Computer Networks 150 (2019) 150–169

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Adaptive bandwidth binning for bandwidth management

Gongbing Hong[a,*], James Martin[b], James Westall[b]

[a] Computer Science Dept., Georgia College & State University, 301 Atkinson Hall, Milledgeville, GA 31061, USA
[b] School of Computing, Clemson University, 100 McAdams Hall, Clemson, SC 29634, USA

ARTICLE INFO

Article history:
Received 11 January 2018
Revised 27 November 2018
Accepted 31 December 2018
Available online 5 January 2019

Keywords:
Cable access
Active queue management (AQM)
Packet scheduling algorithms
Network performance evaluation

ABSTRACT

In this paper, we present a novel bandwidth management scheme that we call *adaptive bandwidth binning (ABB)*. ABB is presented in the context of a DOCSIS cable network, but it has obvious applicability to downstream service on any shared medium access network in which all downstream traffic is scheduled by a single headend device such as a CMTS or wireless base station or access point.

ABB is capable of providing approximate weighted max-min fair sharing of downstream bandwidth via a low-overhead scheduler that requires only a small number of permanently allocated queues. A modern delay-based active queue management (AQM) technique is employed to control delays.

The performance of ABB is evaluated via *ns-2* simulations in which workloads include FTP, HTTP-based adaptive streaming (HAS), and web traffic targeted for different tiered service quality levels. Our results show that ABB is able to provide approximate weighted max-min fair bandwidth allocation among responsive high bandwidth flows while isolating them from low bandwidth and latency sensitive flows. The use of CoDel in each ABB queue is effective in managing latency as required. The use of flow weights in ABB supports service tiering in which subscribers pay more for higher service rates.

1. Introduction

Cable based access networks are one of the dominant broadband access network technologies. In contrast to wireless networks whose speeds are constrained by governmental allocation of RF spectrum, cable networks have minimal constraints and consequently, in the past decade, have seen huge increases in available service rates over the 2 GHz of spectrum provided by a coaxial cable medium.

The Data Over Cable Service Interface Specification (DOCSIS) standard defines the operation of hybrid fiber/cable (HFC) networks. DOCSIS networks originally used 6 or 8 MHz cable TV channels to provide Internet connectivity to homes and small businesses. The 6 MHz bandwidth, used in USA, coupled with cable's excellent SINR delivers an aggregate downstream bit rate of approximately 38 Mbps at the network layer. The DOCSIS 3.0 standard allows operators to bond multiple channels together to form large virtual channels with effective data rates that can extend beyond 100 Mbps.

In the DOCSIS 3.1 standard (D3.1), the 6 and 8 MHz bottleneck was eliminated and downstream channels of up to 192 MHz bandwidth are now supported. A 192 MHz channel can support an aggregate bit rate exceeding 1 Gbps. The thirty-fold increase in bandwidth raises questions regarding the adequacy of existing DOCSIS scheduling and queue management algorithms. Indeed, the D3.1 standard now requires AQM for both downstream and upstream traffic.

In this paper we consider the downstream bandwidth management problem in shared medium access networks. One motivation for the work is the transition to the D3.1 standard [1] and the consequent need to manage the fair sharing of 1 Gbps+ among the customers sharing a cable link. Although our simulations employ DOCSIS networks, our approach is applicable to downstream scheduling in any shared medium access network.

A complementary motivation is the relatively recent research in the networking community in the area of active queue management (AQM) [2]. Where the focus of the early research (e.g., RED and its descendants) was queue length management, the focus of the more recent delay-based AQM algorithms is to control latency. Both CoDel [3,4] and PIE [5,6] implement a pro-active loss process designed to maintain a statistical packet latency target and avoid the well known problems associated with "bufferbloat." [7].

* Corresponding author.
E-mail addresses: gongbing.hong@gcsu.edu (G. Hong), jmarty@clemson.edu (J. Martin), westall@clemson.edu (J. Westall).

https://doi.org/10.1016/j.comnet.2018.12.019
1389-1286/

The main contribution of the paper is the novel packet scheduling and queue management method we call adaptive bandwidth binning (ABB). ABB maintains a small fixed number of queues, referred to as *bins*. For the purposes of this paper a *flow* can be considered to be a TCP connection or a UDP pseudo-connection, but, in practice, it could be implemented as DOCSIS *service flow*.

Flows having similar recent downstream bit rates are assigned to a bin associated with that approximate bit rate. The set of flows associated with a particular bin are called its *flow group*. Packets in each flow group are queued in the group's associated bin and scheduled first-come-first-served (FCFS). Each bin is assigned a weight based upon the number and individual weights of the flows that it holds.

An outer scheduler services the bins in a weighted deficit round-robin (WDRR) [8] manner. The ABB system adapts to changing workloads by periodically remapping flows to bins when flow consumption levels change. By normalizing recent bandwidth consumption with weights associated with the service tier of an individual flow, ABB can support service tiering.

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') is well known, to the best of our knowledge, we are the first to evaluate an adaptive bandwidth binning scheme that provides approximate *weighted* fairness with low complexity and cost.

The remainder of the paper is organized as follows. Section 2 summarizes relevant background and related research. Section 3 presents the details of our implementation of adaptive bandwidth binning. Section 4 details our simulation setup. Section 5 provides the rationale for our choice of experimental parameters. The Sections 6 and 7 contain simulation results for single service tier and multiple service tiers respectively. Finally Section 8 provides our conclusions.

2. Background and related work

Bandwidth management is a core task of a subscription based access network. We consider it in the context of management of downstream traffic from an *access point* (e.g. a DOCSIS Cable Modem Terminating System (CMTS)) over a shared medium to a collection of downstream customers.

The objective of a bandwidth management system is to ensure that bandwidth is allocated fairly to customers in accordance with their service contracts. Aspects of the task include fairly managing degradation of service when total demand exceeds capacity and preventing intentionally abusive customers from degrading the performance of others.

Historically, fairness in the Internet has largely been based on "TCP fairness" [9]. Nevertheless, it is well known that bandwidth sharing among competing TCP flows favors flows with shorter round trip times (RTTs). It is also possible for users to employ parallel TCP connections to defeat TCP fairness, and obviously application protocols operating over UDP transport are immune to all constraints associated with TCP fairness. Therefore, DOCSIS access networks have historically used supplemental bandwidth management systems such as Comcast's protocol-agnostic congestion management system [10]. The objective of such systems is to ensure some form of fair sharing of bandwidth among customers.

A widely accepted fairness criterion is *max-min* fairness. An allocation is max-min fair if it is not possible to increase the capacity allocated to any customer in the system without decreasing the rate allocated to another customer who is receiving an already lower or equal rate [11].

A bandwidth management system is comprised of three major activities: flow management; queue management; and scheduling [12].

A flow is a related collection of packets flowing between a pair of endpoints. Depending upon the implementation, a flow could be defined as a TCP connection or UDP pseudo-connection, all the traffic flowing between a pair of IP end points, an architecture dependent entity such as a DOCSIS service flow, or even all the traffic flowing upstream or downstream. A flow can be considered to be a bi-directional or uni-directional entity. Flow management activities include identifying the initiation of new flows and the termination of existing flows, mapping flows to packet queues, and identifying the flow associated with each arriving packet.

Queue management activities include creating and destroying packet queues, placing arriving packets into the proper location in the associated queue, and identifying which packets to drop in times of congestion.

The scheduling system typically is responsible for identifying which queue to service next and removing its head packet for transmission.

In general, the more effective a bandwidth management system is at providing flow isolation and fairness, the more complex it is. A tremendous amount of research has explored ways to reduce complexity and computational overhead while maintaining desired properties such as good isolation, fairness, and low latency. Fig. 1 provides a number of schemes developed over decades and their relative complexity to each other.

2.1. Survey of previous research

In this section we provide a brief review of research in bandwidth management. The simplest bandwidth management systems employ a single statically allocated queue with drop-tail queue management and FCFS scheduling. In this system all downstream packets are considered to constitute a single flow. The scheduling function is simply the removal of the head packet from the queue. Obviously, this system cannot provide any flow isolation nor enforce any sort of fairness. Its strength is its simplicity and it is commonly used in home WiFi access points and other small network switches.

Fair queuing (FQ) was originally proposed by Nagle [13] and then extended by many others. These extensions constitute a class of bandwidth management systems that are capable of providing excellent flow isolation and max-min fairness. However, these systems require a separate queue for each flow. Consequently both flow and queue management are complex. The underlying scheduling algorithms can be broadly divided into two categories: timestamp based and frame or round based algorithms.

The ideal fair queuing algorithm is generalized processor sharing (GPS) [14] which is based on a fluid flow model. It assumes the traffic is infinitely divisible. Thus GPS can serve an arbitrarily small amount of data to each backlogged flow within any finite time interval. GPS therefore can provide ideal flow isolation and fairness. GPS, however, is not implementable because real packets carry finite header overhead and are not infinitely divisible.

The weighted fair queuing (WFQ) system [15] approximates GPS for finite sized packets. It maintains a virtual time clock and calculates a virtual time stamp / service tag for each packet. The scheduling component selects the packet with the smallest service tag for transmission.

While WFQ and worst case fair WFQ provide close approximation to GPS, They are computationally complex due to the need to maintain the virtual time clock. The complexity of WFQ is dominated by the virtual time stamp computations, which requires $O(n)$ time for each packet transmitted where n is the number of flows.

To address the complexity problem with WFQ, Golestani developed the self-clocked fair queuing (SCFQ) scheme [16]. Instead of using the virtual time derived from GPS system for the calculation of service tags, SCFQ uses the service tag of the packet currently

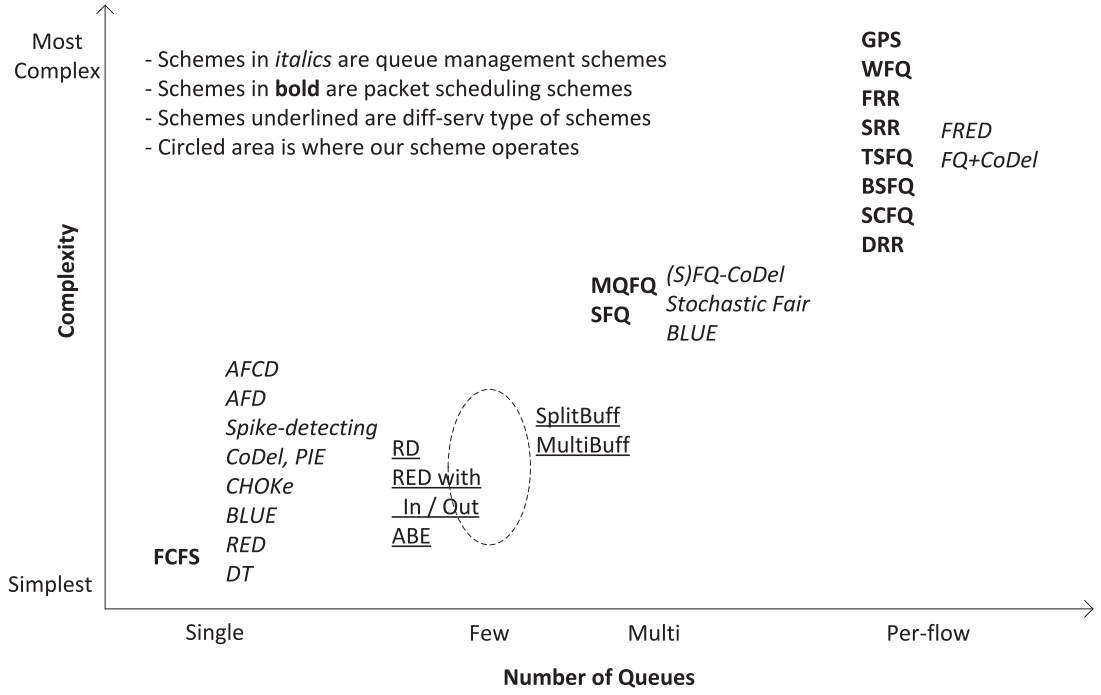


Fig. 1. Ranges of complexity of various bandwidth management methods.

receiving service as an estimate of the system virtual time. This significantly simplifies the computation of service tags while *still* maintains near optimal performance. It also helps reduce the complexity to $O(\log n)$ work per packet, which is the amount of time required for the sorting of service tags.

