel. Due to lack of support for flow weights, both SFQ-CoDel and single queue CoDel are unable to support flow tiering. We also considered the scenario in which responsive TCP flows and high bandwidth unresponsive flows compete. We found the ABB scheme with its use of sufficient number of bins is able to provide certain degree of UDP isolation. We further analyzed application performance with experiments that involved realistic types of traffic including FTP, HAS, and web flows. We found that the ABB scheme is able to provide better HAS application performance for subscribers that subscribe to higher service tiers. On the contrary, conventional service tiering based on max service rates faces difficulty in providing better service quality for high tier users.

In summary, the ABB scheme is a low complexity and cost solution. With CMTS already measuring flow consumptions, its use of flow consumption levels for binning does not incur extra cost. It is shown to be effective in providing long term weighted max-min fairness and optimal latency property close to that of a reference approach.

6.2 Future Work

The research described in this dissertation proposed a significantly new approach for managing bandwidth. Our work should be viewed simply as foundational. While the results presented do show promise, there is a significant number of basic questions and issues that must be addressed. These issues include at least the following:

- More precisely define and quantify fairness outcomes.
- Explore the design space surrounding the choice of the flow consumption sliding window time scale and the scheduling interval.

The future work should be plenty. The direction of this research has been on the bandwidth management in the downstream direction only. We recognize upstream bandwidth management is an integral part of the overall bandwidth management for the emerging DOCSIS networks. Thus the future work should involve bandwidth management in the upstream direction as well.

Future work may also include the refinement and optimization of the ABB scheme. We also recognize some of the shortcomings or potential shortcomings of the ABB scheme. For example, the scheme may be further optimized to provide better fairness through better binning strategies. One of the binning strategies we actively considered was to use the max-min fair share as the bandwidth boundary for the first bin. This would help isolate all non-backlogged flows into the first bin to have more bins for the isolation of other flows or reduce the number of bins needed. But estimating the fair share based on actual flow consumption was proven to be difficult. It may need a combination of other measurements to be successful.

The ABB scheme is found to be not as effective as we hoped in isolating unresponsive high bandwidth UDP flows to provide better fairness when such unresponsive flows are involved. It is inherently hard in a binning scheme to ensure fairness when unresponsive flows are involved. For example, in the scenario where an unresponsive high bandwidth UDP flow is binned with responsive flows in the same bin, the achieved throughput of the responsive flows is at the mercy of the unresponsive flow. Even in the scenario where all unresponsive flows are isolated in separate bins, different unresponsive flows with different sending rates cannot share fairly in the same bin. It would require one bin per unresponsive flow to ensure ideal fairness.

We found in ABB that the scheduling disruption time can be significant due to unresponsive high bandwidth UDP flows. With the involvement of unresponsive flows, the bins may not be well balanced in terms of their lengths, which translates to longer disruption time. This is somehow related to the head-drop CoDel algorithm that allows a queue to build up in one bin. A tail-drop version of the CoDel or PIE may be more effective when used with each bin.

While it may not seem to be hard to identify unresponsive high bandwidth flows, how to handle them is not a simple issue once they are identified. The handling may well depend on specific goal. If the goal is simply to provide protection for responsive flows, the unresponsive flows can be isolated to a single bin. But if the goal also includes providing fairness among all flows including unresponsive flows, it becomes much harder to do.

We plan on addressing these issues in the near future.

Bibliography

[1] The network simulator – ns-2. `http://www.isi.edu/nsnam/ns/`.

[2] Saamer Akhshabi, Lakshmi Anantakrishnan, Ali C Begen, and Constantine Dovrolis. What happens when HTTP adaptive streaming players compete for bandwidth? In *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, pages 9–14. ACM, 2012.

[3] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 157–168. ACM, 2011.

[4] F. Baker and G. Fairhurst. IETF recommendations regarding active queue management. `https://tools.ietf.org/html/rfc7567`, 2015.

