Active Sense Queue Management (ASQM)

Daniel M. Havey
Department of Computer Science
University of California Santa Barbara
dhavey@yahoo.com

Kevin C. Almeroth
Department of Computer Science
University of California Santa Barbara
almeroth@cs.ucsb.edu

***Abstract*—Active Queue Management (AQM) algorithms have seen a lot of attention in recent academic literature as well as in the popular press. The problem with traditional AQM is that it is designed to operate on queues at the IP layer. However, the problem popularly called bufferbloat can move about among many queues some of which are resistant to traditional AQM such as Layer 2 MAC protocols used in cable/DSL links. We call this problem bufferbloat displacement. Our contribution is a new class of AQM algorithm called Active Sense Queue Management (ASQM). ASQM is an IP layer AQM protocol that has the added benefit of managing Layer 2 link delay, i.e. bufferbloat displacement. We provide testbed experiments comparing ASQM's bandwidth delay characteristics with traditional AQM algorithms demonstrating ASQM's ability to manage bufferbloat displacement where traditional AQM algorithms cannot.**

I. Introduction

The queue management problem commonly called bufferbloat has been with us for a long time. However, recent increases in the range of bandwidth capabilities found in network devices have made the problem more exigent. A typical cable modem in use today provides bandwidth from 2-50 Mbps. Statically engineering queue size for a device such as this is essentially impossible because the bandwidth and delay characteristics vary too much. A queue engineered for a delay of 100 ms at 50 Mbps would be 25 times too large at 2 Mbps producing 2.5 seconds worth of delay. Faster speeds are leading to larger queue sizes making the problem worse for lesser speeds.

There are two dynamic AQM algorithms deployed today to manage queue size and control bufferbloat: CoDel and PIE, [25], [27]. The weakness of these algorithms is that they do not control the bufferbloat problem in queues below the IP layer. We call this problem bufferbloat displacement. Queuing theory tells us that bufferbloat is displaced into slowest queue(s) in the path. When the ISP access link cannot meet the IP layer ISP committed rate then the bufferbloat is displaced into the access link filling the MAC layer queues and causing bufferbloat that CoDel and PIE cannot detect.

The Federal Communications Commission (FCC) produces a report every year called Measuring Broadband America.[1] In this report from the years 2010 through 2014 the 80/80 study designed by the North Carolina State University Institute for Advanced Analytics determines how often the access link does not meet the ISP committed rate at the IP layer in the modem during peak traffic hours (from 7:00 pm to 11:00 pm). In 2014 50% of the 16 ISPs measured provided 90% of the committed rate to 80% of the panelists 80% of the time. The other 50% of ISPs provided less than 90% of the committed rate.

A significant amount of bufferbloat occurs during peak traffic hours that CoDel and PIE cannot detect because the access link is providing less than the ISP committed rate. So we develop ASQM to solve this problem. ASQM uses an active sensing mechanism to detect bufferbloat across the access link. Our ASQM algorithm runs at the IP layer, but, senses queuing delay across the entire access link. This novel approach allows ASQM to remove bufferbloat from the access link 100% of the time even during peak traffic hours when bufferbloat displacement occurs that CoDel and PIE cannot detect.

ASQM uses an active sensing approach. Active sensing creates a trade-off between overhead (from the sensory packets), measurement accuracy and feasibility. Our contribution in this work is to investigate these trade-offs and provide the most feasibility and accuracy with the least amount of overhead. Our ASQM algorithm controls bufferbloat on par with CoDel and PIE during non-peak traffic hours and continues to provide excellent bufferbloat protection during peak traffic hours when CoDel and PIE cannot. Our ASQM algorithm can be deployed on any ISP modem or router and requires less than 0.5% overhead.

The remainder of this paper is organized as follows: In Section II we describe related solutions and supporting work. In Section III we describe the primary bottleneck between the modem and the CMTS/DSLAM. In Section IV we describe our ASQM algorithm and its interaction with queues in the primary bottleneck. In Section V we describe our evaluation methodology and testbed. In Section VI, we present our results and in Section VII we summarize, conclude and discuss future work.

[1] http://data.fcc.gov/download/measuring-broadband-america/2014/2014-Fixed-Measuring-Broadband-America-Report.pdf

ISBN 978-3-901882-68-5

II. Related Work

The use of advanced DiffServ (Differentiated Services) scheduling mechanisms such as fair, classful or priority queuing can be effective in helping the network serve the needs of application classes [4], [30], [12], [22], [3], [31]. DiffServ packet scheduling does not replace AQM. DiffServ decides which packet to send next. AQM decides queue length. They are complimentary algorithms. DiffServ should be used in conjunction with an AQM algorithm.

Explicit Congestion Notification (ECN) enhances the performance of AQM by marking packets rather than dropping them. The TCP server (sender) reacts to marked packets as if they were dropped packets [14], [28]. The state of deployment and activation of ECN in the Internet is poor but improving, about 40 percent at the core as of this writing [2], [21]. Some middleboxes (proxies) still do not duplicate these options properly, [18]. Because of this an AQM algorithm must be able to drop packets. However, if ECN is deployed then the AQM should use ECN marking instead of dropping.

