Computer Networks 91 (2015) 390–406

Contents lists available at ScienceDirect

Computer Networks

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

On fairness and application performance of active queue management in broadband cable networks

Gongbing Hong*, James Martin, James M. Westall

School of Computing, Clemson University, 100 McAdams Hall, Clemson, SC 29634, USA

ARTICLE INFO

Article history:
Received 5 March 2015
Revised 29 June 2015
Accepted 27 August 2015
Available online 7 September 2015

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

ABSTRACT

We evaluate the performance of modern delay-based active queue management (AQM) algorithms in downstream DOCSIS 3.0 cable environments. Our simulation-based study involves scenarios that include tiered service levels and application workloads that include FTP, HTTP-based adaptive streaming (HAS), VoIP, and web traffic.

Our results show that AQM maintains target queuing delays and consequently provides strong isolation between responsive high bandwidth and low bandwidth latency sensitive flows. We also show that lowering target queuing delays exacerbates TCP's RTT unfairness. Nevertheless, in the scenarios that we studied, observed application performance was not significantly impacted by the specific AQM in use.

With the potential large deployment of AQM imminent for DOCSIS 3.0, the analysis presented in this paper provides timely feedback to the community concerning how delay-based AQM can manage bandwidth allocation fairness and application performance in realistic downstream DOCSIS 3.0 cable network systems.

1. Introduction

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

Recent enhancements to cable access standards by CableLabs, the research and development organization for MSOs, are ensuring the convergence is technically possible. Current generation systems based on Data Over Cable Service Interface Specification (DOCSIS) version 3.0 (D3.0) can support multiple downstream, fixed bandwidth channels (usually 6.0 or 8.0 MHz) bonded together and multiple upstream channels (usually 6.4 MHz) bonded together[1]. A common configuration involving eight bonded downstream channels and four bonded upstream channels can support data rates up to 320 Mbits/s downstream and 120 Mbits/s upstream. Emerging systems will be based on the recently released DOCSIS 3.1 (D3.1) standard [2]. Higher channel bandwidth along with more efficient modulation and coding will allow downstream and upstream data rates of up to 10 and 1 Gbps, respectively.

* Corresponding author. Tel.: +1 6786369539.
E-mail addresses: hgb.bus@gmail.com, gongbih@clemson.edu (G. Hong), jmarty@clemson.edu (J. Martin), westall@clemson.edu (J.M. Westall).

http://dx.doi.org/10.1016/j.comnet.2015.08.018
1389-1286/

Scheduling and queue management disciplines are fundamental to computer networking and have been studied from many different perspectives. Common objectives include congestion management and provision of fair sharing or differentiated services in IP networks. In spite of this body of knowledge, access networks can still suffer from known problems including bufferbloat, TCP RTT unfairness, and vulnerability to unresponsive flows [3–8]. Shared medium cable networks present additional challenges. On one hand they must attempt to ensure that bandwidth sharing and latency objectives among many competing flows are met. On the other hand, regulatory constraints and concerns pertaining to so-called "net neutrality" limit the mechanisms that can be used in doing so. These regulatory concerns are so strong that flow-aware, multi-queue fair share scheduling systems such as deficit round robin (DRR) or self-clocked fair queuing (SCFQ) that would greatly facilitate bandwidth management are not currently in use.

Two flow-agnostic, single queue active queue management (AQM) algorithms that have been proposed since 2012 are Controlled Delay (CoDel) [4,9] and Proportional Integral Controller Enhanced (PIE) [10,11]. CoDel has been evaluated in different types of networks including 802.11 and high speed wired networks. PIE, which is not yet widely deployed, is based in part on concepts that were introduced previously [12]. The D3.1 standard requires that a cable modem (CM) manages upstream best effort traffic using PIE. Wide deployment of D3.1 equipment is not expected for several years. This prompted CableLabs to modify the current, and widely deployed, D3.0 standard to allow the immediate use of the PIE AQM for upstream traffic [1]. For downstream traffic, the Cable Modem Termination System (CMTS) must support a published AQM algorithm, not necessarily CoDel or PIE, in both D3.0 and D3.1. The standards indicate the equipment must support AQM by default, although a network operator can choose to override the default configuration, possibly selecting to not use AQM or to use an alternative method. The intent of adding the AQM requirement to the DOCSIS standards is to ensure that network operators have tools that can address bufferbloat in a consistent manner. With large scale deployments of delay-based AQM algorithms imminent, there is critical need to understand how these schemes will behave in current and emerging cable networks.

In prior work we have developed an *ns2*-based simulation model of DOCSIS 3.0 [13]. We have used the simulator to explore upstream buffer management in CMs [14]. The focus of that paper was upon preventing upstream bufferbloat in the CM. The validation of downstream scheduling disciplines in bonded channel environments was the focus of [15]. Extending our ongoing work in the area of AQM [16], the research presented in this paper provides detailed analysis on the effectiveness of recently proposed delay-based AQM schemes applied to downstream traffic in cable access networks. Our simulation framework utilizes realistic scenarios involving application traffic models such as FTP, VoIP, web browsing, and HTTP-based adaptive streaming (HAS). The analysis includes scenarios involving both single and bonded downstream channels. We consider scenarios that involve different service tiers, where different cable users are provisioned with different data rates.

To the best of our knowledge, our study is the first to assess the ability of modern delay-based AQM schemes to manage fairness and application performance in realistic, single or bonded channel downstream DOCSIS 3.0 cable network scenarios. The results presented in this paper address the following questions: (1) How effectively do CoDel and PIE support fairness and application performance in realistic cable network scenarios? (2) Are there undesirable side effects when the AQM interacts with tiered service levels? (3) How effectively do the schemes isolate responsive traffic from unresponsive flows?

This paper is organized as follows. Section 2 summarizes relevant research that has been published in the literature. Section 3 introduces the experimental methodology. The next three sections contain the results of our simulation studies with the focus of each section being one of the three fundamental questions enumerated above. Section 7 provides conclusions and identifies our next steps.

2. Related work

Bufferbloat has received significant attention recently [3,4]. Bufferbloat is a side-effect of congestion that occurs when network equipment is provisioned with large unmanaged buffer queues. Its effects are persistently high queue levels that in turn lead to large and sustained packet latency.

AQM has long been considered a solution to bufferbloat. The random early detection (RED) algorithm manages a queue by randomly dropping packets in a manner in which the random drop rate is dynamically adjusted based on an average queue size estimate and a configured maximum allowed drop rate (referred to as *maxp*) [7]. Most RED implementations offer the 'gentle' option where the drop rate increases linearly from *maxp* to 1 once the average queue level exceeds the target queue size [8,17].

While RED is widely available, it is not widely used. It has been shown that the average queue delay with RED is sensitive to traffic loads and to parameter settings[18,19]. 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 [20]. This adaptation is performed periodically. We refer to this parameter as the *control_interval*. We refer the reader to [21] for a thorough summary of relevant AQM research.

CoDel and PIE are AQM algorithms that were specifically designed to address the shortcomings of (A)RED. Both AQMs are delay-based as they proactively drop packets to ensure average packet queue 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 timescale 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 [4] and [10], respectively.

The PIE algorithm performs early packet drops as packets arrive at the queue. The CoDel algorithm, as originally proposed in [4,9] performs early packet drops as queued packets

are serviced. The DOCSIS vendor community has expressed concern in implementing a head-drop AQM such as CoDel in a CM due to complications with hardware buffer control logic. This is likely to be true in other low cost network devices such as Ethernet switches or WiFi APs.