[5] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. A quest for an Internet video quality-of-experience metric. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 97–102. ACM, 2012.

[6] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *ACM SIGMETRICS Performance Evaluation Review*, volume 26, pages 151–160. ACM, 1998.

[7] Jon C. R. Bennett and Hui Zhang. WF^{2}Q: worst-case fair weighted fair queueing. In *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, volume 1, pages 120–128. IEEE, 1996.

[8] Cable Television Laboratories, Inc. Data-Over-Cable Service Interface Specifications: Operations Support System Interface Specification. `http://www.cablelabs.org/cablemodem/specifications/specifications20.html`, 2008.

[9] Cable Television Laboratories, Inc. Data-Over-Cable Service Interface Specifications: DOCSIS 3.1 MAC and Upper Layer Protocols Interface Specification, 2015. CM-SP-MULPIv3.1-I06-150611.

[10] Shun Y Cheung and Corneliu S Pencea. BSFQ: Bin sort fair queueing. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1640–1649. IEEE, 2002.

[11] David Clark and John Wroclawski. An approach to service allocation in the internet. Technical report, Internet Draft draft-clark-diff-svc-alloc-00. txt, July 1997, also talk by D. Clark in the Int-Serv WG at the Munich IETF, 1997.

[12] Robert G Cole and Joshua H Rosenbluth. Voice over IP performance monitoring. *ACM SIGCOMM Computer Communication Review*, 31(2):9–24, 2001.

[13] Nicola Cranley, Philip Perry, and Liam Murphy. User perception of adapting video quality. *International Journal of Human-Computer Studies*, 64(8):637–647, 2006.

[14] Mark E Crovella and Azer Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *Networking, IEEE/ACM Transactions on*, 5(6):835–846, 1997.

[15] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. Elastic: A client-side controller for dynamic adaptive streaming over HTTP (DASH). In *Packet Video Workshop (PV), 2013 20th International*, pages 1–8. IEEE, 2013.

[16] Luca De Cicco and Saverio Mascolo. An adaptive video streaming control system: Modeling, validation, and performance evaluation. *Networking, IEEE/ACM Transactions on*, 22(2):526–539, 2014.

[17] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM Computer Communication Review*, volume 19, pages 1–12. ACM, 1989.

[18] Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, and Hui Zhang. Understanding the impact of video quality on user engagement. *Communications of the ACM*, 56(3):91–99, 2013.

[19] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.

[20] Constantinos Dovrolis, Dimitrios Stiliadis, and Parameswaran Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *ACM SIGCOMM Computer Communication Review*, 29(4):109–120, 1999.

[21] Mandouh Droubi, Nasser Idirene, and Charles Chen. Dynamic bandwidth allocation for the hfc docsis mac protocol. In *Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on*, pages 54–60. IEEE, 2000.

[22] Eric Duzamet. Fair queue CoDel packet scheduler. `https://dev.openwrt.org/browser/trunk/target/linux/generic/patches-3.3/042-fq_codel-Fair-Queue-Codel-AQM.patch?rev=33560`, 2012.

[23] Zyad Dwekat and George N Rouskas. A practical fair queuing scheduler: Simplification through quantization. *Computer Networks*, 55(10):2392–2406, 2011.

[24] G Fairhurst, A Sathiaseelan, and R Secchi. Updating tcp to support rate-limited traffic (draft-ietf-tcpm-newcwv-12). `https://tools.ietf.org/html/draft-ietf-tcpm-newcwv-12`, 2015.

[25] Federal Communications Commission. 2015 Broadband Progress Report, 2015.

[26] Anja Feldmann, Anna C Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 301–313. ACM, 1999.

[27] Wu-chang Feng, Kang G Shin, Dilip D Kandlur, and Debanjan Saha. The BLUE active queue management algorithms. *IEEE/ACM Transactions on Networking (TON)*, 10(4):513–528, 2002.