Loss based TCP congestion control algorithms come in many flavors with varying levels of aggression [7], [29], [24], [6], [23], [13]. Algorithms commonly deployed and active on the Internet today include Cubic (Linux), Compound (Windows XP), and NewReno (mac, Windows Vista+), [8], [26], [33]. Loss based TCP protocols seek to fill the slowest queue as much as possible and increase bufferbloat. The ubiquitously deployed TCP Cubic and NewReno are loss based. TCP Compound is a hybrid of loss and delay based TCP.

Delay based TCP protocols are sensitive to RTT and eliminate bufferbloat except when in the presence of a loss based TCP such as Cubic or NewReno. This makes delay based TCP protocols ineffective against bufferbloat in general because they are likely to face one of the ubiquitously deployed loss based NewReno or Cubic TCPs. However, in a controlled environment delay based TCP can reduce bufferbloat. Examples of delay based TCP are Vegas, CAIA CDG, and Fast, [17], [34], [5]. Many other TCP variants have been created over the years including equation based TCP [15], Multipath TCP [16], split TCP [19], and Network Coding TCP (NCTP) [32]. However, no TCP variant exists today that can control bufferbloat in the presence of the ubiquitously deployed loss based Cubic and NewReno TCPs.

AQM algorithms are designed to manage bufferbloat. The IETF currently recommends that an AQM algorithm should be implemented in network devices in compliment with a DiffServ scheduler.[2] Recent and popular AQM algorithms such as CoDel and PIE use burst size (queue delay over time) to control the queue, [25], [27]. Older algorithms such as RED and its variants measure queue size directly, [11], [10], [20], [9].

[2]http://tools.ietf.org/html/draft-ietf-aqm-recommendation-03

The problem with traditional AQM is that it does not protect Layer 2 MAC protocols from bufferbloat. These protocols are resistant to traditional end to end AQM because they are Link Layer protocols and do not communicate with the TCP sender. In ASQM we take a different approach using active sensing at the IP Layer to determine the queuing delay across the Link Layer and send a mark/drop signal to the end to end TCP sender to slow down.

III. Managing the Right Queue

The next step is to examine the queuing theory that AQM algorithms are built from. This queuing theory includes edge router queuing equations, typical network neighborhood topology and the primary bottleneck at the access link.

The queue size in backbone routers is governed by the arrival/departure processes as shown in Appenzeller et al. [1]. However, in edge routers queue size is governed by the $Q = BD$ equation as demonstrated in Villamizar et al. [35]. ASQM is intended for edge routers so we expand the Villamizar bandwidth delay equation to the following:

$$Q_{size} = B * D \rightarrow D = optimal; B = optimal \quad (1)$$

$$Q_{size} < B * D \rightarrow D = optimal; B < optimal \quad (2)$$

$$Q_{size} > B * D \rightarrow D > optimal; B = optimal \quad (3)$$

Bandwidth B is the emission rate from the queue, and delay $D \approx RTT$ where RTT is the Round Trip Time for the path. If the queue size is too small as in Equation 2 then the flow will lose bandwidth. Otherwise if the queue size is too large as in Equation 3 then the flow will have excessive delay. Only when the queue size is just right as in Equation 1 will the flow have the maximum bandwidth at the minimum RTT.

These equations demonstrate why adding more bandwidth is no more than a temporary solution at best. A flow experiencing bufferbloat is acting according to Equation 3. Increasing bandwidth will bring the flow towards Equation 1. Increasing bandwidth further will drive the flow into Equation 2 causing bandwidth loss. The proper way to solve the problem is to size the edge router queue according to Equation 1. However, because the bandwidth and RTT are unknown in advance and vary over time we must use dynamic AQM algorithms in order to control bufferbloat.

In this paper we refer to typical consumer access link using 802.11 ac wireless, Gigabit Ethernet and 2-50 Mbps cable technology. We know that faster link technology exists including Gigabit fiber. This is an example of throwing more bandwidth at the problem to temporarily fix it. Assuredly many consumers who purchase Gigabit fiber will also find the wherewithal to switch to faster access technologies such as 10 or even 100 Gigabit Ethernet and 60 GHz WiFi will certainly have more throughput than the 802.11 line.

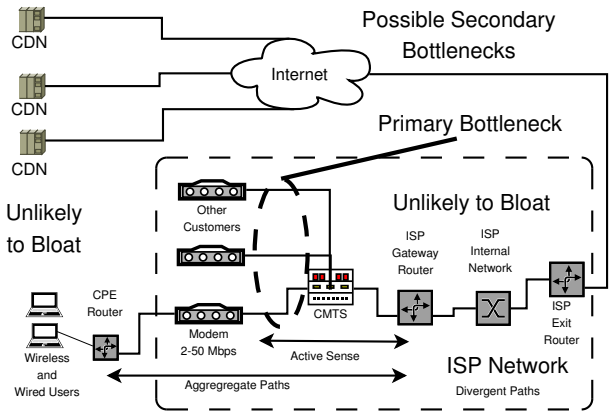

Fig. 1. Typical Consumer Network Neighborhood

It has been widely reported in the press that business disputes between providers have been resulting in congestion at interconnection points and network paths. These secondary bottlenecks shown in Figure 1 can cause performance degradation for users. We know that secondary bottlenecks can cause variable degradation in performance from user to user. However this problem is out of scope for this paper because there is nothing that an AQM algorithm can do about a secondary bottleneck caused by a business dispute. ASQM focuses on bufferbloat that occurs in the primary bottleneck not on performance degradation caused by business disputes.