AQM involving per-flow accounting or the use of multiple queues has been proposed to address the combined problem of fairness and bufferbloat. Proposals including Fair RED (FRED), Stochastic Fair BLUE, and Approximate Fair Dropping (AFD) strive to achieve fairness while retaining benefits of AQM [12,19,22]. Recent proposals have considered per-flow accounting along with delay-based AQM [23,24]. An AQM scheme called SFQ-CoDel [25], which is based on CoDel and Stochastic Fair Queuing (SFQ) [26], has been actively evaluated [27,28].

In follow-on work, we are exploring the combined problem of buffer management and packet scheduling when multiple queues are involved. However, given the interest of the cable community in single queue delay-based AQM, we limit the scope of the research presented in this paper to low complexity AQM schemes.

2.1. Overview of DOCSIS cable networks

Modern broadband Internet access over cable involves a hybrid-fiber coaxial (HFC) infrastructure with a DOCSIS MAC protocol operating between the CMTS and the subscriber CMs. The most widely deployed version of the standard, DOCSIS 3.0 (D3.0), uses 6 MHz (or 8 MHz in certain regions) of bandwidth for the shared downstream channel and up to 6.4 MHz of bandwidth for the shared upstream channel. The 6 MHz downstream channel supports physical layer data rates up to 42.88 Mbps (55.62 Mbps in 8 MHz regions) and the upstream channel supports data rates up to 30.72 Mbps. Prior to D3.0, CMs were limited to a single downstream channel and a single upstream channel (although, for the purposes of load balancing, the CMTS could dynamically change a CM's downstream or upstream channel assignment). D3.0 allows a CM to support multiple downstream or upstream channels. A bonding group is a set of channels that specific subscriber traffic can use. A downstream service group is the complete set of channels that can potentially reach a CM. A bonding group is a specific subset of channels from the service group assigned to carry subscriber traffic.

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

Upstream operation is a shared bus relying on a hybrid contention-based request MAC protocol for managing access to the channel by competing CMs. Similar to the service model defined by the Worldwide Interoperability for Microwave Access (WiMAX) [29], upstream DOCSIS provides multiple services to support different classes of traffic. The centralized upstream scheduler operating at the CMTS allocates periodic grants to modems (i.e., the unsolicited grant service) or the CM requests a bandwidth allocation using either a request piggy-backed to a previously allocated transmission or a contention-based request mechanism.

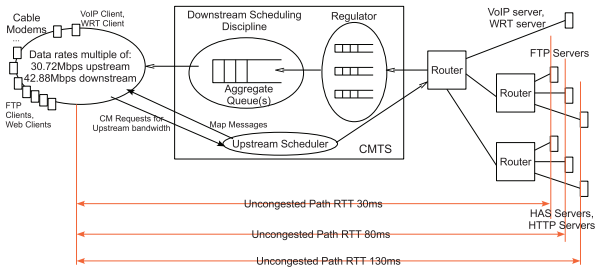


Fig. 1. Simulated cable network diagram.

A service tier can be defined by any combination of these service parameters. In this paper, we define a downstream service tier by the rate shaping and limiting parameters. We refer to the sustained traffic rate simply as the service rate.

3. Methodology

Fig. 1 illustrates the simulation model used in these studies. We model a single DOCSIS MAC domain in which one CMTS interacts with a number of CMs. Each CM represents a subscriber who, in practice, might have multiple TCP/IP devices interacting with a variety of Internet services. However, in the experiments described here, each CM hosts a single data flow.

The model assumes ideal channels that offer downstream and upstream link layer data rates of 38 and 27 Mbps, respectively. We also consider scenarios involving four bonded downstream channels providing an effective data rate of 152 Mbps.

Our analysis considers scenarios with and without imposition of service rate limits. The regulator component of the simulator, shown in Fig. 1, models a DOCSIS token bucket with service rate parameters being the desired maximum sustained rate and bucket size (specified in bytes). We use the default DOCSIS token bucket size of 3044 bytes. A widely used option called Powerboost [30] employs a significantly larger bucket size to allow a flow to exceed its service rate limit during an initial traffic burst. This can provide performance benefits to brief connections in *uncongested* situations. We do not consider this option in our analysis because we are primarily interested in AQM behaviors during congested scenarios.

The queue management algorithms are located in the Downstream Scheduling Discipline component of Fig. 1. For DRR, there is a dedicated queue for each flow. For all other queue management schemes, all flows are served through a single shared FIFO queue.

Open source ns2 implementations of adaptive RED, a tail-drop variant of CoDel, and PIE were adapted by us to work in our ns2 DOCSIS simulation framework. The source code

for the AQMs that we used in our analysis is available to the research community.[1]

We analyze several scenarios using varying numbers of downstream application flows including FTP, VoIP, web, and HAS video streams. The simulated client sides of these applications run on nodes attached to CMs. The simulated servers run on nodes located outside the cable network. The cable network endpoint of each simulated flow is attached to a unique CM.

To model Internet path diversity, we set the propagation delays of the application server access links to different values. As illustrated in Fig. 1, the range of uncongested path round trip times varies from 30 to 130 ms. This captures the inherent variability that subscriber flows might experience with respect to the physical distance between communication endpoints.

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

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

- TCP: TCP/Sack (ns2 TCP agent: Agent/TCP/Sack1).
- Maximum window/cwnd: 10000 segments.
- Initial window: 2 segments.
- Delayed ACK mean interval: 100 ms.
- Maximum segment size: 1460 bytes.
- Number of segment arrivals required to trigger an ACK: 2

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

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

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

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

[1] Available at http://people.cs.clemson.edu/~jmarty/AQM/AQMPaper.html

3.1. Traffic models and performance metrics

The results presented in this paper involve workloads consisting of FTP, HAS, web, and UDP CBR traffic. Other than HAS, the traffic models we use are those provided in ns2.

3.1.1. HAS traffic model for ns2

As described in Sandvine's most recent Internet traffic report [31], Internet video streaming, commonly known as HAS, consumes more than 50% of the downstream bandwidth delivered to fixed access endpoints.

The interaction between an HAS client (i.e., the player) and server has been established in recent academic research [32–35]. In an HAS system, video and audio content is encoded and made available at servers located either at the content provider's facilities or distributed to locations in the Internet by a content delivery network provider. Multiple representations of the content are created that reflect a range of possible encoded bit rates corresponding to different levels of video quality and subsequently different levels of bandwidth consumption. While the range of bit rates is continually being extended, the literature suggests that a bit rate range of 0.50–5.0 Mbps is reasonable [32,36,37]. The client requests portions of the content in chunks referred to as *segments*. A segment size between 2 and 10 seconds is reasonable [36,38]. The client maintains a playback buffer that serves to compensate for the jitter in the traffic arrival process. The literature suggests that a playback buffer capacity ranging from 60 to 240 s is reasonable [36,38].

The HAS adaptation algorithm, which determines the video quality for the next segment requested by the client, is an active research topic. Initial ideas explored *capacity-based* approaches where the client monitors the rate of arriving video traffic and selects the quality level based on a prediction of the available TCP bandwidth [32,33,39–42]. If the client predicts that available bandwidth might decrease over the next segment time interval, it requests a lower quality segment. If the prediction suggests that conditions are improving the client will 'switch-up' and request a higher quality segment. Other research, however, suggests that it is difficult for HAS to reliably predict available TCP bandwidth [33,42–44]. In addition to variability of available bandwidth due to competing traffic in the network or from changing wireless channel conditions, rate-limited applications can induce pathological behaviors due to complex dynamics between the application and TCP's congestion control algorithm [34]. The use of *buffer-based* adaptation has been suggested to avoid the issues [36].