All GPS derived approaches are (weighted) max-min throughput fair in the limit, but have different yet bounded differences between their actual packet transmission completion times and the corresponding GPS packet transmission completion times.

The round-robin approach [13] requires one queue per flow. The scheduler maintains a circular list of queue descriptors. When a packet is to be scheduled it is selected from the next non-empty queue on the list. This avoids the sorting bottleneck of a time stamp based approach and achieves low complexity of $O(1)$. However the scheme is throughput unfair if packet sizes vary.

This unfairness is corrected in deficit round-robin (DRR) [8]. A system parameter, *max_bytes* limits the number of bytes that can be transmitted from each queue per scheduling round. Associated with each queue is a *deficit counter* which is initialized to 0. At the start of each round *max_bytes* is added to the deficit counter of each backlogged queue. When the scheduler processes a queue, it decrements the queue's deficit counter by the length of each packet transmitted. The scheduler moves onto the next backlogged queue when the queue currently being processed becomes empty or transmitting another packet would cause the deficit counter to become negative. DRR is also max-min throughput fair in the limit but has higher bounds on the maximum difference in actual packet transmission times when compared to GPS and virtual clock based transmission times.

Time stamp sorting and per-flow queuing requirement can lead to unacceptably inefficient packet scheduling in a system with a large number of high data rate flows. Various techniques to mitigate the inefficiency have been proposed. These include two level scheduling in which flows are aggregated using attributes such as flow weight and packet size. The upper level scheduler identifies the next aggregate to be scheduled and then the lower level scheduler identifies the individual flow and packet to be transmitted. Examples of this approach include SRR [17], FRR [18], TSFQ [19]. In these three flow aggregation schemes, per-flow queuing is still required.

Flow aggregation schemes that do not require per-flow queuing are commonly called *binning* systems. In a binning system packets from a group of flows share a queue (e.g., BSFQ [20] and SFQ [21]).

Because the aggregate packet flow from the upstream to the shared medium headend device may exceed the downstream bandwidth of the shared medium, queue size control is an essential element of the queue management system.

The simplest queue management scheme is drop-tail (DT). Arriving packets are dropped when the number of queued packets reaches a pre-defined capacity. While DT is appropriate for limited capacity systems such as a single WiFi Access Point, it is known to have serious shortcomings [2] in high data rate multi-flow scenarios that employ long maximum queue lengths. Consequently, some form of active queue management (AQM) has long been considered a beneficial and appropriate way to support the effectiveness of TCP fairness.

The random early detection (RED) [22] algorithm is one of the earliest AQM algorithms. RED manages a queue by randomly dropping packets when the average queue size reaches a threshold. The drop rate is linearly increased as a function of average queue size until the average queue size exceeds a second threshold and the drop probability becomes 1.

A variant of RED called 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 [23]. This adaptation is performed periodically. We refer to this parameter as the *control_interval*. More AQM schemes (e.g. BLUE [24]) were developed since then but choosing the appropriate set of parameters for these schemes turns out to be difficult [2].

In the 25 years since the introduction of RED, there have been substantial changes in the nature of Internet traffic. Interactive audio and video applications such as Skype are very intolerant of significant latency and streaming video is intolerant of highly variable latency. Consequently, long queue lengths and their associated latency under simple drop tail FCFS queue management are no longer tolerable, and AQM systems designed to limit queue res-

idency time to a target maximum have been introduced in the last decade.

Controlled Delay (CoDel) [3] and Proportional Integral Controller Enhanced (PIE) [5] are two delay-based AQMs designed to directly control queuing latency. Both proactively drop packets to ensure average packet queuing 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 [3] and [5] respectively.

The PIE algorithm performs early packet drops as packets arrive at the queue. The CoDel algorithm, as originally proposed in [3,25] performs packet drops at the time packets are dequeued for transmission.

The work presented in this research is based in part on an open source implementation of drop-tail variant of CoDel (which we refer to as CoDel-DT) that was contributed to *ns-2* by CableLabs.[1]

AQM algorithms may drop packets from all flows indiscriminately with same probability. This may result in unfair sharing of the link capacity. Delay-based AQMs simultaneously addressing the fairness issue have also been proposed. One example is AFCD [26] that blends AFD and CoDel. A flow whose sending rate exceeds its fair share will be controlled by CoDel with a shorter delay target. Thus, assuming TCP end points, the resulting higher drop rate for such a flow causes the source to reduce its sending rate.

FQ-CoDel [27] is another example that simultaneously addresses unfairness and excessive queue size. An implementation of FQ-CoDel for ns-2 [28] is called SFQ-CoDel [29]. Regardless of the number of active flows, (S)FQ-CoDel randomly 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. 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 Fig. 1.

3. Adaptive bandwidth binning

We now present our approximate bandwidth management system. It has considerably less complexity and operational overhead than per-flow approximations of weighted fair queuing, and it can approximate WFQ's weighted max-min fair allocation. Our system is also effective with a small number of statically allocated bins (typically 3 to 5) and requires no per flow queuing at all.

The system is illustrated in Fig. 2. The static set of bins is labeled b_j, $1 \le j \le k$. The system tracks a variable number of flows f_i, $1 \le i \le n$. Associated with each flow is a weight w_i.

As previously described, the precise definition of flow is flexible and is implementation dependent. The adaptive bandwidth binning system maintains a mapping M: $f_i \rightarrow b_j \ \forall i$ that maps each flow to a specific bin. Periodically, at fixed time intervals, τ, the monitor process, shown at the top of the figure, reconstructs the mapping based upon the downstream transmission rate realized by each flow in the recent past.

A two-level packet scheduling framework is employed. Each bin has a single associated queue in which the packets of all presently assigned flows are scheduled FCFS as augmented by drop-tail CoDel. Service to the bins is controlled by the inter-bin scheduler called the *outer* scheduler. The outer scheduler can use any weighted fair queuing scheduling algorithm. Our implementation uses weighted DRR.

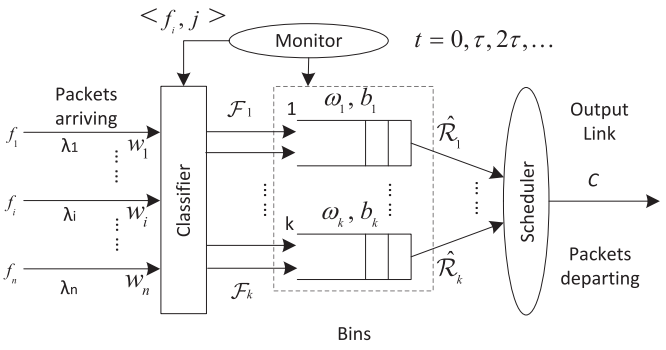


Fig. 2. Approximate bandwidth management model.

The weight for each bin j is given by $\omega_j = \sum_{f_i} w_i$ where the sum is taken over all flows, f_i, presently mapped to bin j and w_i is the weight of f_i. Each time the flows are remapped, ω_j changes accordingly.

3.1. ABB and weighted fair queuing

Let C be the capacity of the shared downstream medium. Assume that each bin, b_j, has an associated bandwidth consumption cap C_j. Without loss of generality, assume $C_j \le C_{j+1} \ \forall j$ so that bins are arranged in increasing order of bandwidth consumption caps. Also let $C_0 = 0$ and $C_k = C$. The details of how the values $C_j \{0 < j < k\}$ are established will be addressed subsequently.

Let $\hat{r}_i^t$ represent the approximate realized downstream transmission rate of flow f_i at time t. It is computed as an exponentially decaying auto regressive moving average. Let r_i^t be the measured transmission rate of flow i in the interval $[t-\tau, t]$. Then $\hat{r}_i^t = 0.6\hat{r}_i^{t-\tau} + 0.4r_i^t$.

At each remapping time, t, flow f_i is mapped to bin j if $C_{j-1} < \hat{r}_i^t/w_i \le C_j$. We refer to $\hat{r}_i^t/w_i$ as the normalized flow transmission rate of f_i at time t. In this way, flows of similar normalized consumptions are classified into the same bin.

The outer and inner schedulers work together to provide approximate global max-min fair allocation on a per flow basis. Let $\hat{\mathcal{R}}_j^t$ be the aggregate transmission rate from bin j at time t. Note that even though the per flow caps C_j are strictly increasing with j, this is not necessarily true of $\hat{\mathcal{R}}_j^t$ which is proportional to the number of flows assigned to a bin.

The weighted DRR outer scheduler guarantees that the values $\hat{\mathcal{R}}_j^t$ represent weighted max-min fair sharing of the aggregate capacity C with respect to the bin weights ω_j. The binning algorithm ensures the inner FCFS schedulers are dealing with flows of similar bandwidth demand.

The following idealized example serves to illustrate that the hierarchical weighted scheduling process works as intended.

Assume $C = 100$. Each flow in bin j has identical demand r_j as shown in Table 1 and a weight $w_i = 1.0$. Then the aggregate demand for each bin $\hat{\mathcal{R}}_j$ is r_j multiplied by the number of flows in bin j and ω_j is equal to the number of flows in bin j.

Processing this data using weighted per-bin loads with the standard weighted max-min fair algorithm produces the correct per bin allocations and consequently the correct per flow allocations shown in Table 2.

[1] Available at http://sourceforge.net/p/nsnam/patches/24/

Table 1
Bin assignments and flow demands.

Bin	Flows	r_j	$\hat{\mathcal{R}}_j$
1	20	3	60
2	3	6	18
3	1	10	10
4	2	20	40

Table 2
Weighted max-min fair bin and flow allocation.

Bin	Flows	Bin alloc	Flow alloc
1	20	60.000	3.000
2	3	18.000	6.000
3	1	7.333	7.333
4	2	14.667	7.333

3.2. Implementation of ABB

In this section we present the design of our ABB system. Low level details of our *ns-2* implementation can be found in [30].

3.2.1. The binning system

A small number (3 to 5) of bins are statically allocated. A single FCFS packet queue is associated with each bin. Capacity of each queue is managed using independent instances of drop-tail CoDel. Parameters for CoDel such as target delay and control interval are those given in [3,4].

Each bin has an associated value of C_j, the upper bound on weighted bandwidth consumption of member flows, and ω_j the sum of all member flow weights. C_j and ω_j are dynamic and recomputed at each flow remapping.

3.2.2. The flow management system

The flow management system is responsible for maintaining a dynamic list of *active* flows where an active flow is defined as one generating downstream traffic at a rate exceeding a minimum threshold.

At each remapping time t, every flow's bandwidth consumption is computed as an exponentially decaying moving average:

$$r_i^t = 0.4 \times bits/\tau + 0.6 \times r_i^{t-\tau}$$

where τ is the length of the remapping interval and *bits* is the number of bits transmitted on behalf of flow i in $[t-\tau, t]$.

State variables associated with each flow are its weight w_i, current bandwidth consumption estimate r_i and the identity j of the bin to which it is presently bound.