Because the access link is the primary bottleneck AQM algorithms are normally deployed at IP layer in the CPE modem. This configuration is fine when the ISP access link provides 100% or more of the committed rate given in the Service Level Agreement (SLA). However, the ISP access link commonly delivers less than 100% of the committed rate during peak traffic hours. This is because providing 100% of the committed rate during peak traffic hours would cause a terrible waste of ISP capacity during non-peak traffic hours. When the access link rate is less than the committed rate bufferbloat will occur in layers below the IP where CoDel and PIE cannot detect it.

We believe that a different approach is required. It is unreasonable to expect all ISPs to maintain 100 percent of the advertised bandwidth 100 percent of the time. ISPs are faced with a wide variety of conditions under which they must operate. Distance, competing traffic, and RF conditions on the wire (especially for twisted pair) affect their networks and 90 percent of advertised rates during peak periods is quite reasonable performance. We propose ASQM as an answer to the problem. ASQM uses active sensing between the IP layers to detect bufferbloat and manage all of the queues between them regardless of where in the link the bufferbloat is occuring.

ASQM's active sense mechanism is shown in Figure 1 in two equivalent configurations. One is operating from the IP layer on the modem to the ISP gateway router and the other is operating from the IP layer on the Customer Provided Equipment (CPE) router. These configurations are equivalent because the link between the CPE router and the modem is 10/100/1000 Mbps Ethernet. Both configurations protect the primary bottleneck including the Cable MAC layers of the modem and the CMTS. The rest of the ISP network from the gateway router to the exit router(s) is also unlikely to bloat because these links are traffic engineered and maintained.

IV. Active Sense Queue Management

Our ASQM algorithm uses active sense packets to determine the queuing delay in the access link. Successfully employing an active sensory system requires a detailed understanding of the queuing arrangements in the layers below the IP as well as the DiffServ packet scheduling applied by ISPs. In addition active systems create overhead so careful attention must be taken in order to reduce overhead.

Figure 2 shows a detail of the CMTS and modem stacks. DSL technology uses a similarly arranged stack. Packets flowing through the system in either direction (upload/download) encounter a hierarchical token bucket rate limiter used by the ISP located at IP in Layer 3. This is the ISP rate limiter where AQM will be deployed.

As demonstrated in Nichols and Jacobson any queue in the lower layers that is slower than the advertised (token bucket) rate will defeat the DiffServ/AQM system at the IP layer [25]. The FCC study Measuring Broadband America indicates that this happens frequently especially during peak traffic periods from 7:00 pm to 11:00 pm. This portion of the link is called out in Figure 2 by a dashed rectangle. We did not include the 802.2 LLC since these layers are extremely fast and unlikely to suffer from bufferbloat. In any case some form of traditional AQM is still possible at the 802.2 LLC layer.

At the cable MAC, AQM activities become impossible without a significant redesign of the protocol and a reassignment of layering responsibilities. AQM protocols must be able to drop packets. Dropping frames is in direct conflict with the MAC's retransmission scheme. ECN marking is preferable to dropping, however, in cases where ECN is not implemented in every device in the path dropping is required. Even in cases where ECN is implemented in every device, it is impossible to guarantee that the proper bits exist in a MAC frame to signal an ECN mark because fragmentation could have already occurred.

At layers below the cable MAC, dropping or otherwise signaling the sending end to slow down is even more improbably difficult. The Discrete Space Time Coder (DSTC) converts MAC frames into QAM signals for transmission by the Physical Media Dependent layer across the wire. At this point concepts such as packets and end to end signaling no longer exist and traditional AQM activities are impossible. ASQM is our answer to these problems.