Evaluating a particular HAS design is challenging due to the many factors that impact an end user's perceived quality [45–47]. While the issue of assessing HAS quality of experience is intense study, the literature does suggest that the following measures are useful for evaluating HAS [33,35,45,48,49]:

- *videoPlaybackRate*: Attributes of the stream such as the resolution, pixel density, frame rate as well as end device capabilities all determine the base quality of the video stream. This measure represents the average bit rate of the stream based on the rate (in Mbps) at which data are

dequeued from the playback buffer. If the video player stalls, samples are not recorded.

- *adaptationRate*: It has been shown that frequent adaptations and sudden changes to video quality are distracting [48,50]. This metric counts the number of adaptations that occur during the simulation time. The measure is normalized to represent the rate of adaptations per hour.
- *rebufferTime*: Buffer stalls have a significant impact on perceived quality [45,49]. Based on the ratio of total time the player is stalled to the stream duration, this measure represents the percentage of time that the player is stalled.
- *applicationBias*: The system should provide fair allocation. We focus primarily on the fairness allocated to groups of flows. This measure computes the percentage above or below the expected max–min fair allocation.

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

When the video player requires a new segment and the playback buffer holds less than one segment of video data, the player moves to a stall state where it remains until a configured number of segments arrive (2 segments by default). The client adaptation algorithm is based on a previously published capacity-based adaptation algorithm [40]. Each time the client issues a new segment request, it computes the ratio (μ as defined in equation (1)) of the time required to play a segment (referred to as the media segment duration or MSD) to the download time of the previous segment (referred to as the segment fetch time or SFT).

$$\mu = \frac{\text{MSD}}{\text{SFT}} \tag{1}$$

The algorithm switches to the next higher quality level if

$$\mu > 1 + \varepsilon \tag{2}$$

where ε is defined as

$$\varepsilon = \max\left\{\frac{b_{r_{i+1}} - b_{r_i}}{b_{r_i}}, \forall i = 0, 1, \ldots, n-1\right\}. \tag{3}$$

The b_{r_i} represents the encoded bit rate of the ith representation where b_{r_n} represents the highest bit rate.

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

$$\mu < \gamma_d. \tag{4}$$

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

$$b_{r_i} < \mu b_c \tag{5}$$

where b_c represents the bit rate of the current segment.

The client supports a configured maximum playback buffer size. The client will not issue a new segment request once the buffer is full. As pointed out in recent work, this algorithm can be improved through more careful selection of γ_d and by computing μ based on multiple SFT samples [51]. However, for the research presented in this paper, we chose to use a simple published adaptation algorithm. The objective of the HAS experiment is to explore the impacts of different buffer management strategies on HAS performance.

3.1.2. VoIP performance metric

To assess the impact that high bandwidth applications might have on low bandwidth, latency sensitive flows, we establish a VoIP performance monitor between a server node and a CM. A 'talker' located on the VoIP server node (in Fig. 1) sends a stream of UDP traffic in a manner similar to a G.711 session to a 'listener' located on the CM labeled VoIP client. We estimate the call quality (referred to as the R-value) using the technique described in [52]. The R-value ranges from 0 to 100. Toll quality requires an R-value of 70 or higher. The model assumes that two 10 ms 'chunks' of digitized voice data are sent in an IP packet (of size 238 bytes) every 0.020 s.

3.1.3. Web performance traffic generator and metric

The web traffic model is based on well understood behaviors of web browsing users [53,54]. The ns2 simulation package provides extensive support for web traffic models. The model includes a set of web servers (identified in Fig. 1) and web browsing users. A variable number of CMs are configured as web clients. Based on empirically derived distributions, these CMs periodically issue HTTP GET requests to randomly selected servers. The model is parameterized by user think times and web object sizes derived from heavy-tailed Pareto distributions.

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

3.1.4. Fairness metric

We use Jain's Fairness Index (JFI) [55] and the min–max ratio (MMR) to quantify fairness in achieved throughput among competing flows. The JFI is a widely used assessment of how achieved resource allocation differs from the desired outcome. We assume that *max–min fair* throughput allocation criterion is the desired fairness objective. An allocation is max–min fair if it is not possible to increase the rate allocated to any user in the system without decreasing

the rate allocated to any other user who is receiving an already lower or equal rate [56]. It has been shown that packet scheduling algorithms that implement close approximations of generalized processor sharing (GPS) [57,58], such as DRR packet scheduling [59], can achieve max–min fairness. In recent work [15], we have shown that DRR packet scheduling also achieves max–min fair allocation in downstream D3.0 bonded channel environments when no channel is assigned to more than one bonding group at a time.

JFI is computed as follows. Suppose there are n competing flows. Let $x_i = {}^{T_i}/_{r_i}$ where T_i is the achieved throughput of the ith flow and r_i is the expected outcome based upon the max–min fair criterion. The JFI is defined as:

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

The MMR is defined as:

$$\frac{\min\{x_i\}}{\max\{x_i\}}.$$

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

3.2. Experimental definitions

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

3.2.1. Set 1

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

3.2.2. Set 2

We explore implications of the use of different service tiers.

3.2.3. Set 3

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

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

Table 1
Experiment definition.

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

The *DT* queue manager employs a single queue with first-come-first-served (FCFS) scheduling discipline that drops packets arriving to a full queue. The DT queue capacity is set to *QUEUE_CAPACITY* (2048) full size packets.

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

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

The *ARED* queue manager is a simple extension to RED that adapts *maxp* such that the average queue level tracks a target queue level. We used the recommended target of $(minth + maxth)/2$.

Configuration parameters are based on recommendations from [20]. The *minth* and *maxth* thresholds are the *QUEUE_CAPACITY* divided by 20 and by 2, respectively. The *control_interval* parameter is 0.50 s. The initial setting of *maxp* is 0.10.

The original CoDel description recommends a *target_delay* parameter set to 0.005 s [4]. 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. The original CoDel algorithm proactively drops packets at dequeue time. Due to potential implementation issues of head dropping raised by DOCSIS community, we implemented a tail drop variant of CoDel where packets are proactively dropped at enqueue time. Our implementation is based on an ns2 tail-drop variant of CoDel contributed by CableLabs. PIE is another delay-based AQM and is described in [10]. The D3.1 specification [2] provides recommended algorithm configuration settings for a CM running PIE. We use the default settings described in [10] for the algorithm run on CMTS. We set the *target_delay* parameter to 0.020 s and the T_{update} (frequency of update) parameter to 0.030 s. The *max_burst* parameter is 0.100 s. The internal burst control parameters, α and β, are set to 1.25 and 0.125, respectively.

The results of our simulation studies are presented in the next three sections of the paper. Each section addresses one of the three questions posed in the introduction. First, we evaluate how effectively each AQM maintains throughput fairness and application performance in scenarios that do not involve service rate limits. Next, we analyze the interaction of service rate management with AQM. Finally, we evaluate the

Table 2
Experiment BASE TCP throughput: Mean/Stddev/Sum (Mbps).

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

ability of AQM to protect responsive flows from unresponsive ones.

4. Throughput fairness and application performance

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

4.1. *BASE* simulation results

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

4.1.1. Bandwidth utilization

Table 2 summarizes the throughput achieved by the FTP flows. The total throughput was 37.4 Mbps in all but 8 of the 36 simulations run. Throughput is marginally less for CoDel and PIE when the number of flows is small because the limited amount of buffering can cause the TCP pipeline to stall and, when combined with a small number of flows, loss of throughput results. Aggregate throughput for DT20 never reaches 37.4, demonstrating that the more complex queue management algorithms of CoDel and PIE do provide value. When a sufficient number of flows are active, all queue management mechanisms except DT20 produce the same aggregate bandwidth utilization.