3.2.3. The scheduling system

The mission of the outer scheduler is to select the bin to be served when a downstream transmission opportunity occurs. Transmission opportunities are created when a packet is enqueued and the medium is idle or when a transmission completes. The scheduler is implemented as weighted DRR with bin j having weight ω_j. In our implementation, the deficit counter of a backlogged bin, j, is incremented by the size of one maximum MTU Ethernet packet multiplied by the bin's current weight, ω_j, at the start of each round.

The inner scheduler is FCFS augmented by drop tail CoDel.

3.2.4. The reclassification system

At the end of every reclassification time interval τ, the operational parameters of the ABB system are recomputed. For every active flow, f_i, the flow's estimated consumption rate is updated as previously described. The value of C_j, the maximum weighted per-flow consumption allowed in bin j is recomputed for all bins. Each flow, i, is then assigned to the bin j whose C_j is the first C_j that exceeds r_i/w_i. Then the bin weights $\omega_j = \sum_{f_m} w_m$ for flows f_m bound to bin j are recomputed for all bins. Though the reclassification operation runs with a complexity of $O(n)$, where n is the number of flows, it is appropriate to run it *only* once per several thousand or even more packets scheduled.

3.2.5. Computing the bin bandwidth thresholds

We now describe the procedure for computing the bin bandwidth consumption thresholds, C_j. For the results reported in this study, we employed 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.

A simple, though unrealistic, example illustrates the approach. Suppose the shared medium downstream has a capacity of 40 Mbps and six flows of equal weight have demands of 4, 6, 7, 8, 9, and 10 Mbps, respectively. In computing the max-min fair share for the flows, the first max-min fair breakpoint is the total capacity divided by the total number of flows which is $40/6 = 6.7$. With this breakpoint, the demands of the first two flows can be completely satisfied with a remaining capacity of 30 Mbps to be shared among the four remaining flows. Consequently, the next breakpoint is thus $30/4 = 7.5$ and the demand of the third flow can be satisfied leaving 23 Mbps unallocated. The third max-min fair breakpoint is then $23/3 = 7.7$, Since the demands of the remaining flows exceed 7.7 they each receive 7.7 Mbps. Assuming there are four bins then the consumption thresholds are $C_1 = 6.7$, $C_2 = 7.5$, $C_3 = 7.7$, and $C_4 = C = 40$. Flows 1 and 2 are assigned to bin 1, flow 3 to bin 2, no flows to bin 3, and flows 4, 5, and 6 to bin 4. If the system had been configured with only three bins a perfect mapping could also have been achieved since no flows were associated with the 7.7 Mbps break point.

Nevertheless, since the number of flows is variable and potentially large, the number of max-min fair break points capturing at least one flow can certainly exceed the fixed number of bins. In fact it can be shown that for any number of flows n, it is possible to construct an increasing sequence of demands $r_1 < r_2 < \ldots < r_n$ such that each r_i is a max-min fair break point and consequently each flow is technically "entitled" to its own bin. Obviously in this case, $(r_{i+1} - r_i) \to 0$ as $i \to n$.

Therefore, when the number of breakpoints exceeds the number of bins the following steps can resolve the issue:

- Eliminate any breakpoints with no associated flows (e.g. the 7.7 Mbps breakpoint in the preceding example).
- Bins in which all flows are receiving their current demands can be merged (e.g. bins 1 and 2 in the preceding example).
- Identify the breakpoint C_j for which $C_{j+1} - C_{j-1}$ is minimal and eliminate C_j or
- Identify the breakpoint C_j for which the number of flows having demand between C_j and C_{j-1} is minimal and eliminate C_{j-1}.

3.2.6. Avoiding out of order packet transmissions

While a flow is bound to a particular bin, its packets will be sent in FCFS order under ABB. When a flow is moved to another bin, packets may remain in the queue of the previous bin and during a small window of time on the order of the CoDel target delay, it would be possible for packets to be sent out of (global) FCFS order.

If the CoDel target delay is 20 ms and the reclassification interval is set to 1 s, the degree of potential packet reordering is estimated to be no more than $0.02/1 = 2\%$. Although the potential reordering does not appear to be a significant issue it could have some negative effect over the TCP performance.

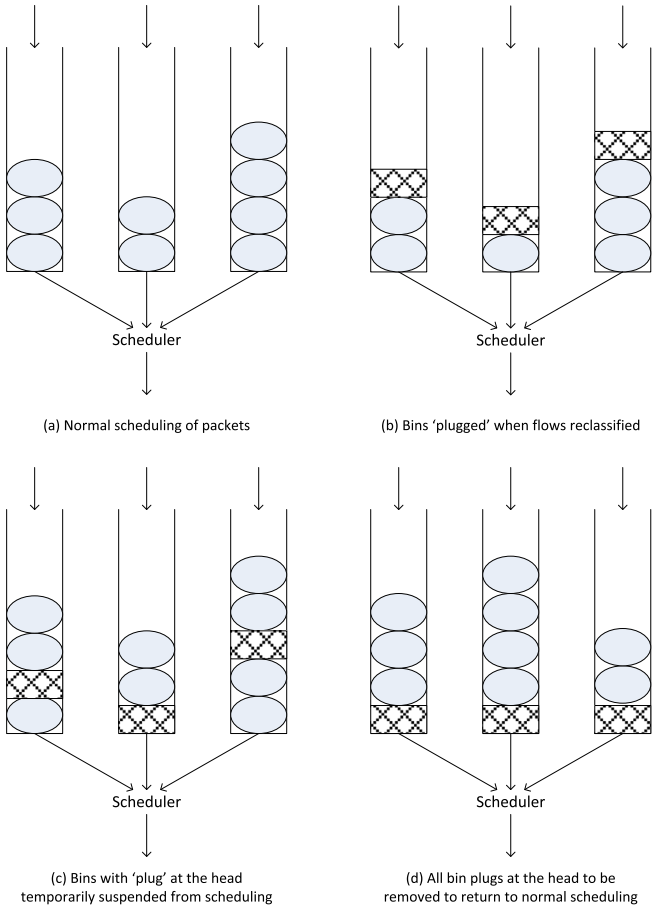


Fig. 3. Addressing packet reordering issue by bin plugging.

The same packet reordering problem exists with other schedulers such as SFQ when periodical perturbation of the hash function is performed. But the problem was not raised and addressed in [21]. 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 chronological order. This can be quite costly. The FQ-CoDel code in the Linux kernel, on the other hand, does not do periodical 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 [27] and potential collisions are simply accepted.

We address the out-of-order issue with a low complexity solution illustrated in Fig. 3 using three bins as an example. First, packets are scheduled normally from the bins (Fig. 3(a)). At each flow reclassification, if there is any flow moving to a new bin, all bins losing or gaining flows are "plugged" (Fig. 3(b)). The 'plugs' serve the purpose of dividing the unsent 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 queue. From this time on, any queue that has its plug at its head is temporarily suspended from scheduling (Fig. 3(c)) preventing the new packets from being sent before the old packets queued before the remapping. The temporary scheduling suspension of a bin continues until the plugs of all bins reach their bin heads (Fig. 3(d)). At this time, all plugs are removed and the bins return to the normal scheduling as before (Fig. 3(a)).

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. We will show that the disruption is negligible.

4. Simulation setup

Our studies are carried out in a simulated DOCSIS 3.x cable environment on the ns-2 simulator [28]. The support for DOCSIS in ns-2 was developed in prior work [31].

4.1. Simulated network

Fig. 4 illustrates the simulated network used in the studies. We model a single DOCSIS MAC domain in which one CMTS interacts with a number of CMs. In the DOCSIS 3.0 case, the model assumes channels that offer downstream and upstream physical layer data rates of 42 and 30 Mbps respectively. For DOCSIS 3.1, 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 simulations 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.

The simulated network also includes a regulator to support the conventional service tiering model based on a maximum sustained service rate. The regulator can be either enabled or disabled, but was disabled for the results reported here.

Our simulations involve workloads consisting of a number of traffic types including FTP, HTTP Adaptive Streaming (HAS), web, exponential on/off, and CBR traffic. Other than HAS, the traffic models we use are those provided in ns-2. Details of the operation of the HAS model are found in [32] and [30].

4.2. Simulation definition

We compare the downstream performance of ABB to several alternative bandwidth management systems. We use weighed DRR with per-flow queuing and CoDel as the reference for performance evaluation because weighted DRR is max-min throughput fair, and the addition of CoDel provides the desired control of latency.

We also evaluate the performance of SFQ-CoDel and single queue FCFS with CoDel. Our implementation of SFQ-CoDel is adapted from [29] to work in our DOCSIS simulation framework. The FCFS single-queue CoDel is adapted from an open source tail-drop variant of CoDel for DOCSIS.

Our analysis covers several scenarios using varying numbers of downstream application flows, traffic loads, and traffic types. 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 considered as simulation parameters. In each set of simulations, 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. For delay-based AQMs, the maximum buffer size setting is less significant as long as the buffer size is large enough to accommodate occasional bursts. For the results reported here, the maximum buffer size at the CMTS for delay-based AQMs supports a maximum 150 ms queuing delay.

In the simulations presented, all network layer PDUs are 1500 bytes. The bandwidth delay product of a round trip path consisting

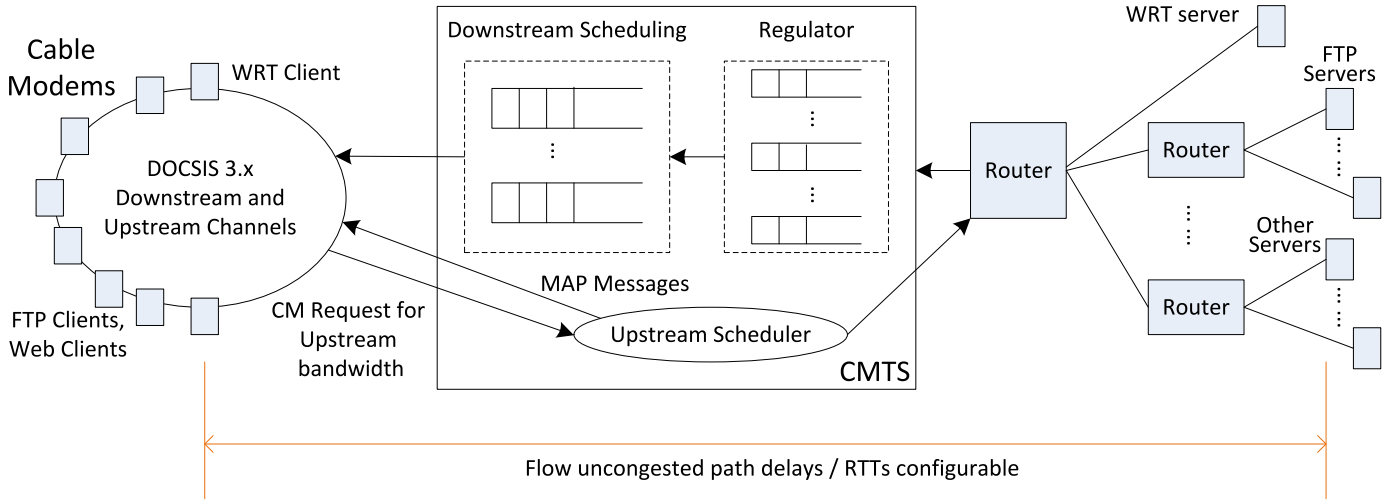


Fig. 4. Simulation network model.