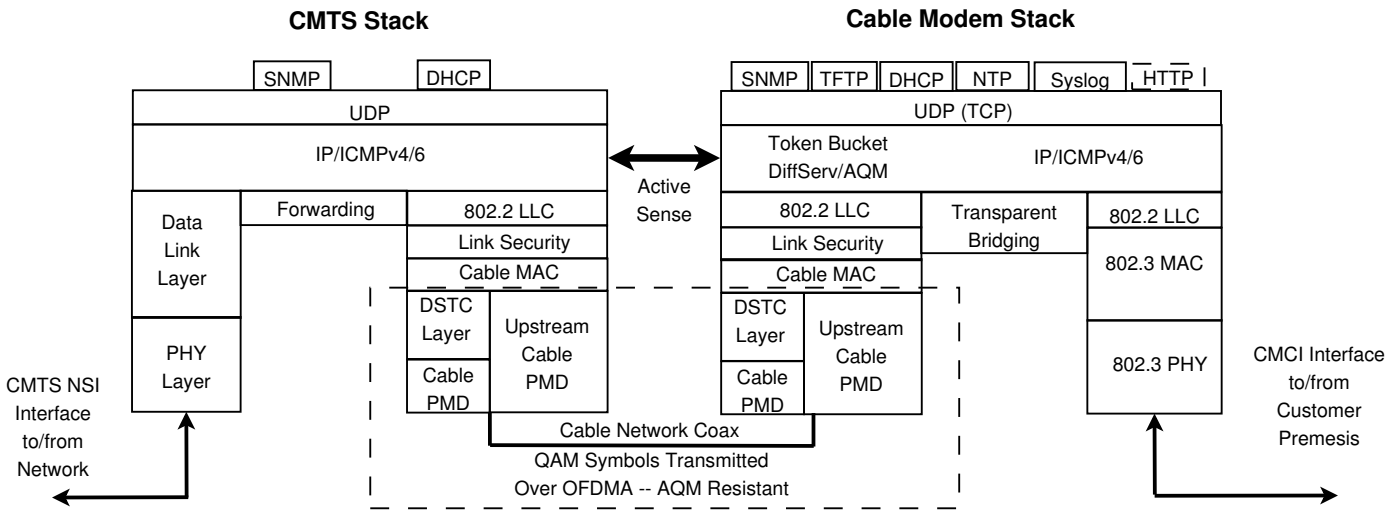

Fig. 2. Primary Bottleneck – Cable Modem Termination System to Cable Modem Link (DOCSIS 3.1)

ASQM is an IP layer protocol that uses active sensing to determine delay occurring in all of the queues in the lower layers as shown in Figure 2. The active sensing mechanism consists of packets sent from IP Layer to IP Layer.

ASQM's sensing packets have been the subject of much discussion and research in our laboratory. The goal is to measure the RTT of the access link. We have tried ICMP packets, SNMP packets and injected packets as well as periodic sensory packets and sensory packets that are proportionate to the flow. Each method offers trade-offs between overhead, sensing accuracy, scalability and feasibility. For the sake of brevity we only discuss sensory packet injection in proportion to the flow in detail.

Our ASQM algorithm generates sensory packets by copying a packet header from the flow, manipulating the IP address fields, marking, timestamping and injecting them into the stack. These sensory packets flow from the IP layer at the modem (where they are created) to the IP layer at the ISP router and back. Upon receiving a returning sensory packet our ASQM algorithm generates link RTT samples by subtracting the timestamp from the current time. Then our ASQM algorithm generates a link RTT estimate by smoothing the estimates according to the calculation detailed in RFC 6928.[3]

Many (if not all) ISPs employ DiffServ packet scheduling (not a substitue for AQM) in order to separate voip and other high priority real time traffic from best effort traffic. The high priority traffic is forwarded into the Expedited Forwarding (EF) queue and the best effort traffic goes into the Assured Forwarding queue (AF). The EF queue serves low bandwidth flows such as voip and needs no AQM. ASQM operates only on the AF queue for best effort flows. This assures that sensory packets transmitted across the access link always go through the correct queuing.

The next step is to find ways to reduce overhead. There are two ways of doing this; reduce the size of the sensory packet and reducing the ratio of sensory packets relative to data packets. ASQM uses a 20 byte IP packet header with an 8 byte timestamp. This is the smallest packet we could design and transmitted at a ratio of one sensory packet for every four data packets this translates into an overhead of less than 0.5% when using 1500 byte packets and even less when using larger MTUs. These packets are forwarded using an *iptables* rule.[4] Smoothing is accomplished using the TCP RTT calculation.

In our design of ASQM's sensory packets we have taken a great deal of care to reduce overhead and to ensure that the packets flow through the correct queue at the ISP. In any case we expect the ISP to cooperate or at least not actively seek to confuse the sensory packets. With this cooperation in mind we can take a very accurate measurement of queuing delay encountered in the access link. The next step is to design our ASQM algorithm to take corrective action (marking/dropping) when bufferbloat is detected in the access link.

When the smoothed link estimate reaches a threshold value that we call *target* (default 100 ms) then our ASQM algorithm takes corrective action. The 100 ms target queue size is a default tunable parameter. The default target queue size value was chosen based on our own experiments and on the default value from Nichols and Jacobson [25]. This default value has been shown to give good performance for a wide range of RTTs from 10 ms to 500 ms. ASQM's dropping/marking activities are governed by a control law that produces an approximately linear slowdown from the sender. Starting with an interval of 100 ms (default) each drop interval is reduced by $1/\sqrt{n}$ where n is the number of marks/drops since the beginning of marking/dropping activity. When an active sense measurement less than 100 ms is received marking/dropping

[3]https://tools.ietf.org/html/rfc6298

[4]http://linux.die.net/man/8/iptables

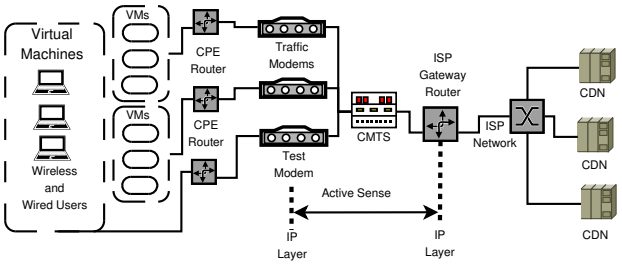

Fig. 3. Hardware emulation testbed

activity ceases and n is reset to zero.