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

Table 3 shows the mean and standard deviation of the loss rate experienced by the FTP flows for each simulation. The lost rates are expressed as a percentage and peak at only 0.245% (approximately 25 packets out of every 10,000 are lost.) We conclude that in all cases TCP is adapting very well to the available bandwidth and buffer space. Not surprisingly, marginal increases in loss rate are again proportional to the number of flows and inversely proportional to queue capacity. When the number of flows exceeds one, DT20 produces significantly higher loss rates than CoDel and PIE. As pointed out in [60], it is possible to tune the parameters of RED (or ARED in our case) to match the behavior of CoDel and PIE, as long as the channel's bit rate stays constant. However, in the presence of adaptive modulation and dynamic channel bonding in networks such as wireless or DOCSIS 3.1, ARED, as currently defined, would not be able to maintain a target queue delay.

Table 4 shows the average RTT experienced by the TCP packets of the FTP flows. The reported RTT is the sum of the inherent path delay and the queuing delay. For the 11 flow case, the mean inherent path delay is 80 ms. Therefore, mean queuing delay can be inferred by subtracting 80 ms from the values shown in the 11 flow column. We observe that PIE, CoDel, and DT20 maintain RTT values consistent with the target of 20 ms. ARED's queue latency ranges from 45 to 60 ms as the number of competing flows increase.

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

Table 3
Experiment BASE TCP loss rate (percentage): Mean/Stddev

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

Table 4
Experiment BASE TCP RTT: Mean/Stddev.

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

Table 5
Experiment BASE 9 FTP flow 5 run average/95% confidence interval.

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

Table 6
Experiment BASE DT 5 run average/95% confidence interval.

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

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

We briefly consider the statistical accuracy of the results associated with the BASE experiment. We ran five independent replications with nine active TCP flows. The 95% confidence intervals for the average throughput, loss rate, and RTT are shown in Table 5. The widths of the confidence intervals were no more than 0.002 Mbps for throughput for all queue management schemes and no more than 0.2 ms in RTT for all three AQMs.

Given the synchronization and lockout issues that are associated with DT, we ran the DT simulation five times with different random seeds and involving different numbers of FTP flows. The results are given in Table 6.

4.1.2. Throughput fairness in the BASE experiment

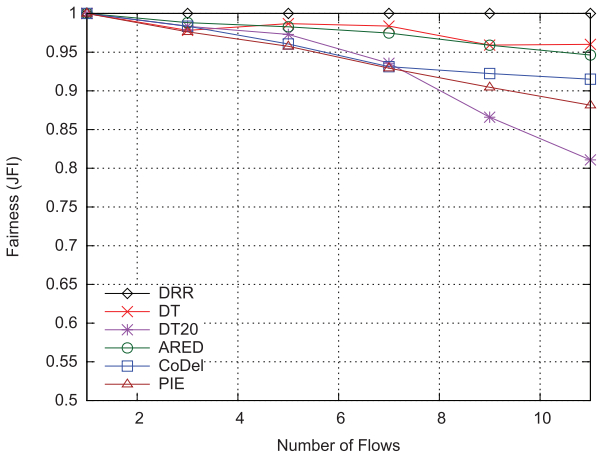
Fig. 2 shows the throughput fairness results. These results are consistent with the standard deviation values previously discussed. The JFI results confirm that DRR is approximately max–min fair. The figure also shows that at high loads DT20 exhibits the highest level of unfair allocation. CoDel and PIE are significantly fairer but still worse than the others. Furthermore, it is apparent that unfairness is increasing with the number of flows for all queue management schemes other than DRR.

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

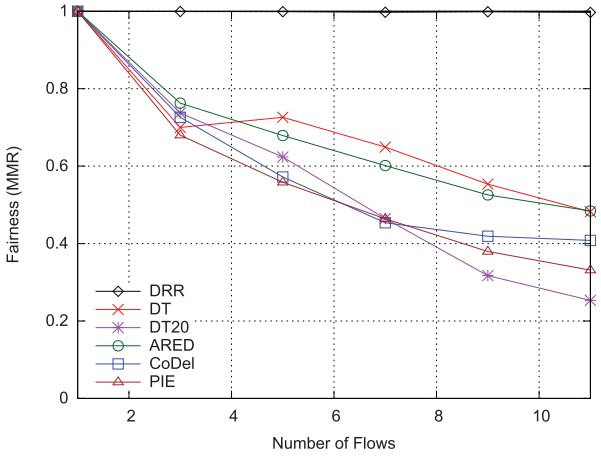
Nevertheless, the underlying cause of the unfairness is not the increase in the number of flows *per se*. Instead, it is primarily a function of the disparity in uncongested RTTs. Table 7 shows the throughput of each of the individual FTP flows for an 11 flow experiment in one **BASE** simulation. The uncongested path RTTs for individual flows are given in the bottom row. It can be seen in this table that the achieved throughput for an individual flow under AQM management is inversely proportional (in a non-linear way) to its path RTT.

This is in agreement with the model proposed by Mathis [61]. TCP/RTT unfairness is in large measure to the fact that achieved throughput depends on factor of 1/RTT. The result is that TCP flows having shorter RTT paths can starve longer path flows of buffer resource at a bottleneck. Each DRR flow has a dedicated amount of buffer resource, and this completely eliminates buffer starvation of one flow by another. The DT flows have a large *QUEUE_CAPACITY* shared buffer resource which mitigates the effect. Because of the parameters chosen, the ARED flows also had more effective buffer capacity (as evidenced by their longer RTTs) than did CoDel and PIE and therefore less unfairness due to buffer contention.

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



(a) Jain's Fairness Index



(b) Min-Max Ratio

Fig. 2. Experiment BASE throughput fairness results.

To further explore the effect of path diversity on fairness, we ran a modified BASE simulation with all FTP flow paths set to the 80 ms mean RTT of the original BASE simulation. The fairness results are given in Tables 8 and 9. Except for DT, the fairness results for all schemes improve across all workloads. In addition, the allocation is not shown to be less fair as the workload increases when RTT does not vary.

Table 8
JFI of BASE with same path RTT.

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

Table 9
MMR of BASE with same path RTT.

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

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

The results as shown in Fig. 3 clearly indicate that the AQMs under investigation become less fair as the range of RTTs widens. We conclude path RTT diversity has an adverse impact on throughput fairness under AQM.

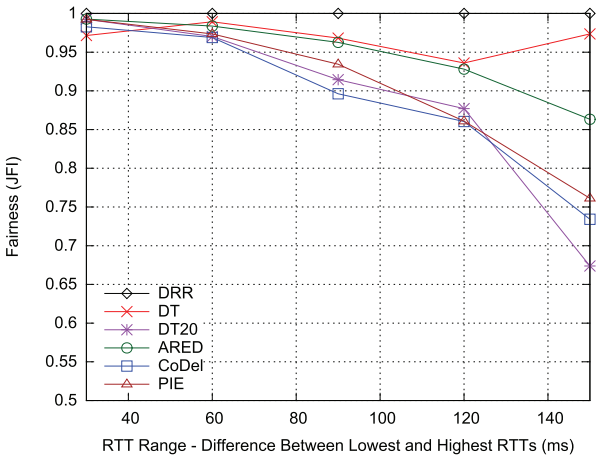
4.1.3. Flow isolation

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

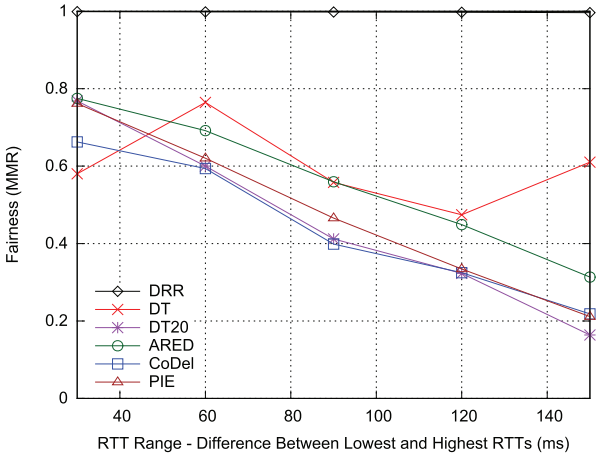
The results are summarized in Table 10. The latency is average one-way latency experienced by the simulated VoIP packets. The mean loss rate experienced by the VoIP flow is of the same order of magnitude as those of the FTP flows which are shown in Table 3. As would be expected, the VoIP loss

Table 7
Throughput of 11 individual FTP flows.