[28] Sally Floyd. RED (random early detection) queue management. `http://www.icir.org/floyd/red.html`, 2008.

[29] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking (TON)*, 7(4):458–472, 1999.

[30] Sally Floyd, Ramakrishna Gummadi, Scott Shenker, et al. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. *Preprint, available at http://www.icir.org/floyd/papers.html*, 2001.

[31] Sally Floyd and Van Jacobson. Traffic phase effects in packet-switched gateways. *ACM SIGCOMM Computer Communication Review*, 21(2):26–42, 1991.

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

[33] Manfred Georg, Christoph Jechlitschek, and Sergey Gorinsky. Improving individual flow performance with multiple queue fair queuing. In *Quality of Service, 2007 Fifteenth IEEE International Workshop on*, pages 141–144. IEEE, 2007.

[34] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark buffers in the Internet. *Queue*, 9(11):40, 2011.

[35] S Jamaloddin Golestani. A self-clocked fair queueing scheme for broadband applications. In *INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE*, pages 636–646. IEEE, 1994.

[36] T Hoeiland-Joergensen, P McKenney, D Taht, J Ghettys, and E Dumazet. FlowQueue-Codel (draft-ietf-aqm-fq-codel-02). `https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-02`, 2015.

[37] Gongbing Hong, James Martin, Scott Moser, and James Westall. Fair scheduling on parallel bonded channels with intersecting bonding groups. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 89–98. IEEE, 2012.

[38] Gongbing Hong, James Martin, and James M. Westall. On fairness and application performance of active queue management in broadband cable networks. *Computer Networks*, 91:390–406, November 2015.

[39] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, pages 225–238. ACM, 2012.

[40] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 187–198. ACM, 2014.

[41] Paul Hurley, J-Y Le Boudec, Patrick Thiran, and Mourad Kara. ABE: Providing a low-delay service within best effort. *Network, IEEE*, 15(3):60–69, 2001.

[42] Saima Jabeen, Muhammad Bilal Zafar, Ihsan Ayyub Qazi, and Zartash Afzal Uzmi. Splitbuff: Improving the interaction of heterogeneous rtt flows on the internet. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2315–2319. IEEE, 2013.

[43] Van Jacobson. Congestion avoidance and control. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 314–329. ACM, 1988.

[44] Rajendra K Jain, Dah-Ming W Chiu, and William R Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. `http://www1.cse.wustl.edu/~jain/papers/ftp/fairness.pdf`, 1984.

[45] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, pages 97–108. ACM, 2012.

[46] Srinivasan Keshav. *An Engineering Approach to Computer Networking: ATM Networks, The Internet and Telephone Network*. Addison Wesley, Reading, Massachusetts, 1997.

[47] Naeem Khademi, David Ros, and Michael Welzl. The new AQM kids on the block: Much ado about nothing? Technical report, University of Oslo, October 2013.

[48] S Shunmuga Krishnan and Ramesh K Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *Networking, IEEE/ACM Transactions on*, 21(6):2001–2014, 2013.

[49] Nicolas Kuhn, Emmanuel Lochin, and Olivier Mehani. Revisiting old friends: Is CoDel really achieving what RED cannot? In *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, CSWS '14, pages 3–8, New York, NY, USA, 2014. ACM.

[50] Wen-Kuang Kuo, Sunil Kumar, and C-CJ Kuo. Bandwidth allocation and traffic scheduling for docsis systems with qos support. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 2, pages 1990–1994. IEEE, 2002.

[51] Jean-Yves Le Boudec. Rate adaptation, congestion control and fairness: A tutorial, 2012.

[52] Zhi Li, Ali C Begen, Joshua Gahm, Yufeng Shan, Bruce Osler, and David Oran. Streaming video over HTTP with consistent quality. In *Proceedings of the 5th ACM Multimedia Systems Conference*, pages 248–258. ACM, 2014.