ASQM's active sensing mechanism measures round trip queuing delay in the access link and the algorithm takes corrective action (marking/dropping) when the queuing delay exceeds the target threshold. ASQM is designed to be employed in both the upload and download direction on the best effort (AF) DiffServ queue. Using this novel mechanism ASQM is able to remove bufferbloat from the access link even during peak traffic hours when other algorithms are unable to detect the bufferbloat.

V. Evaluation methodology and testbed

The evaluation we undertook tried to illuminate ASQMs queue management capabilities by comparing ASQM to popular AQM algorithms such as CoDel and PIE [25], [27]. The goal of this evaluation was to demonstrate the bandwidth delay tradeoffs of each algorithm both when the Layer 2 link was providing 100% of the rated speed and when the link is only providing 90% because of traffic. All three algorithms performed well when the Layer 2 link was providing 100% of the advertised speed. However, ASQM continued to perform well even when the Layer 2 link was only providing 90% of the advertised speed as is common during peak usage hours.

The testbed we constructed is shown in Figure 3. The traffic was generated by virtual user machines and CDNs constructed from PCs running the Linux 3.2 kernel. The traffic was generated as specified in the IETF draft AQM Evaluation Guidelines; five repeating TCP transfers of 5MB each, one continuous TCP transfer and four HTTP web traffic (repeated downloads of 700kB).[5] This traffic configuration is specifically designed to investigate bufferbloat particularly with a mix of short flows in combination with long flows.

Experiments were performed in the upload (data flowing from the users to the CDNs) as well as in the download direction. Each experiment was run for 120 seconds with 5 experimental runs then compiled into bandwidth and delay CDFs. Bandwidth and delay CDFs were provided for all three algorithms in each link speed configuration (100% and 90% of the committed speed). The experiments cover a range of RTTs from 50 ms to 200 ms because this range represents in large part the conditions that will be found in end user access links. In any case all three of the algorithms (CoDel, PIE and our ASQM) begin to break down above 250 ms and become unusable by 500 ms.

The CPE routers, the modems, the CMTS and the ISP gateway were constructed from PCs running the Linux 3.15 kernel. The modems and CMTS were equipped with ASQM, CoDel and PIE, [25], [27]. This was done using a Hierarchical Token Bucket (HTB) as is common in routers from Cisco and other manufacturers.[6] We note that the HTB is a packet scheduler and does not manage queue size.

VI. Evaluation

The goal of this evaluation was first to demonstrate that ASQM performs on par with CoDel and PIE during non-peak usage hours. All AQM algorithms are expected to perform well when the link is providing 100% or more of the rated bandwidth. Secondly we wanted the evaluation to show that ASQM is the only algorithm that continues to perform well when the link is providing less than 100% of the rated bandwidth as is common during peak usage hours.

We have performed thousands of experiments with a large range of network factors and parameters; RTT 10-1000 ms, bandwidth 1-50 Mbps, target queue size 10-500 ms in the upload and download directions. In order to demonstrate that ASQM achieves both goals we have chosen to present two sets of graphs for each algorithm. We have chosen to present two sets of CDFs for each algorithm demonstrating AQM behavior during non-peak hours (100% committed rate or more) and during peak hours (less than 100% committed rate)

The modems in both sets of experiments were configured to provide 8 Mbps committed rate and 16 Mbps peak rate (using HTB borrowing). For the non-peak traffic hours experiments we set the link up to provide the peak rate for each modem simultaneously (48 Mbps). With this configuration we expected to see about 15.5 Mbps from each modem. For the peak traffic experiments we set the link up to provide about 90% of the committed rate for each of the three modems (21 Mbps). With this configuration we expected to see about 6.5 Mbps from each modem.

The CDF's presented examine the bandwidth and delay characteristics of each algorithm in each traffic condition. We found that all algorithms perform well in terms of bandwidth in both peak and non- peak traffic scenarios. In terms of RTT all algorithms perform well in non-peak traffic scenarios, but, only ASQM performs well when buffering displacement occurs because of peak traffic conditions.

[5] http://tools.ietf.org/html/draft-kuhn-aqm-eval-guidelines-00#section-3.2.4

[6] http://linux.die.net/man/8/tc-htb

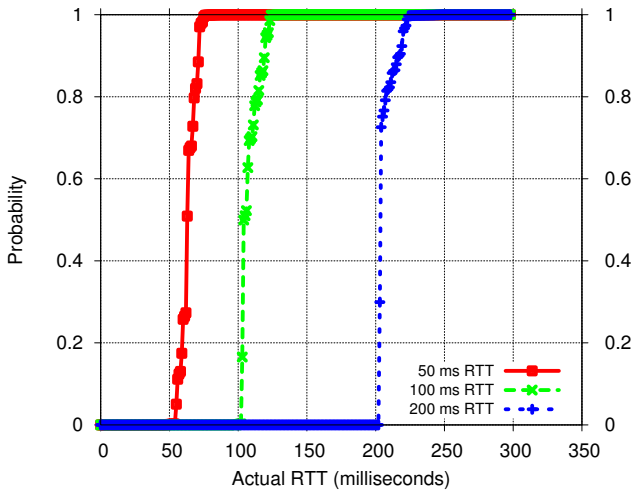

Fig. 4. CoDel RTT CDF (Non-Peak Hours)