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



(a) Jain's Fairness Index



(b) Min-Max Ratio

Fig. 3. BASE fairness results for seven FTP flows with varied RTT ranges.

rates under AQM are generally larger than the higher bandwidth FTP flows. It is also the case that the disparity between FTP and VoIP loss rates is consistently larger for CoDel than it is for PIE. Furthermore, DT20 yields lower loss rates and higher R-values than either CoDel or PIE across the board. The reasons for these disparities are yet to be fully understood.

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

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

4.2. *HAS* simulation results

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

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

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

During the startup phase, the client makes back-to-back requests until the buffer threshold (2 segments) is reached. The client then enters a steady state where it periodically (every segment time) issues a new segment request. As an example, consider an uncongested scenario involving a single HAS flow and no FTP flows. Once the HAS client reaches steady state, it requests a maximum quality segment (which is 2.5 Mbytes) every 4 s which corresponds to the maximum bit rate of 5.0 Mbps that HAS is configured to consume. Now suppose that the new segment arrives 1.07 s after the request is sent. Then ε is $4/1.07 = 3.74$ and thus the estimated available capacity is 5.0×3.74 or 18.7 Mbps. This capacity estimate underestimates the actual available capacity (38 Mbps) by more than a factor of two. The estimate is low in part because the TCP stack resets the TCP congestion window after the application has been idle for one retransmission timeout period (which happens each on/off cycle). This is a known issue with rate-limited applications [34,44,62].

Table 10
Experiment BASE VoIP isolation performance (mean latency/mean loss rate (percentage)/R-value).

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

Table 11
Experiment HAS throughput efficiency.

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

Table 13
HAS performance metrics (videoPlaybackRate/adaptationRate).

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

Table 11 shows the percentage utilization of the downstream channel for each experiment. The results show that in all cases except for simulations involving five HAS flows and one FTP flow, the channel utilization is at least 94%. CoDel and PIE exhibited a utilization 1–2% lower than that of ARED and DRR.

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

The average throughput for FTP and HAS applications as well as the *applicationBias* is shown in Table 12. For all buffer management schemes, the throughput achieved by both FTP and HAS decreased as the number of competing flows increased.

The *applicationBias* metric is the percentage above or below the max–min fair allocation of the bandwidth that was actually consumed in the experiment (as opposed to the nominal channel bandwidth of 38 Mbps.) The expected max–min fair allocation for an HAS flow is based on a demand of 5 Mbps, which is the highest HAS encoded video bit rate. Using the low congestion scenario (one FTP flow and five HAS flows) as an example, Table 11 indicates that the consumed bandwidth under DRR is 88.3% of the 38 Mbps. The max–min fair allocation of the 33.5 Mbps actually consumed under DRR is 5 Mbps for HAS and 8.55 Mbps for FTP. Consumed bandwidth under CoDel was significantly lower and max–min fair allocation is 5 Mbps for HAS and 6.6 Mbps for FTP. For the CoDel results the actual FTP throughput was 8.17 Mbps and the average HAS throughput was 4.69 Mbps. The HAS *applicationBias* is $(4.69 - 5.00)/5.00$ or -6.3%. For the scenario involving five FTP and five HAS flows, under CoDel, the bandwidth actually consumed is 36.1 Mbps. So the max–min fair allocation is 3.6 Mbps for all flows. In this case, for CoDel, the HAS *applicationBias* is $(3.1 - 3.6)/3.6$ or -13%.

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

Table 13 indicates the HAS performance metrics. The *rebufferTime* results (not shown) was 0 in all simulations. The results show that AQM significantly improves HAS performance compared to DT based on the *videoPlaybackRate* metric. Among the three AQMs, the *videoPlaybackRate* results were very similar. The results suggest that CoDel leads to higher HAS *adaptationRate* than the other AQMs. We have observed that PIE is less tolerant of bursts and consequently during times of heavy congestion does maintain the target queue latency generally more reliably than CoDel. We conjecture that differences in the AQM's burst tolerance behavior contribute to the observed difference with the *adaptationRate* metric. However, it is likely that changing AQM configuration parameters would alter the behavior.

Table 12
FTP and HAS average throughput and allocation bias.

Number of FTPs	DRR	DT	ARED	CoDel	PIE
(a) FTP flow average throughput					
1	8.59 (0.5%)	8.60 (78.2%)	8.59 (2.8%)	8.17 (23.8%)	7.89 (12.4%)
3	6.22 (36.6%)	7.07 (55.3%)	5.90 (29.9%)	5.48 (23.4%)	5.13 (15.4%)
5	4.36 (19.8%)	5.40 (48.1%)	4.32 (18.7%)	4.10 (13.4%)	3.98 (10.2%)
7	3.37 (11.0%)	4.34 (42.9%)	3.46 (14.0%)	3.34 (10.7%)	3.26 (7.8%)
9	2.82 (8.4%)	3.34 (28.4%)	2.88 (10.7%)	2.79 (8.1%)	2.77 (6.8%)
11	2.37 (4.2%)	2.89 (26.9%)	2.46 (8.1%)	2.43 (7.5%)	2.45 (7.6%)
(b) HAS flow average throughput					
1	4.99 (−0.2%)	4.07 (−15.6%)	4.95 (−0.9%)	4.69 (−6.3%)	4.83 (−3.5%)
3	3.55 (−22.0%)	3.04 (−33.2%)	3.73 (−17.9%)	3.82 (−14.0%)	4.03 (−9.3%)
5	2.92 (−19.8%)	1.89 (−48.1%)	2.96 (−18.7%)	3.13 (−13.4%)	3.24 (−10.2%)
7	2.57 (−15.4%)	1.21 (−60.0%)	2.44 (−19.6%)	2.57 (−14.9%)	2.69 (−10.9%)
9	2.21 (−15.0%)	1.27 (−51.2%)	2.10 (−19.3%)	2.21 (−14.6%)	2.28 (−12.3%)
11	2.07 (−9.2%)	0.93 (−59.1%)	1.87 (−17.9%)	1.88 (−16.6%)	1.89 (−16.8%)

Therefore, further study is required before we can draw any conclusions.

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

The work in [34] shows that HAS is prone to a 'spiral down' effect where it is possible for an HAS flow to be 'beaten down' by competing greedy TCP application flows. In this situation, the HAS flow is likely to remain at the lowest bit rate quality level to oscillate across the lowest quality levels. In our results, the *applicationBias* metric shows evidence that aspects of this effect are present. Primarily due to relatively well behaved dynamics, the system does not suffer the full spiral-down effects observed in [34].

4.3. *WEB* simulation results

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

The parameters of the web traffic model, derived from [63], include user and session attributes designed to reproduce the variability inherent in web traffic. The web sessions start randomly within the first second of the simulation. The *inter-page time* is set to a Pareto distribution with mean and shape equal to 5 and 2.0, respectively. The *objects per page* is set to Pareto distribution with mean and shape equal to 4 and 1.2, respectively. The *inter-object time* is set to Pareto distribution with mean and shape equal to 0.5 and 1.5, respectively. The *object size* in KB is set to Pareto distribution with mean and shape equal to 20 and 1.5. Both of these sets of parameters produce distributions with infinite variance. Therefore, it is difficult to obtain repeatable estimates of statistics such as mean response time with simulations of practical duration.