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 for a data rate of 1 Gbps and 350 for a data rate of 42 Mbps. Maximum buffer capacities at the simulated routers 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 CMTS is set to 11,875 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 TCP configuration for the ns-2 simulator include following settings:

- TCP: TCP/Sack (ns-2 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 the workload involving HAS, the ns-2 Agent/TCP/FullTCP/Sack was used for HAS.

For CoDel, we configure the target delay to be 0.020 s. The original CoDel recommends a target delay of 0.005 s. We determined that a setting of 0.020 s 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 s. Following the recommendation, SFQ-CoDel is configured to use 1024 bins as in the original code.

For ABB, we ran the simulations with a small number of bins (2, 3, and 4 in most cases). The reclassification interval is set to 1 s. Each flow's bandwidth consumption is computed as an auto regressive moving average as previously described.

We decompose our analysis into two sets of simulations as shown in Table 3. The first set is limited to a single bandwidth tier with a D3.0 PHY layer channel speed of 42 Mbps. Objectives include evaluating the ability of ABB to mitigate TCP RTT unfairness, provide flow isolation, ensure low latency, and adapt to changing workload.

In the second set, the analysis is conducted in a multi-tier environment with a D3.1 PHY layer channel speed of 1 Gbps. We focus on evaluating the ability of ABB to provide weighted fairness based on flow tiers, and to provide flow isolation with respect to unresponsive UDP flows. We also explore the ability of ABB to operate with realistic traffic workloads involving FTP, HAS, and web flows.

4.3. 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 [33]. Defining fairness is challenging as the term has different meanings in different contexts. For packet scheduling based on approximations of fair queuing (e.g., WFQ [15], SCFQ [16]), fairness is assessed on packet transmission time scales.

In a scheduling environment that is based on single queue AQM such as AFD [34] and AFCD [26] that attempt to isolate "elephant" flows from "mice" flows, the definition of fairness is significantly relaxed. A measure such as Jain's Fairness Index (JFI) [35] computed based on hundreds of seconds are common.

JFI is a widely used assessment on how achieved resource allocation differs from the desired outcome. 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 desired weighted fair share outcome. JFI is defined as:

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

An alternative fairness index is called the min-max ratio. It is defined as:

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

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

We assume that weighted *max-min fair* throughput allocation criterion is the desired fairness objective. It has been shown that packet scheduling algorithms that implement close approximations of weighted fair queuing, such as weighted DRR packet scheduling [36], can achieve weighted max-min fairness.

We assess throughput fairness over time scales of multiple seconds or minutes using 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.

Table 3
Simulation definition.

Set	Simulation	Summary
Single Tier	**BASE**	For different simulation runs, varies the number of competing downstream FTP flows. 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** but 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.

Results from our simulation studies are presented in Sections 6 and 7. Each section corresponds to each set of simulations in Table 3. Common performance metrics we use include measurements of throughput and packet delay. A number of application specific performance metrics we use are given in [30,32].

5. Parameter selection and analysis

It is necessary to define two parameters when configuring an ABB system: the number of bins k and the reclassification interval τ. In this section we describe the way in which our binning algorithm and workload characteristics determine reasonable parameter choices.

5.1. Number of bins

The number of useful bins in an ABB system is inherently limited by the nature of the max-min fair breakpoint guided binning algorithm and the characteristics typical of Internet flows.

5.1.1. Binning algorithm constraints

As previously described the standard max-min fair algorithm for bandwidth allocation is driven by: the number of flows n, the capacity of the shared link C, and the demand of each flow $d_i\,(1 \le i \le n)$. In the Internet, the transmission rate of most flows is reduced by the source of the flow in response to congestion indicators. Therefore, there is no reliable way for an ABB router to identify the true demand d_i of any given flow. Consequently, in the ABB system, demand d_i is estimated as a moving average of recent bandwidth consumption.

Given n flows $\{f_i : i = 1, \ldots, n\}$, let d_i now represent the estimated demand of flow i, and assume that the flows are ordered such that $d_i < d_j$ for $i < j$. Let C_j represent the upper bound on demand, d_i, for flows belonging to bin $\{b_j : j = 1, \ldots, k\}$. Let $W_j = C_j - C_{j-1}$ represent the bandwidth of bin b_j. Let $\{n_j : j = 1, \ldots, k\}$ represent the number of flows in bin b_j. Then $n = \sum_{j=1}^{k} n_j$.

From the max-min fair algorithm, $C_1 = C/n,\ C_k \le C$, and by convention $C_0 = 0$. Depending on the value of the demands $d_i\,(1 \le i \le n)$, the number of occupied max-min fair bins can range from 1 to n. If $d_i < C/n\ \forall i$, then each flow, f_i, maps to bin b_1 and its allocation a_i is d_i. If $d_i > C/n\ \forall i$, then each flow, f_i, maps to bin b_2 and its allocation a_i is C_1. This is a scenario where there is only one useful bin occupied by all flows.

The following procedure can be used to construct a sequence of demands d_i that produces n distinct bins with each bin occupied by only a single flow. Since the mapping of flow to bin is one to one in this case, let the subscript i below refer to both a flow and its corresponding bin. Let R_i be the residual capacity after flow, f_i, is assigned to bin b_i. The underlying idea is for d_i to lie halfway between C_{i-1} and C_i. Initialize by setting $R_0 = C$, and $d_0 = C_0 = 0$. Then repeat the following steps for $i = 1$ to n:

1. $C_i = R_{i-1}/(n - i + 1)$
2. $d_i = a_i = (C_i + C_{i-1})/2$
3. $R_i = R_{i-1} - a_i$

Table 4
One flow per bin with $d_i = (C_{i-1} + C_i)/2$.

i	d_i	C_i	W_i	R_i
1	3.200	6.400	6.400e+00	124.800
2	6.484	6.568	1.684e-01	118.316
3	6.571	6.573	4.678e-03	111.745
4	6.573	6.573	1.376e-04	105.172
5	6.573	6.573	4.300e-06	98.599
:	:	:	:	:
20	6.573	6.573	0.000e+00	0.000

Table 5
One flow per bin with $d_i = C_{i-1} + \epsilon$.

i	d_i	C_i	W_i	R_i
1	0.000	6.400	6.400e+00	128.000
2	6.400	6.737	3.368e-01	121.600
3	6.737	6.756	1.871e-02	114.863
4	6.756	6.757	1.101e-03	108.108
5	6.757	6.757	6.880e-05	101.351
:	:	:	:	:
20	6.757	6.757	0.000e+00	0.000

We illustrate the procedure with $C = 128$ Mbps and $n = 20$ flows. The results are shown in Table 4. The table shows that the bandwidth, W_i, of the individual bins rapidly converges to 0 as C_i converges to 6.573. Consequently, the bins useful for distinguishing flow demands are limited to just a few: [0, 6.4], (6.4, 6.568], and possibly (6.568, 6.573].

It is natural to ask if there are different demand sets that will provide more widely dispersed bin boundaries. Bin size separation can be maximized for a single flow per bin system by letting $d_i = C_{i-1} + \epsilon$. In the limit as $\epsilon \to 0$, the bin boundaries shown in Table 5 are realized. Here the potentially useful bins, again limited to just a few, are [0, 6.4], (6.4, 6.737], and (6.737, 6.757].

A more useful and realistic modification to the demand profile also produces better bin separation than single flow per bin. In this case we create demands that map m flows to each bin. This obviously reduces the number of distinct bins by a factor of m. Small changes to the procedure previously presented generate a set of m demands such that the mean value of the m demands is $(C_i + C_{i-1})/2$. These results are shown in Table 6 with $m = 4$. The bandwidth decays more slowly but there is still a little separation beyond the first 2 bins. Each line in this table represents the characteristics of four flows (f_{i-3}, f_i) that occupy bin b_j.

It is possible to reduce further the decrease in bin bandwidth by creating demands d_i such that the bin populations n_j decrease exponentially. When this is done, the number of possible bins is $\log_2(n) + 1$. For the results shown in Table 7 the values of n_j are 16, 8, 4, 2, 1, 1 and the mean demand of the flows in bin j is

Table 6
Four flows per bin with $\bar{d}_i = (C_{i-1} + C_i)/2$.

i	d_i	C_i	W_i	R_i
4	3.200	6.400	6.400e+00	115.200
8	6.800	7.200	8.000e-01	88.000
12	7.267	7.333	1.333e-01	58.933
16	7.350	7.367	3.333e-02	29.533
20	7.375	7.383	1.667e-02	0.033

Table 7
Exponentially decaying flows per bin with $\bar{d}_i = (C_{i-1} + C_i)/2$.

i	d_i	C_i	W_i	R_i
16	2.000	4.000	4.000e+00	96.000
24	5.000	6.000	2.000e+00	56.000
28	6.500	7.000	1.000e+00	30.000
30	7.250	7.500	5.000e-01	15.500
31	7.625	7.750	2.500e-01	7.875
32	7.812	7.875	1.250e-01	0.062

Table 8
Exponentially decaying flows per bin with $\bar{d}_i = C_{i-1} + \epsilon/n_j$.

i	d_i	C_i	W_i	R_i
16	0.000	4.000	4.000e+00	128.000
24	4.000	8.000	4.000e+00	96.000
28	8.000	12.000	4.000e+00	64.000
30	12.000	16.000	4.000e+00	40.000
31	16.000	20.000	4.000e+00	24.000
32	20.000	24.000	4.000e+00	4.000

$(C_j + C_{j-1})/2$ as before. The bandwidth in this case is still decaying exponentially fast with $W_j = 4/2^{j-1}$.

Finally, in the limit it is possible to craft a set of populations, n_j, and demands d_j such that the bin bandwidth does not decay at all. In this case we let the n_j decay exponentially as above and we let $d_i = C_{j-1} + \epsilon/n_j$ for all flows in bin j. Then in the limit, as $\epsilon \to 0$ the bin boundaries shown in Table 8 are obtained. The residual $R_{32} = 4$ occurs because the max-min fair share for flow $f_{32} = 24$, but its demand $d_{32} = 20$.

We conclude that the maximal useful number of max-min fair bins is $\log_2(n) + 1$ where n is the number of flows, but for realistic demand sets it likely to be 4 or 5 bins at the very most.

5.1.2. Impact of flow characteristics

The characteristics of realistic Internet traffic flows are also consistent with the use of a relatively small number of bins. The demand of a flow is said to be *elastic* if the demand is effectively unbounded. For example, an FTP file transfer has elastic demand. In contrast, a flow consisting of uncompressed CD quality stereo audio is *inelastic*. Its demand is fixed at 88 Kbps.

A flow is said to be *responsive* if it reduces its transmitting rate in response to network congestion. A flow is said to be *unresponsive* if it transmits at a static rate $d_i = \hat{d}_i$ regardless of congestion indicators.

The objective of the response mechanism is to limit the rate at which packets of the flow are being transmitted by the source host to the approximate rate at which the packets of the flow are presently being forwarded by the bottleneck router. The most widely used mechanism for implementing the response mechanism is the additive increase with multiplicative decrease (AIMD) system used in standard implementations of the TCP protocol. A well-known disadvantage of the AIMD mechanism is that it introduces an RTT unfairness among competing responsive flows in which the throughput of a flow is inversely proportional to its RTT. Gavaletz and Kaur [37] show that RTT unfairness pertains to every end-to-end congestion management system that operates at an *RTT-long* time scale.

As previously noted the ABB algorithm uses a moving average of recent throughput to estimate d_i. We will use $\hat{d}_i$ to refer to the true demand of an inelastic flow. Since the ABB algorithm is applied to measured throughput, d_i, it will always be the case that $\sum_{i=1}^{n} d_i \le C$. Consequently, $C_k \le C$ and in applying the max-min fair breakpoint binning algorithm, it will always be the case that when f_i is mapped to bin j, then $a_i = d_i \le C_j \ \forall i$.