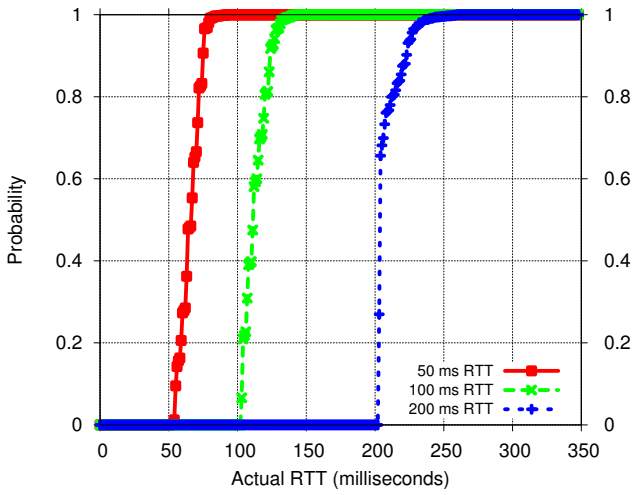

Fig. 6. PIE RTT CDF (Non-Peak Hours)

A. AQM During Non-Peak Hours (100% Committed Rate or More)

We present this series of CDFs in order to demonstrate that our ASQM algorithm performs on par with CoDel and PIE during non-peak traffic hours. In Figure 4, we present the end to end delay curves for a CoDel (with default parameters) managed link. We examined three different RTTs from 50 ms to 200 ms. CoDel had excellent RTT response across the range of RTTs. At 50 ms RTT CoDel's management kept the end to end delay within a range from 50-75 ms. At 100 ms RTT the end to end delay range was 100-125 ms and at 200 ms RTT the range was 200-225 ms. In Figure 5, we present corresponding bandwidth curves for these experiments.

Each link had a committed rate of 8 Mbps with a peak rate of 16 Mbps. Since the the non-peak hours link had enough bandwidth to supply all three modems with their full peak rate each modem was able to develop about 15.5 Mbps measured bandwidth. They did not reach 16 Mbps because of overhead from Ethernet, IP and Transport headers. When the RTT was 50 ms the CoDel managed link developed full bandwidth. However as the RTT increased the bandwidth decreased slightly. This is because CoDel was keeping the queue size at about 100 ms (the default) which is slightly too small for the 200 ms flow.

In Figure 6, we present the end to end delay curves for a PIE managed link across a range of RTTs (50-200 ms). PIE also had an excellent RTT response. It kept the actual RTT range experienced by the link to about 50-75 ms for a 50 ms RTT link, 100-125 ms for a 100 ms RTT link and 200-225 ms for a 200 ms link for 90% of the measurements.

In Figure 7, we present the bandwidth curves for PIE. The CDF curves show that PIE delivered similar bandwidth performance as CoDel at 50 and 100 ms RTT. However, PIE delivered slightly less bandwidth than CoDel at 200 ms RTT. This shows that PIE was slightly more

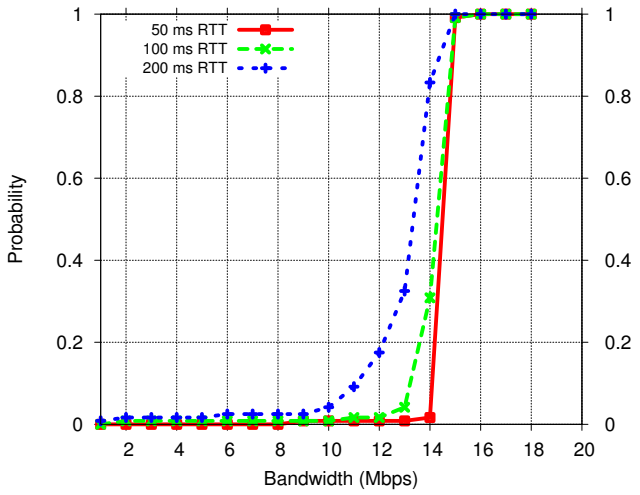

Fig. 5. CoDel Bandwidth CDF (Non-Peak Hours)

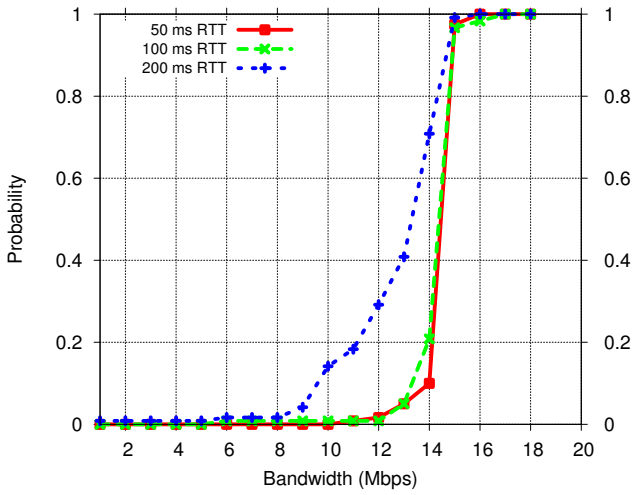

Fig. 7. PIE Bandwidth CDF (Non-Peak Hours)