It should be noted that models such as Feldmann's no longer properly characterize the traffic generated by browsing commercial web sites. The objects at such sites typically contain many embedded links to different advertising sites that are not co-located with the primary page. The objects loaded via the embedded links commonly contain a mix of text, images, audio, video, and possibly even additional embedded links. Modeling traffic of this type is well beyond the scope of this paper. Feldmann's model is designed to model browsing of a non-commercial document collection in which the user may download citations, abstracts, or full text of scientific papers of various sizes.

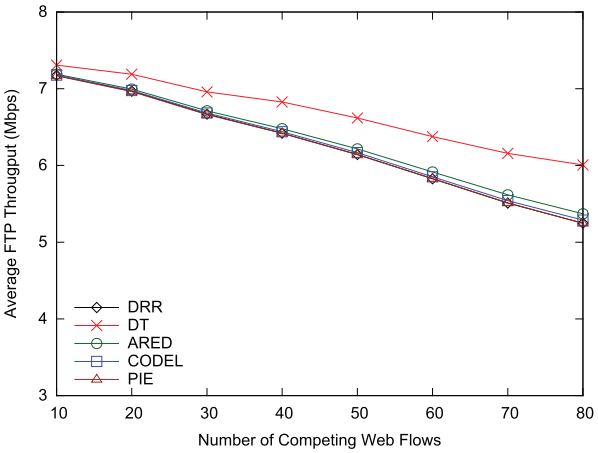


Fig. 4. WEB simulation average FTP throughput.

Therefore, as described in Section 3, the WRT metric reflects the response time experienced by an additional designated web client in requesting and receiving a web object of a particular size. By default, the WRT object size is 20 KB, and the think time before the next request is issued is 10 s. The range of uncongested path RTTs for both the web browsing flows and the competing FTP flows is between 30 and 70 ms. The designated flow for which the WRT metric is computed has an uncongested path RTT of 50 ms which is the mean of the RTTs of the competing traffic.

Fig. 4 shows the average throughput achieved by FTP flows as the number of web browsing flows increases from 10 to 80. Although not shown, the mean bandwidth demand of a single web flow is 0.13 Mbps and the total observed bandwidth consumption of the web flows ranges from 1.3 to 10.3 Mbps for all schedulers except DT. In a max–min fair allocation of the available 38 Mbps, each web flow should receive its entire demand and the five FTP flows should divide the remainder. It can be observed in Fig. 4 that this is approximately the case for all schedulers except DT. For example, when there are 80 web flows, the web flows consume an average of 0.129 Mbps and the five FTP flows consume approximately 5.2 Mbps each. This indicates that the 36.3 Mbps of consumed bandwidth is fairly divided between the web and FTP traffic.

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

Table 14 provides the WRT results for the experiment. As shown, AQM schemes are able to support a range of low bandwidth web flows with consistently low WRT. TCP slow-start requires multiple RTTs before the 20 KBytes of data are successfully received by the WRT client. As the number of web flows increase, there is significant increase in WRT only with the DT queue manager.

In a separate WEB experiment, we explore how the different queue management techniques might impact the WRTs as the WRT object size increases. The scenario involves 5 FTP

Table 14
Web response time/standard deviation (seconds).

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

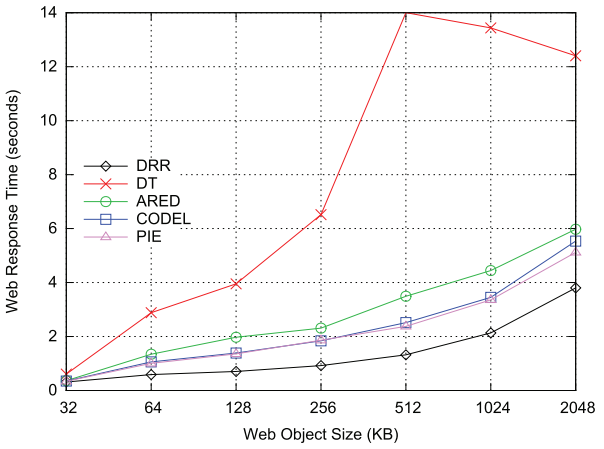


Fig. 5. WEB simulation WRT results for varied object sizes.

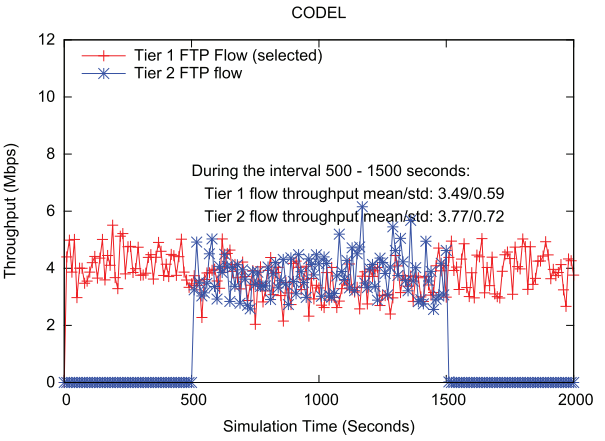


Fig. 6. TIER simulation results with CoDel (1 Tier-2 flow competing with 9 Tier-1 flows). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

flows that compete with 50 web flows based on the web traffic model. Unlike the previous WRT experiment, we vary the WRT web object size from 32 KBytes to 2048 KBytes. The WRT results are shown in Fig. 5. As expected, the WRT increases along with the web object size. The best mean response time was obtained by DRR, and the worst by DT. DT response was also quite sensitive to stochastic effects. In one test, five independent replications of the DT simulation produced mean response times of 6.8, 8.9, 8.2, 6.7, and 6.2 s. In contrast, the mean response times produced by ARED, CoDel, and PIE had small variation.

5. Implications of service rate management

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

Fig. 6 illustrates the observed behavior of nine Tier-1 flows competing with the single Tier-2 flow under CoDel scheduling. The Tier-1 FTP flows run for the entire simulation time of 2000 s, but the Tier-2 flow is active only from time 500 to 1500 s. Each data point plotted represents measured throughput during an interval of 10 s. The Tier-1 flow with 80 ms uncongested RTT is shown in red, and the Tier-2 flow in blue. During the time period when the Tier-2 flow is active, the throughput of the selected Tier-1 flow and Tier-2 flow are 3.49 and 3.77 Mbps respectively, and the Tier-2 flow displays higher variability. The weighted max–min fair allocation is 3.45 Mbps for the Tier-1 flows and 6.9 Mbps for the Tier-2 flow. The unweighted max–min fair allocation is 3.8 Mbps for all flows. These results illustrate that single queue AQM will not support differentiated service based on tiers. Whether or not this outcome is desirable depends on one's perspective. From the perspective of the Tier-2 customer, this is clearly an undesirable outcome and conversely so for the Tier-1 customer.

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

Table 15 summarizes the average allocation achieved by all Tier-1 flows and the allocation achieved by the Tier-2 flow. Simulations involving 1 and 3 competing Tier-1 flows are not shown as these reflect uncongested conditions, and the expected tiered allocations are achieved in all cases.

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

Table 15
TIER flow average throughput allocation (Mbps).

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

Table 16
TIER flow average throughput allocation (Mbps) with 4-channel bonding group.

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

RTTs ranging from 30 to 130 ms while the Tier-2 throughput is that of a single flow.