It might seem surprising that an algorithm that always finds the existing bandwidth allocation to be max-min fair with respect to d_i produces a bin mapping that results in a bandwidth allocation that approximates max-min fairness with respect to the underlying true demands, $\hat{d}_i$. This occurs because flows that have similar values of d_i inherently compete fairly within the queue to which they have been assigned, and the weighted DRR scheduling of the competing bins generally enforces inter-bin fairness.

In the long term scheduling of steady-state flows, the objective of the ABB system is to obtain approximate max-min fair scheduling. It does so by distributing the flows as follows:

1. Bin 1 contains flows with inelastic demands for which $d_i \le C/n$. These flows may be responsive or unresponsive and have no negative impact on other flows. In the stable state this bin will experience no queuing loss.
2. Bin k, the last bin, contains the unresponsive flows having demand d_i that exceeds the demand of all responsive flows. High demand unresponsive flows can be identified via their queuing loss rate or other means. These flows do not compete fairly in a queue with any other type of flows or each other and must be quarantined.
3. Bin 2 through bin $k-1$ are used to group flows for which $d_i > C/n$ but less than that of bin k flows into bins whose boundaries are computed by the max-min fair breakpoint binning algorithm. The flows that get mapped to bin b_j have demands d_i such that $C_{j-1} < d_i < C_j$. Consequently the flows, which can include responsive and unresponsive flows with both elastic and inelastic demands, do share similar observed throughput. The purpose of these bins is to compensate for RTT unfairness and unfairness due to differences in the underlying demand, $\hat{d}_i$, which can be reflected in achieved throughput, d_i. If significant RTT disparities exceed the number of available bins, a flow will periodically migrate between adjacent bins to mitigate the unfairness.

To corroborate the above analysis, we ran additional simulations. The results were consistent with the analysis. In the following, all simulations with their results reported in the next two sections are thus configured to use no more than 5 bins.

5.2. Reclassification interval

The reclassification interval, τ, in this paper is one second. It was experimentally selected by noting when additional runs of our test workloads with smaller intervals did not significantly alter the results. The binning boundaries for the stochastically static workloads used in this section converge to their final values in about 10 s.

In general proper value of τ is clearly strongly dependent on the mean duration of the flows comprising the workload along with the variability of $\hat{d}_i$ during the life of each flow. For any fixed τ, it would be easy to configure a workload that would perform poorly. This indicates that ideally τ should be dynamic, but that is an issue we leave for future research.

Table 9
Simulation BASE TCP throughput (Mbps): mean / CoV / sum.

#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

6. Results and analysis – single tier

In this section we present the results of the single tier set of simulations shown in Table 3. The objective of this set of simulations is to evaluate the effectiveness of the ABB scheme in providing fairness and controlling latency.

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. Unless otherwise specified, the reclassification interval τ is set to 1 s. This value was chosen based upon the analysis of our preliminary simulations but is definitely not thought to be optimal across differing workloads. For this set of simulations, the simulation time is set to 2000 s unless otherwise specified.

6.1. *BASE* simulation results

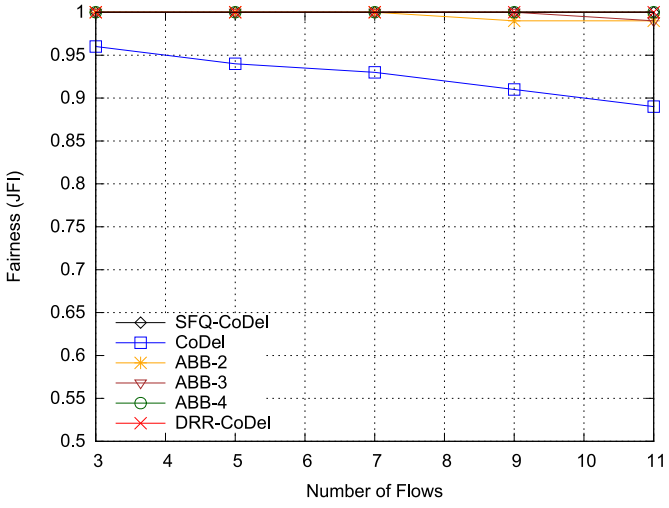
Simulation **BASE** employs a varying number of FTP flows ranging from three to eleven. The flows compete for a total available network layer bandwidth of 38 Mbps. Each simulated FTP flow starts at a random time within the range of 0.0 to 2.0 s and has a unique uncongested path RTT. The mean uncongested RTT of the FTP flows is 80 ms in all cases, but individual FTP flows have different uncongested path RTTs using the pattern $\{80, 70, 90, 60, 100, \ldots\}$ ms.

Fig. 5 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 shown to get worse as the number of flows increases. As shown in our prior work [32], this is primarily a function of the disparity in the uncongested path RTTs. It is known as TCP RTT unfairness.

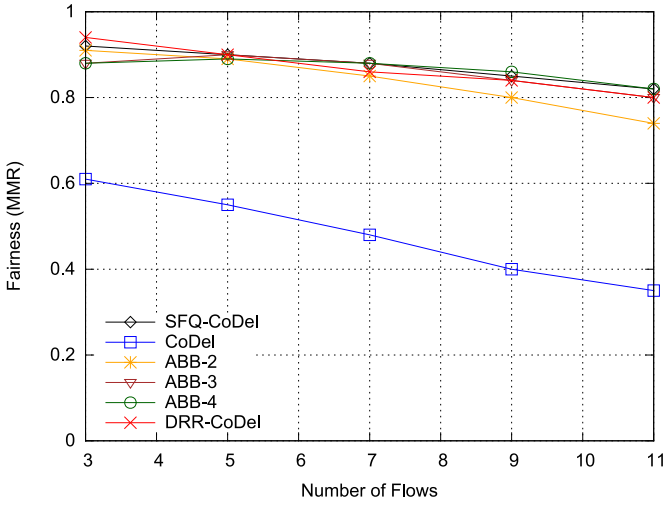
DRR-CoDel provides fairness closest to max-min fairness. In results not shown, DRR without CoDel is max-min fair, but has much higher variation in latency than obtained with AQM. SFQ-CoDel has almost identical fairness to that of DRR-CoDel. This is not surprising because the number of bins in use by SFQ-CoDel far exceeds the number of flows. Consequently, the chance of collision is very small. 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. With the use of more bins, ABB scheme in general is able to better address the RTT unfairness issue.

Fig. 6 shows moving throughput fairness computed in windows of 30 s of simulated time for the run of nine FTP flows. Results for the 3 ABB schemes along with single queue CoDel are shown. For all ABB schemes, the fairness over time is both stable and close to that of long term fairness shown in Fig. 5. For other runs of different number of FTP flows, the results are similar. These results indicate 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 9 summarizes the throughput results by the FTP flows under different schemes. The three values shown in each block are the mean throughput per flow, the coefficient of variation of throughput across the flows, and the aggregate throughput. DRR without CoDel achieved 37.4 Mbps of aggregate throughput in all



(a) Jain's fairness index



(b) Min-max ratio

Fig. 5. Simulation BASE throughput fairness results.

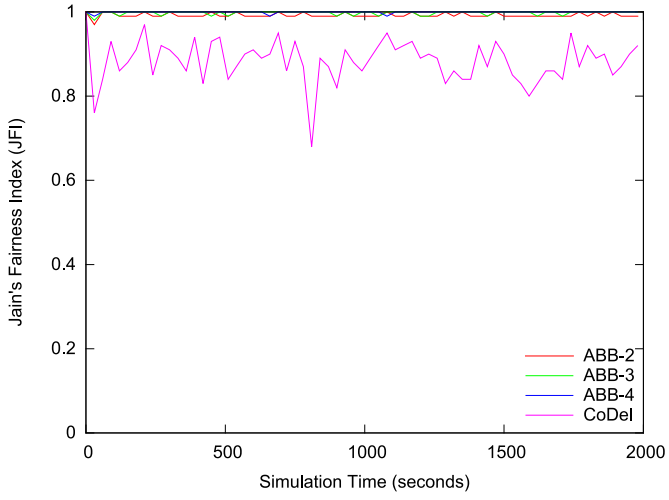
cases. This indicates that the slightly lower aggregate throughput achieved for smaller numbers of flows is caused by AQM. When compared to the DRR-CoDel, the total throughput results from the ABB scheme with 3 and 4 bins are shown to be the same or occasionally higher. This suggests that the ABB scheme provides good utilization of the channel capacity. The coefficient of variation (CoV) values are consistent with Fig. 5.

6.1.1. Latency control

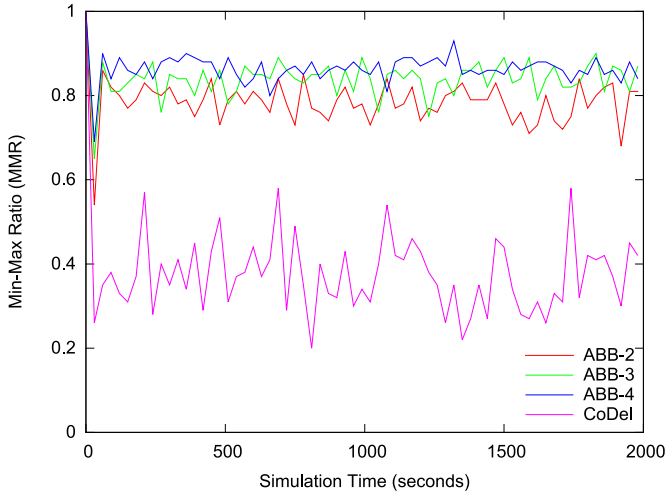
Table 10 shows the average RTT experienced by the TCP/FTP packets. The average RTT includes the inherent path delay plus the

Table 10
Simulation 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



(a) Jain's fairness Index



(b) Min-max Ratio

Fig. 6. Simulation BASE throughput fairness results over time.

queuing 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. ABB-3 and ABB-4 are shown to provide good delay performance comparable to that of DRR-CoDel and SFQ-CoDel.

6.1.2. Dynamic behavior of ABB

Table 11 illustrates the dynamic behavior of the ABB scheme. The results provide per-flow bin switches and the rate of such switching. The switching rate is the number of switches per second per flow. Therefore 1213 switches over the 2000 s of simulation for 3 FTP flows yields a rate of $1213/3/2000 = 0.202$. The table also provides the disruption time and ratio due to queues being plugged to prevent packet reordering. The ratio is calculated as the total 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.

6.2. Mice / elephant (*ME*) simulation results

The set of ME simulations 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 ABB. We are also interested in the dynamic behavior of the scheme with this type of traffic load.

The first variant, called ME1, runs 36 concurrent on/off TCP "mice" flows. Time in the on state is uniformly distributed in [2, 4] and has a mean of 3 s. Off time is three times the preceding on time. Transmit rate in the on state is 2.0 Mbps. Consequently, each flow generates an average rate of 500 Kbps, and the aggregate load averages 18 Mbps.

For all "mice" flows the time between packet transmissions is exponentially distributed. The mean is set to produce the target transmission rate in the *on* state.

Four always on "elephant" TCP/FTP flows sourced from four separate FTP servers compete with the on/off flows. Given a 38 Mbps channel capacity, the max-min fair allocation for the FTP flows is 5 Mbps each. 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 12 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. Single

Table 11
Simulation BASE – ABB dynamic behavior.

#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)	10,268 (0.47)
ABB-4	1514 (0.25)	4381 (0.44)	7280 (0.52)	10,055 (0.56)	12,045 (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)

Table 12
Simulation ME1 flow throughput (Mbps) (mean / CoV / sum).