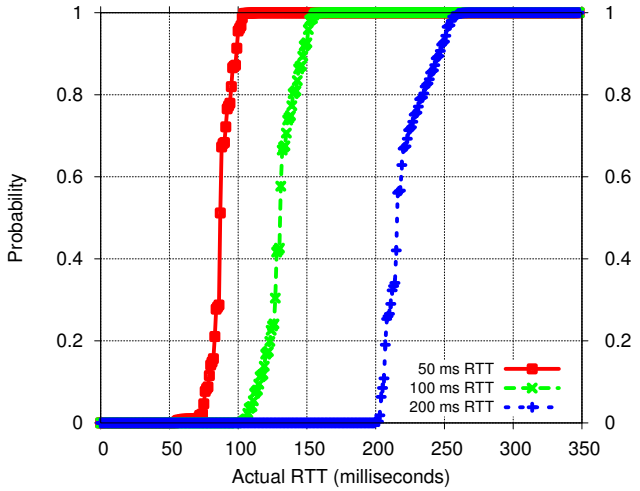

Fig. 8. ASQM RTT CDF (Non-Peak Hours)

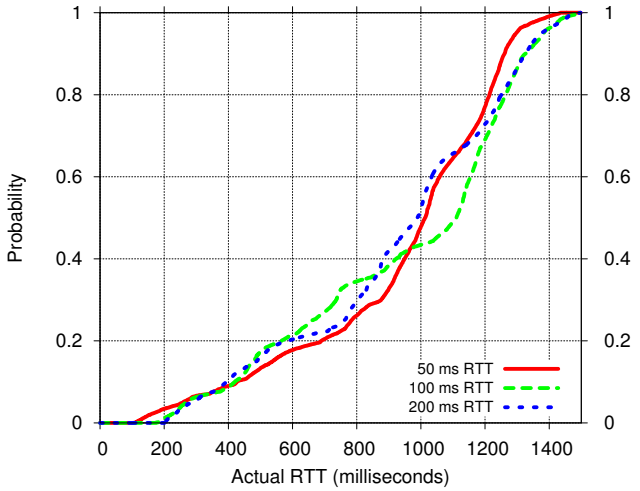

Fig. 10. CoDel RTT CDF (Peak Hours)

aggressive with its dropping policy. In any case, both of these algorithms deliver excellent queue management characteristics across a wide range of RTTs.

In Figures 8 and 9, we present the bandwidth and delay curves for ASQM. ASQM also delivered excellent RTT response as shown in Figure 8 (although not quite as good as CoDel and PIE). ASQM allowed the actual measured RTT to exceed the link RTT by only as much as 50 ms. In Figure 9, we present the bandwidth curves for ASQM. ASQM achieved slightly more bandwidth than either CoDel or PIE across the range of link RTTs (50-200 ms). This is because ASQM is slightly less aggressive than CoDel or PIE in its dropping policy.

These bandwidth and delay CDFs show that all three algorithms are fully capable of delivering excellent queue management during non-peak traffic hours when the link is capable of providing 100% or more of the rated bandwidth. Each algorithm performs slightly better or worse than the others in given scenarios. These slight differences are insignificant to the end user. As expected each of these three algorithms performed well in the non-peak traffic scenario and ASQM is on par with the others.

B. AQM During Peak Hours (Less than 100% Committed Rate)

We present this series of CDFs in order to demonstrate that our ASQM algorithm continues to perform excellently during peak traffic hours when CoDel and PIE cannot. In order to demonstrate this performance we designed a series of experiments designed to emulate peak traffic periods in ISP networks. All three modems had a committed rate of 8 Mbps and a peak rate of 16 Mbps. The access link however was only capable of delivering 21 Mbps (about 90% of the committed rates for all three modems) as is common during peak traffic hours from 7:00pm to 11:00pm. This caused bufferbloat displacement defeating

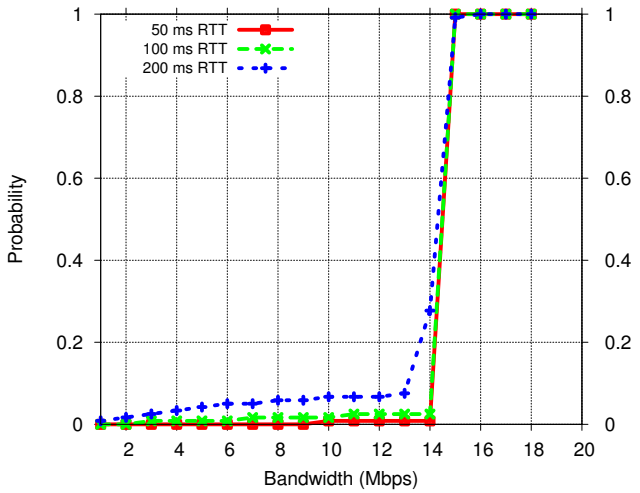

Fig. 9. ASQM Bandwidth CDF (Non-Peak Hours)