DOCSIS 3.0 supports channel bonding to provide fat data pipes. A bonding group as deployed in the field often involves 4–8 downstream channels bonded together. Such fat pipes can support higher service rates and tiers. We run **TIER** with a 4-channel bonding group. We set the Tier-1 service rate to 24 Mbps and the Tier-2 service rate to 48 Mbps. To adjust for the increased bandwidth-delay product, we increase the *QUEUE_CAPACITY* and TCP window size by a factor of 4. Again for the uncongested scenarios (i.e., those involving 1 and 3 competing Tier-1 flows), our simulation results show each flow receives a throughput equal to its service rate. For congested scenarios, Table 16 shows the results of this simulation. The results show that our previous conclusions for the single channel case still hold for the channel bonding case.

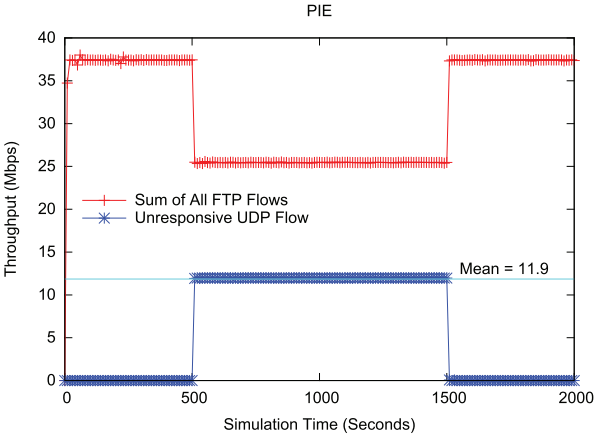


Fig. 7. Experiment UDP results with PIE.

6. Management of unresponsive flows

To evaluate how effectively the AQM schemes under investigation manage unresponsive flows, experiment **UDP** runs a varying number of FTP flows along with an unresponsive UDP CBR flow. Fig. 7 illustrates the results using PIE queue management in a **UDP** simulation in which nine FTP flows compete with an unresponsive UDP CBR flow for a total available bandwidth of 38 Mbps. The UDP flow was configured to send 1500 byte packets at a constant rate of 12 Mbps during the time period 500–1500 s of the 2000 s simulation. No service rate limitations were imposed. As shown, the unresponsive flow was allocated almost its total demand of 12 Mbps by PIE and the FTP flows were allocated the remaining bandwidth. For all other schemes, only DRR correctly allocates equal share (about 3.7 Mbps) to all flows during the time period when the UDP flow was active. The results for DT, ARED, and CoDel were very similar to the PIE result shown in Fig. 7.

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

7. Conclusions

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

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

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

We briefly summarize our conclusions by addressing the three motivating questions raised in the introduction and then provide a statement of next steps.

(1) *How effective are CoDel and PIE in realistic cable network scenarios?*
The schemes are quite effective at maintaining the target queue delay. This in turn provides isolation between responsive flows in the scenarios we explored. Both CoDel and PIE exhibited similar results in all of the experiments, and even as the level of congestion became very high, both were able to provide strong flow isolation.
All schemes except DRR are shown to not allocate bandwidth fairly in the presence of heterogeneous RTTs. However, the principal cause of the unfairness is TCP's well known bias toward short-RTT connections. Nevertheless, delay-based AQMs such as CoDel and PIE are particularly sensitive to the RTT induced unfairness issue when compared to other queue management schemes that allow sustained large queues. When a buffer bloated queuing point causes 100s of milliseconds of packet delay, the fairness outcome of competing TCP flows that might have uncongested path delays of a fraction of this are less sensitive to TCP RTT-based unfairness. When delay-based AQM removes 'bufferbloated' queues, differences in uncongested path delays have more of an impact on TCP performance. Nevertheless, it should be noted that throughput fairness is not a primary objective of delay-based AQMs.

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

(3) *How effectively do the schemes isolate responsive traffic from unresponsive flows?*
An inherent problem with the single queue schemes is that they cannot effectively manage unresponsive flows. This is a huge challenge network operators have to address. One of our next steps includes addressing this problem with a multiple queue solution that can also meet the constraints of CMTS vendors.

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

Since our experiments made no use of new DOCSIS features such as dynamic channel bonding, we believe that our results are applicable to any system in which a head-end device or router manages a single shared medium downstream channel. In practical terms, the only networks of this type in widespread use at present are cable networks and some wireless networks such as WiMax and WiFi (operating in infrastructure mode).

References

[1] Cable Television Laboratories, Inc., Data-over-cable service interface specifications: DOCSIS 3.0 MAC and Upper Layer Protocols Interface specification, 2014, CM-SP-MULPIv3.0-I24-140403 http://www.cablelabs.com.
[2] Cable Television Laboratories, Inc., Data-over-cable service interface specifications: DOCSIS 3.1 MAC and Upper Layer Protocols Interface Specification, 2014, CM-SP-MULPIv3.1-I02-140320 URL http://www.cablelabs.com.
[3] J. Gettys, K. Nichols, Bufferbloat: Dark buffers in the Internet, Commun. ACM 55 (1) (2012) 57–65.
[4] K. Nichols, V. Jacobson, Controlling queue delay, Commun. ACM 55 (7) (2012) 42–50.
[5] S. Floyd, V. Jacobson, Traffic phase effects in packet-switched gateways, ACM SIGCOMM Comput. Commun. Rev. 21 (2) (1991) 26–42.
[6] 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.
[7] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. 1 (4) (1993) 397–413.
[8] V. Rosolen, O. Bonaventure, G. Leduc, A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic, ACM SIGCOMM Comput. Commun. Rev. 29 (3) (1999) 23–43.
[9] 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.
[10] 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: 2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR), IEEE, 2013, pp. 148–155.
[11] R. Pan, P. Natarajan, C. Piglione, M. Prabhu, V. Subramanian, F. Baker, B.V. Steeg, PIE: A lightweight control scheme to address the bufferbloat problem (draft-pan-aqm-pie-00), 2013, https://tools.ietf.org/html/draft-pan-aqm-pie-02.

[12] R. Pan, L. Breslau, B. Prabhakar, S. Shenker, Approximate fairness through differential dropping, ACM SIGCOMM Comput. Commun. Rev. 33 (2) (2003) 23–39.

[13] J. Martin, J. Westall, A simulation model of the DOCSIS protocol, Simulation 83 (2) (2007) 139–155.

[14] J. Martin, J. Westall, T. Shaw, G. White, R. Woundy, J. Finkelstein, G. Hart, Cable modem buffer management in DOCSIS networks, in: 2010 IEEE Sarnoff Symposium, IEEE, 2010, pp. 1–6.

[15] G. Hong, J. Martin, S. Moser, J. Westall, Fair scheduling on parallel bonded channels with intersecting bonding groups, in: 2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, 2012, pp. 89–98.

[16] J. Martin, G. Hong, J. Westall, Managing fairness and application performance with active queue management in DOCSIS-based cable networks, in: Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing, ACM, 2014, pp. 9–14.

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

[18] M. May, J. Bolot, C. Diot, B. Lyles, Reasons not to deploy RED, in: 1999 Seventh International Workshop on Quality of Service, 1999 (IWQoS'99), IEEE, 1999, pp. 260–262.

[19] D. Lin, R. Morris, Dynamics of random early detection, in: ACM SIGCOMM Computer Communication Review, 27, ACM, 1997, pp. 127–137.

[20] S. Floyd, R. Gummadi, S. Shenker, et al., Adaptive RED: An algorithm for increasing the robustness of RED's active queue management, Tech. rep. (2001). http://www.icir.org/floyd/papers/adaptiveRed.pdf.

[21] R. Adams, Active queue management: A survey, IEEE Commun. Surveys Tutor. 15 (3) (2013) 1425–1476.

[22] 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.