Scheme	FTP	On-off	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

Table 13
Simulation ME1 flow average RTTs (second) (mean / stddev).

Scheme	FTP	On-off	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 14
ABB dynamics with simulation 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)

Table 15
Simulation ME2 flow throughput (Mbps) (mean / CoV / sum).

Scheme	FTP	On-off	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 16
Simulation ME2 flow average RTTs (second) (mean / stddev).

Scheme	FTP	On-off	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

Table 17
ABB dynamics with simulation 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)

queue CoDel is much less fair and has 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 reasonable channel utilization.

The RTT information for ME1 is given in Table 13. 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, as expected, single queue CoDel does not provide a lower average RTT for the mice flows because it employs a single queue. This indicates that the ABB scheme, like DRR-CoDel and SFQ-CoDel, is able to provide better flow isolation between 'elephants' and 'mice' than does single queue CoDel.

The dynamic behavior of ABB in terms of flow bin switches and disruption time in ME1 is shown in Table 14. 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.

The second variant, called ME2, is similar to ME1 except it runs 180 on/off TCP "mice" flows. On time is constant at two seconds with off time also constant at three seconds. The use of exponentially distributed inter packet times in the on state reduces the undesirable synchronization effects associated with fixed holding times in each state. The transmission rate in the on state is 250 Kbps. The sustained transmission rate for each flow is therefore 100 Kbps.

The results for ME2 are similar to that of ME1. The detailed data are given in Tables 15–17.

The third variant, called ME3, runs 90 low rate on-off flows. These flows are identical to the ME2 "mice" flows. There are 90 additional mid-rate on-off flows whose transmission rates in the on state are uniformly drawn from (0.75Mbps, 1.5Mbps) at the start of each flow. Holding times in the on and off states remain constant at two and three seconds for all "mice" flows. The sustained rate of the low rate on-off flows remains at 100 Kbps. For the mid-rate on-off flows it varies between 300 Kbps and 600 Kbps.

The low rate on-off flows have an aggregate demand of 9 Mbps, and the mid-rate on-off flows have an aggregate demand of 40.5 Mbps. The 4 FTPs are unchanged. Consequently, ME3 is a highly congested scenario. Max-min fair allocation is 100 Kbps for the low rate flows and 310 Kbps for the mid-rate and FTP flows.

The throughput and RTT results are given in Tables 18 and 19 for ME3. For the throughput results, we can see that the ABB schemes provide throughput fairness that is comparable to that of DRR-CoDel and SFQ-CoDel. On the other hand, single queue CoDel is not shown to be fair as it does not provide good isolation between different classes of flows. It allows mid-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

Table 18
Simulation ME3 flow throughput (Mbps) (mean / CoV / sum).

Scheme	FTP	Low On-off	Mid On-off	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 19
Simulation ME3 flow average RTTs (second) (mean / stddev).

Scheme	FTP	Low On-off	Mid On-off	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

Table 20
ABB dynamics with simulation ME3.

Scheme	Number of Bin Switches (rate)	Disruption Time (ratio)
ABB-2	73,880 (0.20)	22.5 (0.011)
ABB-3	106,564 (0.29)	38.9 (0.019)
ABB-4	117,082 (0.32)	40.3 (0.020)

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 under ABB.

It is to be expected that in the highly congested scenario the ABB schemes will exhibit a high rate of bin switches. The results given in Table 20 confirm this. However, the overall disruption these switches caused remains small.

7. Results and analysis – multiple tiers

We now report the results from the multi-tier set of simulations that were identified in Table 3. For this set of simulations, we set the channel capacity to be 1 Gbps, which is on par with the D3.1 channel capacity. Usable network layer bandwidth is approximately 948 Mbps. The multi-tier set of simulations is designed to evaluate the effectiveness of ABB in providing *weighted* max-min fairness. Unless otherwise specified, the reclassification interval of ABB is set to 1 s and all simulation times are set to 1000 s.

The tiering support implemented in ABB is designed to support weighted max-min fair sharing of the full downstream bandwidth of the system. As such it is comparable to weighted DRR-CoDel.

Nevertheless, it is not comparable to the tiering support typically in use in current DOCSIS systems. In these systems, each tier has an associated maximum transmission rate. Arriving packets must pass through a token bucket based regulator before entering the single queue CoDel transmission queue. The regulator thus limits the long term downstream transmission rate to the maximum rate of the tier regardless of the total downstream load and is not generally capable of delivering weighted max-min fairness. Our ABB implementation can be configured to use a token bucket regulator, but for these simulations, it was not.

7.1. Implementing tiering in ABB

As previously stated the objective is to provide weighted max-min fair allocation. It is important to note that weighted max-min fair allocation applies to *flows* as contrasted to *tiers*. Per tier max-min fair allocation biases the allocation against tiers with the largest populations of flows.

For example, suppose there are three tiers with relative throughput weights of 1, 2, and 4 respectively and the downstream link supports a bit rate of 77 Mbps. Suppose there are four flows having elastic demands with one flow in tier one, one flow in tier two, and the remaining two flows are in tier three. Then the weighted max-min fair allocation to the four flows is 7, 14, 28, and 28 Mbps respectively. For weighted max-min fair allocation among tiers, the per tier allocation is 11, 22, and 44 Mpbs and the per flow allocation is consequently 11, 22, 22, and 22.

As described in Section 3, a flow weight, w_i is associated with each flow, f_i. In the single tier experiments it was assumed that $w_i = 1\ \forall i$, and therefore the relative bin weight, ω_i, was simply the number of flows in bin b_i.

An obvious way to implement ABB with multiple service tiers is to provide a separate, dedicated set of ABB managed bins for each service tier and confine the flows of each tiers to the associated bins. Let $b_{i,j}$ represent bin i in tier j. Let χ_j represent the relative weight of tier j and $n_{i,j}$ the number of active flows in bin $b_{i,j}$. With this configuration, the relative bin weight, $\omega_{i,j}$, used by the outer DRR scheduler of ABB is $\chi_j n_{i,j}$. With complete tier isolation ensured, one would expect and will obtain fairness that is equivalent to that obtained in the single tier systems evaluated in the previous section. The bin set per tier design introduces some additional design and implementation complexity and a small amount of additional operational overhead.

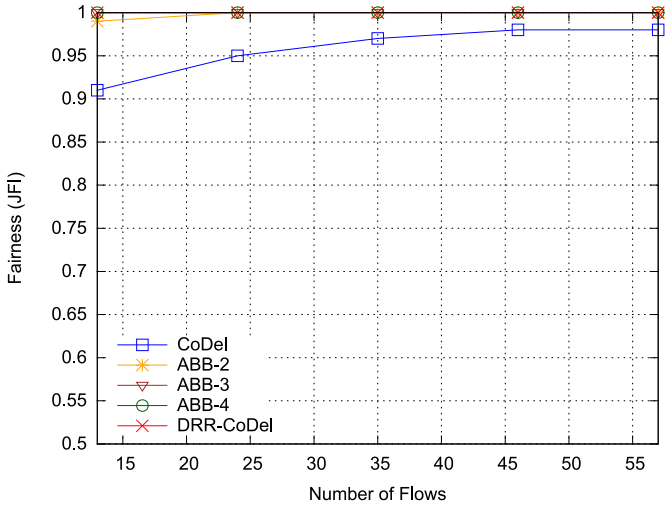
Alternatively, it is also easy to extend ABB to support multiple service tiers with no change to the single tier design and no additional operational overhead. In this approach, when a new flow f_i is recognized, its weight w_i is set to χ_j, the relative weight of the tier to which it belongs. No other change to the binning system or scheduling system is required.

This simple approach clearly has the potential to adversely affect fairness. All flows belonging to a single bin continue to obtain approximately equal throughput. So if a single tier 1 flow happens to reside in a bin with five tier 3 flows, the weight of the bin will provide tier 3-like performance to the unfair benefit of the tier 1 flow. Conversely, a tier 3 flow sharing a bin with multiple tier 1 flows will have its performance adversely affected. The flow reclassification system will detect and attempt to mitigate instances of bandwidth allocation that are not weighted max-min fair, but there is no way in general to predict how effective the mitigation will be.

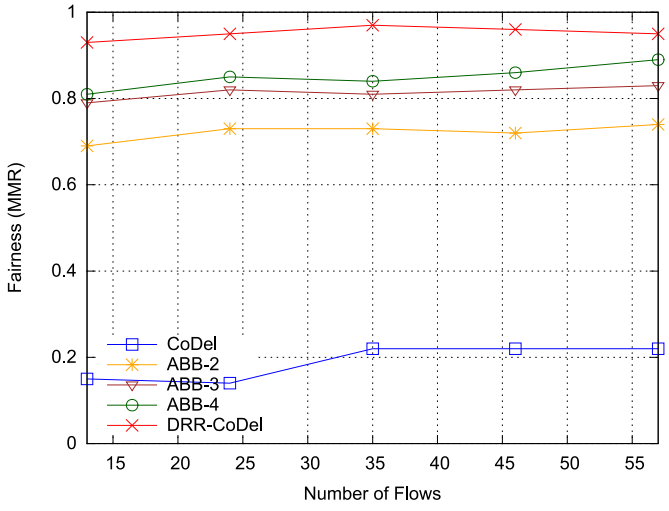
In the remainder of this section we present results obtained using tier weights on a single tier ABB system. It will be observed that, as expected, fairness is not as effectively maintained as in the single tier studies, and that fairness is dependent on the number and nature of the flows in each tier.

7.2. ***TIER*** simulation results

The **TIER** simulation is similar to the BASE simulation in purpose, but it extends the evaluation to include tiered service quality levels. Because SFQ-CoDel is not designed to support weighted or

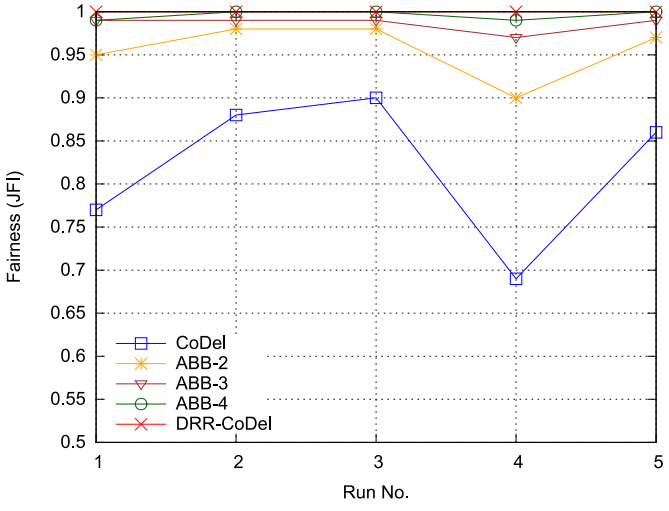


(a) Jain's fairness index

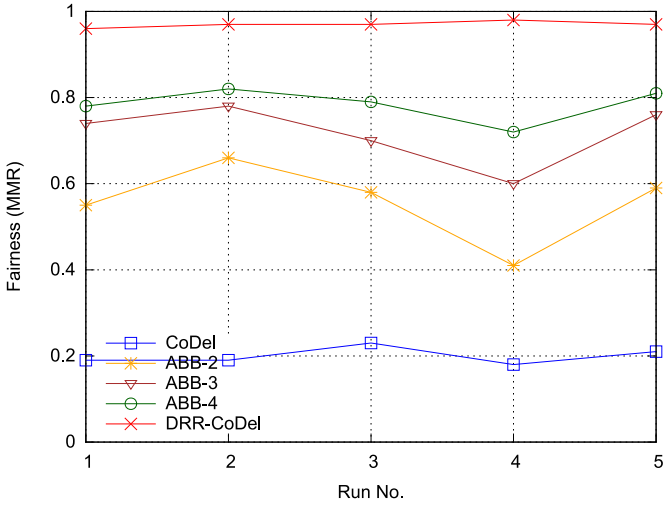


(b) Min-max ratio

Fig. 7. Simulation TierG weighted throughput fairness results.