[53] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali Begen, and David Oran. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *Selected Areas in Communications, IEEE Journal on*, 32(4):719–733, 2014.

[54] Dong Lin and Robert Morris. Dynamics of random early detection. In *ACM SIGCOMM Computer Communication Review*, volume 27, pages 127–137. ACM, 1997.

[55] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. Rate adaptation for adaptive HTTP streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 169–174. ACM, 2011.

[56] Chenghao Liu, Imed Bouazizi, Miska M Hannuksela, and Moncef Gabbouj. Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network. *Signal Processing: Image Communication*, 27(4):288–311, 2012.

[57] James Martin, Gongbing Hong, and James Westall. Managing fairness and application performance with active queue management in docsis-based cable networks. In *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, pages 9–14. ACM, 2014.

[58] Jim Martin, Yunhui Fu, Nicholas Wourms, and Terry Shaw. Characterizing Netflix bandwidth consumption. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 230–235. IEEE, 2013.

[59] Jim Martin and James Westall. A simulation model of the DOCSIS protocol. *Simulation*, 83(2):139–155, 2007.

[60] Matthew Mathis. Reflections on the TCP macroscopic model. *ACM SIGCOMM Computer Communication Review*, 39(1):47–49, 2008.

[61] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997.

[62] Martin May, Jean Bolot, Christophe Diot, and Bryan Lyles. Reasons not to deploy RED. In *Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on*, pages 260–262. IEEE, 1999.

[63] Paul E McKenney. Stochastic fairness queueing. In *INFOCOM'90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies.'The Multiple Facets of Integration'. Proceedings., IEEE*, pages 733–740. IEEE, 1990.

[64] J. Mills, J. Livingood, R. Woundy, T. Klieber, and C. Bastian. Comcast's protocol-agnostic congestion management system, 2010.

[65] Scott A. Moser. *Downstream Resource Allocation in DOCSIS 3.0 Channel Bonded Networks*. PhD thesis, Clemson, SC, USA, 2011. AAI3469539.

[66] Aisha Mushtaq, Asad Khalid Ismail, Abdul Wasay, Bilal Mahmood, Ihsan Ayyub Qazi, and Zartash Afzal Uzmi. Rethinking buffer management in data center networks. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 575–576. ACM, 2014.

[67] John Nagle. On packet switches with infinite storage. *Communications, IEEE Transactions on*, 35(4):435–438, 1987.

[68] Sajid Nazir, Ziaul Hossain, Raffaello Secchi, Matthew Broadbent, Andreas Petlund, and Gorry Fairhurst. Performance evaluation of congestion window validation for DASH transport. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, page 67. ACM, 2014.

[69] Pengpeng Ni, Ragnhild Eg, Alexander Eichhorn, Carsten Griwodz, and Pål Halvorsen. Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 463–472. ACM, 2011.

[70] K Nichols. SFQ-CoDel ns-2 source code. `http://www.pollere.net/Txtdocs/`, 2013.

[71] K. Nichols and V. Jacobson. Controlled delay active queue management (draft-nichols-tsvwg-codel-01). `http://tools.ietf.org/html/draft-nichols-tsvwg-codel-01`, 2013.

[72] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar. Controlled delay active queue management (draft-ietf-aqm-codel-01). `https://tools.ietf.org/html/draft-ietf-aqm-codel-01`, 2015.

[73] Kathleen Nichols and Van Jacobson. Controlling queue delay. *Communications of the ACM*, 55(7):42–50, 2012.

[74] Dessislava Nikolova and Chris Blondia. Bonded deficit round robin scheduling for multi-channel networks. *Computer Networks*, 55(15):3503–3516, 2011.

[75] R. Pan, P. Natarajan, G. White, B. VerSteeg, M.S. Prabhu, C. Piglione, and V. Subramanian. PIE: A lightweight control scheme to address the bufferbloat problem (draft-ietf-aqm-pie-02). `https://tools.ietf.org/html/draft-ietf-aqm-pie-02`, 2015.