[23] 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: 2013 22nd International Conference on Computer Communications and Networks (ICCCN), IEEE, 2013, pp. 1–7.

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

[25] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, E. Dumazet, Flowqueue-Codel (draft-hoeiland-joergensen-aqm-fq-codel-00), 2014, https://tools.ietf.org/html/draft-hoeiland-joergensen-aqm-fq-codel-00.

[26] P.E. McKenney, Stochastic fairness queueing, in: Proceedings of Ninth Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM'90). 'The Multiple Facets of Integration', IEEE, 1990, pp. 733–740.

[27] N. Khademi, D. Ros, M. Welzl, The new AQM kids on the block: Much ado about nothing? Technical Report, University of Oslo, 2013. https://www.duo.uio.no/bitstream/handle/10852/37381/khademi-AQM_Kids_TR434.pdf?

[28] G. White, D. Rice, Active queue management algorithms for DOCSIS 3.0, 2013, CableLabs Technical Report, April 2013, http://www.cablelabs.com/downloads/pubs/Active_Queue_Management_Algorithms_DOCSIS_3_0.pdf.

[29] I.S. Organization, IEEE Standard for WirelessMAN-Advanced Air Interface for Broadband Wireless Access Systems, IEEE Std 802.16.1-2012 (2012) 1–1090, doi:10.1109/IEEESTD.2012.6297413.

[30] S. Bauer, D. Clark, W. Lehr, Powerboost, in: Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks, ACM, 2011, pp. 7–12.

[31] Sandvine Incorporated ULC, Global Internet phenomena report, 2015, https://www.sandvine.com/.

[32] S. Akhshabi, A.C. Begen, C. Dovrolis, An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP, in: Proceedings of the Second Annual ACM Conference on Multimedia Systems, ACM, 2011, pp. 157–168.

[33] S. Akhshabi, L. Anantakrishnan, A.C. Begen, C. Dovrolis, What happens when HTTP adaptive streaming players compete for bandwidth? in: Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video, ACM, 2012, pp. 9–14.

[34] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, R. Johari, Confused, timid, and unstable: Picking a video streaming rate is hard, in: Proceedings of the 2012 ACM Conference on Internet Measurement Conference, ACM, 2012, pp. 225–238.

[35] J. Jiang, V. Sekar, H. Zhang, Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ACM, 2012, pp. 97–108.

[36] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, M. Watson, A buffer-based approach to rate adaptation: Evidence from a large video streaming service, in: Proceedings of the 2014 ACM Conference on SIGCOMM, ACM, 2014, pp. 187–198.

[37] L. De Cicco, S. Mascolo, An adaptive video streaming control system: Modeling, validation, and performance evaluation, IEEE/ACM Trans. Netw. 22 (2) (2014) 526–539.

[38] J. Martin, Y. Fu, N. Wourms, T. Shaw, Characterizing Netflix bandwidth consumption, in: 2013 IEEE Consumer Communications and Networking Conference (CCNC), IEEE, 2013, pp. 230–235.

[39] Z. Li, A.C. Begen, J. Gahm, Y. Shan, B. Osler, D. Oran, Streaming video over HTTP with consistent quality, in: Proceedings of the 5th ACM Multimedia Systems Conference, ACM, 2014, pp. 248–258.

[40] C. Liu, I. Bouazizi, M. Gabbouj, Rate adaptation for adaptive HTTP streaming, in: Proceedings of the Second Annual ACM Conference on Multimedia Systems, ACM, 2011, pp. 169–174.

[41] G. Tian, Y. Liu, Towards agile and smooth video adaptation in dynamic HTTP streaming, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ACM, 2012, pp. 109–120.

[42] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. Begen, D. Oran, Probe and adapt: Rate adaptation for HTTP video streaming at scale, IEEE J. Select. Areas Commun. 32 (4) (2014) 719–733.

[43] L. De Cicco, V. Caldaralo, V. Palmisano, S. Mascolo, Elastic: A client-side controller for dynamic adaptive streaming over HTTP (DASH), in: 2013 20th International Packet Video Workshop (PV), IEEE, 2013, pp. 1–8.

[44] S. Nazir, Z. Hossain, R. Secchi, M. Broadbent, A. Petlund, G. Fairhurst, Performance evaluation of congestion window validation for DASH transport, in: Proceedings of Network and Operating System Support on Digital Audio and Video Workshop, ACM, 2014, p. 67.

[45] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, H. Zhang, Understanding the impact of video quality on user engagement, ACM SIGCOMM Comput. Commun. Rev. 41 (4) (2011) 362–373.

[46] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, H. Zhang, Understanding the impact of video quality on user engagement, Commun. ACM 56 (3) (2013) 91–99.

[47] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, H. Zhang, A quest for an Internet video quality-of-experience metric, in: Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ACM, 2012, pp. 97–102.

[48] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, P. Halvorsen, Flicker effects in adaptive video streaming to handheld devices, in: Proceedings of the 19th ACM International Conference on Multimedia, ACM, 2011, pp. 463–472.

[49] S.S. Krishnan, R.K. Sitaraman, Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs, IEEE/ACM Trans. Netw. 21 (6) (2013) 2001–2014.

[50] N. Cranley, P. Perry, L. Murphy, User perception of adapting video quality, Int. J. Human-Comput. Stud. 64 (8) (2006) 637–647.

[51] C. Liu, I. Bouazizi, M.M. Hannuksela, M. Gabbouj, Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network, Signal Process.: Image Commun. 27 (4) (2012) 288–311.

[52] R.G. Cole, J.H. Rosenbluth, Voice over IP performance monitoring, ACM SIGCOMM Comput. Commun. Rev. 31 (2) (2001) 9–24.

[53] P. Barford, M. Crovella, Generating representative web workloads for network and server performance evaluation, in: ACM SIGMETRICS Performance Evaluation Review, vol. 26, ACM, 1998, pp. 151–160.

[54] M.E. Crovella, A. Bestavros, Self-similarity in world wide web traffic: Evidence and possible causes, IEEE/ACM Trans. Netw. 5 (6) (1997) 835–846.

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

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

[57] A.K.J. Parekh, A generalized processor sharing approach to flow control in integrated services networks, Massachusetts Institute of Technology, 1992 (Ph.D. thesis).

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

[59] M. Shreedhar, G. Varghese, Efficient fair queueing using deficit round robin, in: ACM SIGCOMM Computer Communication Review, 25, ACM, 1995, pp. 231–242.

[60] N. Kuhn, E. Lochin, O. Mehani, Revisiting old friends: Is CoDel really achieving what RED cannot? in: Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop (CSWS'14), ACM, New York, NY, USA, 2014, pp. 3–8.

[61] M. Mathis, J. Semke, J. Mahdavi, T. Ott, The macroscopic behavior of the TCP congestion avoidance algorithm, ACM SIGCOMM Comput. Commun. Rev. 27 (3) (1997) 67–82.
[62] G. Fairhurst, A. Sathiaseelan, R. Secchi, Updating tcp to support rate-limited traffic (draft-ietf-tcpm-newcwv-12), 2015, https://tools.ietf.org/html/draft-ietf-tcpm-newcwv-12.
[63] A. Feldmann, A.C. Gilbert, P. Huang, W. Willinger, Dynamics of IP traffic: A study of the role of variability and the impact of control, in: ACM SIGCOMM Computer Communication Review, 29, ACM, 1999, pp. 301–313.

Gongbing Hong is currently pursuing the Ph.D. degree in the School of Computing, Clemson University. His past work experience included more than a decade engineering career in the cable system division at Cisco/Scientific-Atlanta. His current research focus is on bandwidth management for emerging DOCSIS system.

James Martin received the Ph.D. degree from North Carolina State University. Currently, heis 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 DOCSIS 3.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.