(a) Jain's fairness index



(b) Min-max ratio

Fig. 8. Simulation TierM weighted throughput fairness results.

tiered flows, it was not evaluated in these studies. All runs include TCP/FTP flows in three tiers (Tier1, Tier2, and Tier4 with a weight of 1, 2, and 4 respectively). We ran two variants of the simulation.

The first variant is called TierG. It runs a varying number (11 to 55) of FTP flows in 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. Because of the far higher link speed, the simulation time is reduced to 1000 s.

Fig. 7 shows the weighted throughput fairness results from five simulation runs with different numbers 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 not to be expected that single queue CoDel (shown for comparison) provide weighted fairness.

The average tiered flow throughput is given in Table 21. 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 22. The ABB scheme is shown to maintain similar queuing delay for all flow tiers. Again the ABB scheme is able to provide similar performance to that of DRR-CoDel.

The second variant of the TIER simulation is called TierM. For different runs, the total number of FTP flows is fixed at 35, but the number of flows in each tier varies. For the five runs, the mixes of flows in each tier are:

1. T1 / T2 / T4 = 13 / 11 / 11
2. T1 / T2 / T4 = 25 / 5 / 5
3. T1 / T2 / T4 = 5 / 25 / 5
4. T1 / T2 / T4 = 5 / 5 / 25
5. T1 / T2 / T4 = 20 / 10 / 5

The per-flow weighted max-min fair allocation of the available 948 Mbps is shown in Table 23. Even though the number of flows is constant at 35, the different tiering allocations present different challenges with respect to delivering approximate weighted max-min fair allocation. All flows have the same RTT of 80 ms and the simulation time remains 1000 s.

Fig. 8 provides the weighted throughput fairness results from the five simulation runs of TierM. From the JFI, the ABB scheme with 3 and 4 bins provides a close approximation to DRR-CoDel. However ABB is also clearly shown to be less fair than DRR-CoDel by the min-max ratio. Run 4 provides the most significant challenge.

Table 21
Simulation TierG flow throughput (Mbps): mean / CoV.

#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 22
Simulation TierG flow RTT (seconds): 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			

Table 23
TierM: Weighted max-min fair throughput.

Run	T1	T2	T4
1	12.0	24.0	48.0
2	17.2	34.5	68.9
3	12.6	25.3	50.6
4	8.2	16.5	33.0
5	15.8	31.6	63.2

The average tiered flow throughput for TierM is given in Table 24. The small CoV values are indications that the fairness within the same tier is improved when all flows have the same RTT. Among the five runs, the worst approximation of the weighted fairness comes at run 4. It has 25 Tier4 FTP flows and only 5 flows each in Tier1 and Tier2. It can be observed in Table 24 that the cause of the unfairness is excessive bandwidth allocated to the Tier1 and Tier2 flows. This is evidence of a small number of Tier1 or Tier2 flows sharing a bin with a larger number of Tier4 flows as described in Section 7.1.

Weighted max-min fairness discrepancies associated with the ABB schemes consistently showed over-provisioning of tier 1 flows at the expense of tier 4. For ABB-3 and ABB-4, the under provisioning of tier 4 was always smaller than 10%. Total channel utilization with ABB slightly exceeded that of DRR-CoDel.

The average RTT information is given in Table 25. The ABB scheme is shown to maintain similar queuing delay for all flow tiers. Again the ABB scheme is able to provide similar performance to that of DRR-CoDel.

The ABB dynamic behavior for the two variants of the TIER simulation are shown in Tables 26 and 27. The disruption time remains low in all cases, indicating the scheme is demonstrating satisfactorily stable behavior.

7.3. ***UDP*** *simulation results*

The **UDP** simulation is designed to assess the ability of ABB to isolate unresponsive high bandwidth UDP flows. Our prior work [32] showed that single queue schemes such as CoDel or PIE fail to provide UDP isolation. Here, we assess the ability of the use of multiple queues in ABB to provide better protection from unresponsive UDP flows.

The simulation is set up with three flow tiers: Tier1, Tier2, and Tier4 with weights of 1, 2, and 4 respectively. There are 11 FTP flows active in each tier. 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. The weighted max-min fair allocations to the 12 total flows in Tier 1, Tier 2, and Tier 4 are 11.3 Mbps, 22.6 Mbps, and 45.1 Mbps respectively.

Table 28 provides both FTP and UDP flow throughput for each tier. For FTP, the throughput reported is the average of the 11 FTP flows assigned to each tier. As expected, single queue CoDel does not provide UDP isolation at all. DRR-CoDel provides very good UDP isolation with nearly weighted max-min fair allocation but slightly favors the unresponsive UDP flows. In the presence of such unresponsive flows, the degree of unfairness can further increase with fewer queues in use as in the case of ABB.

ABB with both 2 and 3 bins provides little if any UDP isolation. The results are similar to that of single queue CoDel. With 4 and

Table 24
Simulation TierM flow throughput (Mbps): mean / CoV.

#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 25
Simulation TierM flow RTT (seconds): 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 26
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)	12,385 (0.22)
ABB-3	2486 (0.19)	5554 (0.23)	9001 (0.26)	12,564 (0.27)	16,374 (0.29)
ABB-4	2590 (0.20)	6059 (0.25)	10,263 (0.29)	14,445 (0.31)	19,216 (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)

5 bins the situation is measurably improved but remains poor. The disparities are the worst in Tier 1.

Note that this scenario involves UDP flows in three different tiers. To provide weighted UDP flow isolation with ABB it is necessary to isolate unresponsive UDP flows into additional per-tier dedicated bins. ABB already requires tracking the state of each flow. Therefore, it would be straightforward to track per flow arrival and service rates and identify those flows with a significant disparity as unresponsive.

Even with additional bins for the unresponsive flows, it is still likely that ABB as currently implemented would not be able to provide max-min fairness in the presence of unresponsive flows.

Table 27
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	10,257 (0.29)	10,514 (0.30)	10,067 (0.29)	9734 (0.28)	10,426 (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 28
Flow Throughput by Tiers for Simulation 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

Table 29
ABB Dynamics with Simulation 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)

Table 30
Application flow throughput by types and tiers for simulation 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

While such binning can protect the unresponsive flows from each other, it does not necessarily protect responsive flows from unresponsive ones.

The source of the unfairness is attributable to the use of weighted DRR by ABB. When responsive flows with elastic demands back off in response to network congestion, the associated queue may not contain sufficient packets to consume the queue's remaining deficit count before the queue becomes empty. The use of an AQM system such as CoDel increases the likelihood of this situation occurring, and when it does occur, the queue effectively donates its remaining deficit for use by other queues. The queues associated with high rate unresponsive flows always remain full. For max-min fair sharing, an unresponsive flow should never have sustained consumption of bandwidth that exceeds the consumption of any responsive flow with elastic demand. Therefore, max-min fairness will be maintained if and only if queues containing responsive flows with elastic demands always contain more data than the DRR quantum. It follows that the larger the DRR quantum is, the more serious this source of unfairness will be. We are presently investigating the use of dynamic adjustment of the DRR quanta of queues containing high rate unresponsive flows to ensure that max-min fairness is preserved.

The dynamic behavior of ABB is shown in Table 29. Given the number of flows involved, the bin switching rate is not low but is consistent with that of TierG and TierM with similar numbers of TCP flows running. One noticeable difference between the disruption time of this simulation 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 builds up its queue while a bin with only responsive TCP flows maintains short queue length under CoDel. This situation would also be prevented by quarantining unresponsive flows in a dedicated bin or bins.

7.4. *APP* simulation results

The **APP** simulation is designed to evaluate realistic traffic mixes including FTP, HAS (http adaptive streaming), and web browsing. We are particularly interested in evaluating the ability of ABB to provide benefit to higher tier application flows under heavy loads.

7.4.1. The TAPPG simulation

The first variant, called TAPPG, runs a total 100 flows. There are 30 web flows and 20 HAS flows in Tier1, 20 HAS flows in Tier2, 20 HAS flows at Tier4, and 10 FTP flows at Tier4. Therefore, the total workload is comprised of 60 HAS flows, 30 web flows, and 10 FTP flows. 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 30 provides the average FTP throughput and the average HAS throughput by tiers. Table 31 shows HAS application layer video throughput. Comparative analysis shows that HAS overhead is approximately 10% on average.

Under DRR-CoDel the combined consumption of the HAS and FTP flows is 911.3 Mbps, and total bandwidth consumed by the all of Web flows is approximately 20 Mbps. Under the assumption that the maximum demand of the HAS flows is 16.5 Mbps when HAS protocol overhead is included, then for the HAS flows the weighted max-min fair shares of the 911.3 Mbps are {5.81, 11.63, 16.5} for

Table 31
HAS performance by tiers for simulation 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

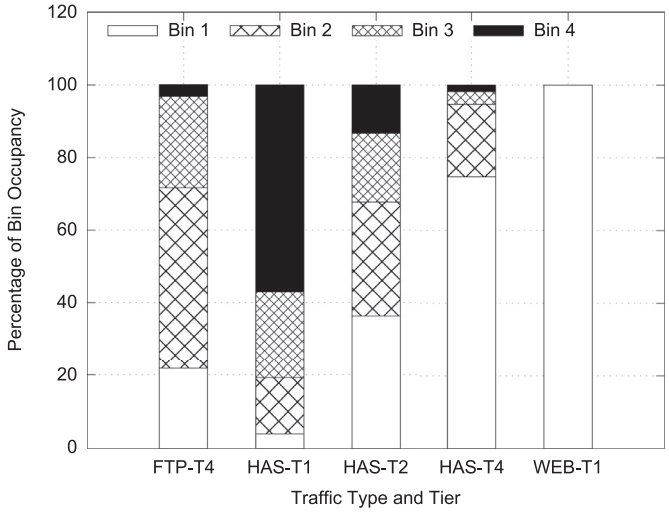


Fig. 9. TAPPG - Bin occupancy in percentage of time for each tier and traffic type.

Tiers 1, 2, and 4, and the weighted max-min fair share for the FTP flows in Tier 4 is 23.25.

Because HAS dynamically switches demands among discrete rates and employs an application layer back off in addition to TCP back off, perfect weighted max-min fair sharing is not to be expected when HAS flows are competing with FTP. Dividing the DRR-CoDel values for the HAS flows in Table 30 by their weighted max-min fair counterparts, {5.81, 11.63, 16.5}, yields {1.06, 0.93, 0.92}. Thus, all values for DRR-CoDel are within 10% of max-min fair but considerable bandwidth has been "donated" to FTP.

ABB-3 and ABB-4 actually do a better job of restricting the FTP flows than does DRR-CoDel, but the HAS flows are not quite as fairly allocated with Tier 1 receiving even more than with DRR-CoDel.

Single queue CoDel, which does not support weighted allocation, fails to provide any benefit to higher tier HAS users.

To provide additional insight on 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 each bin. Under ABB-4, the results are given in Fig. 9. 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 met better.