[76] Rong Pan, Lee Breslau, Balaji Prabhakar, and Scott Shenker. Approximate fairness through differential dropping. *ACM SIGCOMM Computer Communication Review*, 33(2):23–39, 2003.

[77] Rong Pan, Preethi Natarajan, Chiara Piglione, Mythili Suryanarayana Prabhu, Vijay Subramanian, Fred Baker, and Bill VerSteeg. PIE: A lightweight control scheme to address the bufferbloat problem. In *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on*, pages 148–155. IEEE, 2013.

[78] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 942–951. IEEE, 2000.

[79] Abhay Kumar J. Parekh. *A generalized processor sharing approach to flow control in integrated services networks.* PhD thesis, Massachusetts Institute of Technology, 1992.

[80] Maxim Podlesny and Sergey Gorinsky. Leveraging the rate-delay trade-off for service differentiation in multi-provider networks. *Selected Areas in Communications, IEEE Journal on*, 29(5):997–1008, 2011.

[81] Sriram Ramabhadran and Joseph Pasquale. The stratified round robin scheduler: design, analysis and implementation. *IEEE/ACM Transactions on Networking (TON)*, 14(6):1362–1373, 2006.

[82] Vincent Rosolen, Olivier Bonaventure, and Guy Leduc. A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic. *ACM SIGCOMM Computer Communication Review*, 29(3):23–43, 1999.

[83] Sandvine Incorporated ULC. Global Internet phenomena report, 2015.

[84] Neel Shah, Demetres Kouvatsos, Jim Martin, and Scott Moser. A tutorial on DOCSIS: protocol and performance models. In *Proceedings of the international working conference on performance modeling and evaluation of heterogeneous networks, Ikley, UK*, 2005.

[85] Neelkamal P Shah, Demetres D Kouvatsos, Jim Martin, and Scott Moser. On the performance modelling and optimisation of DOCSIS HFC networks. In *Network performance engineering*, pages 682–715. Springer, 2011.

[86] Madhavapeddi Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. In *ACM SIGCOMM Computer Communication Review*, volume 25, pages 231–242. ACM, 1995.

[87] Madhavapeddi Shreedhar and George Varghese. Efficient fair queuing using deficit round-robin. *Networking, IEEE/ACM Transactions on*, 4(3):375–385, 1996.

[88] Subhash Suri, George Varghese, and Girish Chandranmenon. Leap forward virtual clock: a new fair queuing scheme with guaranteed delays and throughput fairness. In *INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, volume 2, pages 557–565. IEEE, 1997.

[89] Ben Teitelbaum and Stanislav Shalunov. Why premium IP service has not deployed (and probably never will). *Internet2 QoS Working Group Informational Document*, 2002.

[90] Guibin Tian and Yong Liu. Towards agile and smooth video adaptation in dynamic HTTP streaming. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 109–120. ACM, 2012.

[91] Jonathan Turner. New directions in communications (or which way to the information age?). *IEEE communications Magazine*, 24(10):8–15, 1986.

[92] Greg White and Joey Padden. Preliminary study of CoDel AQM in a DOCSIS network. Technical report, Technical Report, CableLabs, 2012.

[93] Greg White and Dan Rice. Active queue management algorithms for DOCSIS 3.0, 2013. CableLabs Technical Report, April 2013.

[94] Lin Xue, Suman Kumar, Cheng Cui, Praveenkumar Kondikoppa, Chui-Hui Chiu, and Seung-Jong Park. AFCD: An approximated-fair and controlled-delay queuing for high speed networks. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–7. IEEE, 2013.

[95] Xin Yuan and Zhenhai Duan. FRR: a proportional and worst-case fair round robin scheduler. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 831–842. IEEE, 2005.

[96] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–1396, 1995.