Table 31 provides the HAS performance results. Under single queue CoDel, the video play rates are similar across tiers and the adaptation counts for the HAS flows are very high. DRR-CoDel delivers approximate weighted max-min fair sharing, and HAS flows are able to achieve higher play rates at higher tiers with low adaptation counts across all tiers. ABB-3 and ABB-4 deliver good differentiation of play rates across different tiers, but significantly higher adaptation counts in Tier1 and Tier2.

Table 32
Application flow throughput by types and tiers for simulation 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 33
HAS performance by tiers for simulation 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

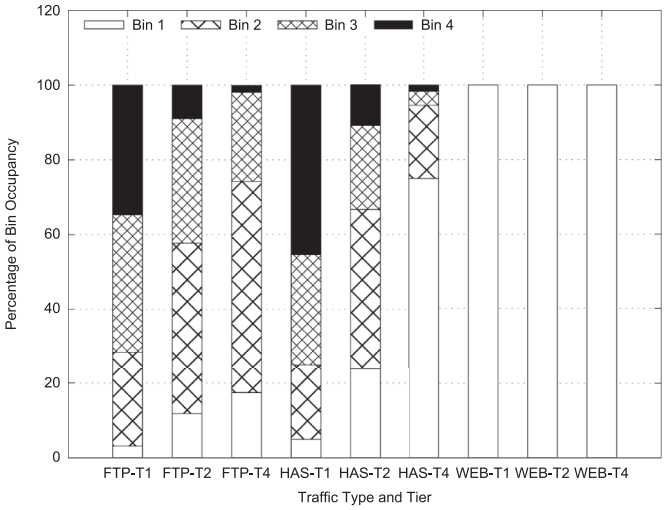


Fig. 10. TAPPR - Bin occupancy in percentage of time for each tier and traffic type.

7.4.2. The TAPPR simulation

We also ran a variant, called TAPPR, 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.

Tables 32 and 33 provide the application throughput and HAS performance results. The throughput results show that, as before, 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 Fig. 10. 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 simulation, it is not difficult to draw the same conclusion we drew before. Under ABB, HAS flows of higher tiers are able to achieve much better HAS video play rate and lower adaption count than with single queue CoDel.

7.4.3. Web response time

The web response time in both APP simulation variants was excellent. In all cases, including DRR-CoDel, single queue CoDel and the three ABB variants mean web response time varied between 0.18 and 0.20 s. Standard deviation varied between 0.060 and 0.065.

8. Conclusion

The focus of this paper has been downstream bandwidth management systems for subscription based, shared medium Internet access networks. We concentrate specifically on the problems of developing a system capable of efficiently approximating weighted max-min fair sharing of bandwidth while maintaining latency that is low enough to support interactive audio or video conferencing.

We have presented a new novel approach called adaptive bandwidth binning (ABB) as a solution. ABB uses a small fixed number of bins and periodically re-maps the flows into these bins based on the flow's recent bandwidth consumption. Flows in the same bin share a queue managed by CoDel for low queuing delay.

This paper reports the results of our initial evaluation of the effectiveness of ABB. The results were derived from *ns-2* simulations conducted on simulated DOCSIS-3.0 and DOCSIS-3.1 networks. In these studies, we evaluated ABB (with two, three, and four bins), along with DRR-CoDel, SFQ-CoDel, and single queue CoDel.

Weighted DRR-CoDel, which is known to provide approximate weighted max-min fair sharing and low latency at the cost of considerable complexity, serves as the performance target. SFQ-CoDel is multi-bin based approach that provides unweighted max-min fair sharing and low latency, but it does not directly support tiers and the number of bins is typically $O(average\_flow\_count)$. Single queue CoDel was also included for reference purpose. It does not provide max-min fair sharing in either single tier or multi-tier environments.

We first analyzed the ABB scheme in the single tier environment where all flows have unit weight. Under our test workloads, ABB with only three or four bins was able to provide reasonable performance in terms of fairness and latency comparable to that of DRR-CoDel and SFQ-CoDel. The ABB system was also able to ameliorate the TCP RTT unfairness issue that impacts single queue management schemes [32].

In a multi-tier environment, our results and analysis showed that under our test workloads ABB was 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 weighted max-min fair sharing on multi-tier systems under heavy loads.

We also evaluated a scenario in which responsive TCP flows and high bandwidth unresponsive flows compete. We found that ABB, as currently implemented, does not effectively isolate unresponsive flows. Nevertheless, we are confident that a simple extension, the addition of a bin (or per-tier bins) dedicated to unresponsive flows, will provide excellent isolation.

We further analyzed application performance with simulations that involved realistic types of traffic including FTP, HAS, and web flows. We found that ABB with three or four bins provides weighted HAS service comparable to that of DRR-CoDel with three active service tiers.

There are some obvious ways in which to extend this work. Most obviously isolation of unresponsive flows to their own bin(s) should be added. Our choice of the remapping interval time, τ, was based upon a small number of preliminary tests. We strongly suspect that a dynamic adjustment of τ based upon changing workload conditions would provide improved performance.

The present bandwidth consumption and bin assignment algorithms also merit further study. Currently, bin assignment is based on a flow's recent downstream transmission rate. Using the flow's downstream arrival rate or some combination of the two rates should be investigated.

References

[1] 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 URL http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-MULPIv3.1-I06-150611.pdf.

[2] F. Baker, G. Fairhurst, IETF recommendations regarding active queue management, 2015, (https://tools.ietf.org/html/rfc7567).

[3] K. Nichols, V. Jacobson, Controlling queue delay, Commun. ACM 55 (7) (2012) 42–50.

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

[5] R. Pan, P. Natarajan, C. Piglione, M.S. Prabhu, V. Subramanian, F. Baker, B. VerSteeg, PIE: a lightweight control scheme to address the bufferbloat problem, in: High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on, IEEE, 2013, pp. 148–155.

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

[7] J. Gettys, K. Nichols, Bufferbloat: dark buffers in the Internet, Commun. ACM 55 (1) (2012) 57–65.

[8] M. Shreedhar, G. Varghese, Efficient fair queuing using deficit round-robin, IEEE/ACM Trans. Netw. 4 (3) (1996) 375–385.

[9] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the internet, IEEE/ACM Trans. Netw. (TON) 7 (4) (1999) 458–472.

[10] J. Mills, J. Livingood, R. Woundy, T. Klieber, C. Bastian, Comcast's protocol-agnostic congestion management system, 2010URL http://tools.ietf.org/html/rfc6057.

[11] J.-Y. Le Boudec, Rate adaptation, congestion control and fairness: a tutorial, URL http://moodle.epfl.ch/pluginfile.php/4391/course/section/8686/cc.pdf 2012.

[12] S. Keshav, An Engineering Approach to Computer Networking: ATM Networks, the Internet and Telephone Network, Addison Wesley, Reading, Massachusetts, 1997.

[13] J. Nagle, On packet switches with infinite storage, Commun. IEEE Trans. 35 (4) (1987) 435–438.

[14] A.K.J. Parekh, A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks, 1992 Ph.D. thesis.

[15] A. Demers, S. Keshav, S. Shenker, Analysis and simulation of a fair queueing algorithm, in: ACM SIGCOMM Computer Communication Review, 19, ACM, 1989, pp. 1–12.

[16] S.J. Golestani, A self-clocked fair queueing scheme for broadband applications, in: INFOCOM'94. Networking for Global Communications., 13th Proceedings IEEE, IEEE, 1994, pp. 636–646.

[17] S. Ramabhadran, J. Pasquale, The stratified round robin scheduler: design, analysis and implementation, IEEE/ACM Trans. Netw. (TON) 14 (6) (2006) 1362–1373.

[18] X. Yuan, Z. 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, 2, IEEE, 2005, pp. 831–842.

[19] Z. Dwekat, G.N. Rouskas, A practical fair queuing scheduler: simplification through quantization, Comput. Netw. 55 (10) (2011) 2392–2406.

[20] S.Y. Cheung, C.S. Pencea, BSFQ: Bin sort fair queueing, in: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 3, IEEE, 2002, pp. 1640–1649.

[21] P.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, IEEE, 1990, pp. 733–740.

[22] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. 1 (4) (1993) 397–413.

[23] S. Floyd, R. Gummadi, S. Shenker, Adaptive RED: an algorithm for increasing the robustness of RED's active queue management, Technical Report, 2001. URL http://www.icir.org/floyd/papers.html.

[24] W.-c. Feng, K.G. Shin, D.D. Kandlur, D. Saha, The BLUE active queue management algorithms, IEEE/ACM Trans. Netw. (TON) 10 (4) (2002) 513–528.

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

[26] L. Xue, S. Kumar, C. Cui, P. Kondikoppa, C.-H. Chiu, S.-J. Park, AFCD: an approximated-fair and controlled-delay queuing for high speed networks, in: Computer Communications and Networks (ICCCN), 2013 22nd International Conference on, IEEE, 2013, pp. 1–7.

[27] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Ghettys, E. Dumazet, FlowQueue-Codel (draft-ietf-aqm-fq-codel-02), 2015, (https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-02).
[28] The network simulator – ns-2, URL http://www.isi.edu/nsnam/ns/.
[29] K. Nichols, SFQ-CoDel ns-2 source code, 2013, (http://www.pollere.net/Txtdocs/).
[30] G. Hong, Downstream Bandwidth Management For Emerging DOCSIS-based Networks, 2015 Dissertation. URL http://tigerprints.clemson.edu/all_dissertations/1579/.
[31] J. Martin, J. Westall, A simulation model of the DOCSIS protocol, Simulation 83 (2) (2007) 139–155.
[32] G. Hong, J. Martin, J.M. Westall, On fairness and application performance of active queue management in broadband cable networks, Comput. Netw. 91 (2015) 390–406.
[33] J.C.R. Bennett, H. 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, 1, IEEE, 1996, pp. 120–128.
[34] R. Pan, L. Breslau, B. Prabhakar, S. Shenker, Approximate fairness through differential dropping, ACM SIGCOMM Comput. Commun. Rev. 33 (2) (2003) 23–39.
[35] R.K. Jain, D.-M. W. Chiu, W.R. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, 1984URL http://www1.cse.wustl.edu/~jain/papers/ftp/fairness.pdf.
[36] M. Shreedhar, G. Varghese, Efficient fair queueing using deficit round robin, in: ACM SIGCOMM Computer Communication Review, 25, ACM, 1995, pp. 231–242.
[37] E. Gavaletz, J. Kaur, Decomposing RTT-unfairness in transport protocols, in: Local and Metropolitan Area Networks (LANMAN), 2010 17th IEEE Workshop on, IEEE, 2010, pp. 1–6.

Gongbing Hong received the Ph.D. degree from Clemson University. He is currently an assistant professor of computer science in the Department of Information Systems and Computer Science, Georgia College & State University. His past work experience included more than a decade engineering career in the cable system division at Cisco / Scientific-Atlanta. His research interests include broadband access, network performance analysis, and cloud computing.

James Martin received the Ph.D. degree from North Carolina State University. Currently, he is an associate professor in the School of Computing, Clemson University. Prior to joining Clemson, he was a consultant for Gartner, and prior to that, a software engineer for IBM. His research interests include broadband access, wireless networks, Internet protocols, and network performance analysis. Current research projects include heterogeneous wireless systems and DOCSIS3.x cable access networks. He has received funding from NSF, NASA, the Department of Justice, BMW, CableLabs, Cisco, Comcast, Cox, Huawei, and IBM.

James M. Westall received the M.S. degree in computer science and the Ph.D. degree in mathematics from the University of North Carolina at Chapel Hill. He is a professor emeritus and research professor of computer science in the School of Computing, Clemson University. His research interests include measurement and modeling of computer systems and networks, computer graphics, and high-performance computing.