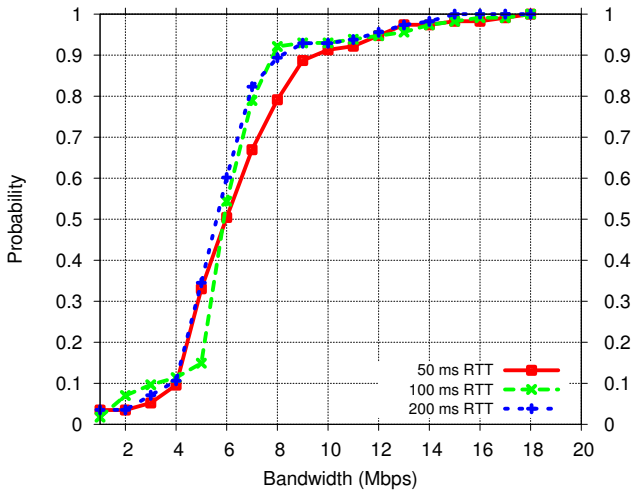

Fig. 11. CoDel Bandwidth CDF (Peak Hours)

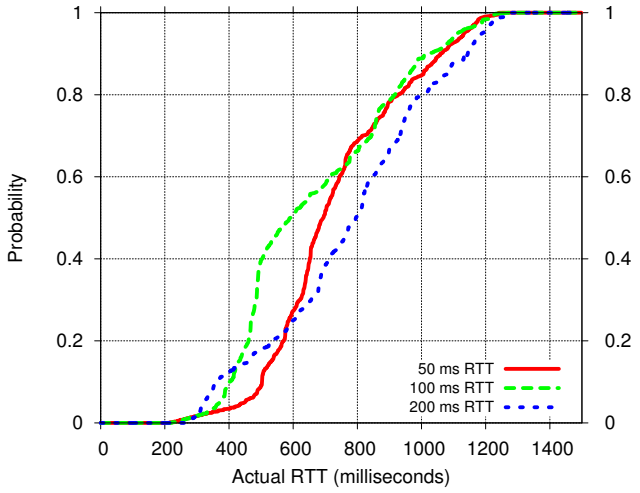

Fig. 12. PIE RTT CDF (Peak Hours)

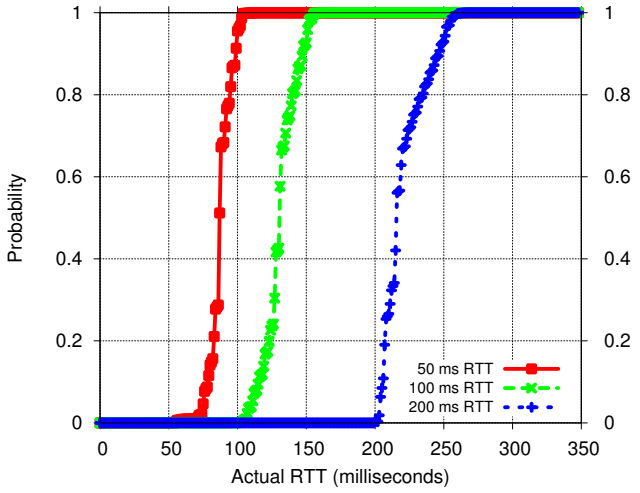

Fig. 14. ASQM RTT CDF (Peak Hours)

PIE and CoDel's ability to manage the queue size. ASQM was able to manage the queue size regardless of the bufferbloat displacement.

In Figures 10 and 11, we present the bandwidth delay curves for CoDel. Figure 10 shows that CoDel was unable to manage the queuing delay in this scenario. The actual RTTs varied widely from 100 ms to about 1500 ms. This is because bufferbloat displacement caused the queuing delay to move into the link where CoDel cannot detect it. Even though the real RTT had skyrocketed to 1500 ms CoDel still did not take corrective action.

Figure 11 shows the bandwidth curves for CoDel. The bandwidth is a lot more variable than in the non-peak traffic hours experiments where CoDel was able to control the queue. This is symptomatic of an uncontrolled queue. Each modem builds up the queue until full and then a massive number of packets are dropped causing the bandwidth of that particular modem to crash. When this happens the other two modems grab the free bandwidth allowing their bandwidth to temporarily surge.

Figures 12 and 13 show a similar story for PIE. The actual measured RTTs varied widely from about 200 ms to about 1200 ms. Like CoDel, PIE was unable to manage the delay effectively when buffering displacement occurred. Actual RTT reached 1200 ms without PIE taking corrective action. We present the bandwidth curves for PIE in Figure 13. Like the bandwidth curves for CoDel during peak traffic hours PIE's bandwidth curves had a lot of variability. The cause of this variability is that PIE's control mechanism is not taking corrective action to control the queue.

Figures 14 and 15 show the bandwidth delay CDFs for ASQM during peak traffic hours. ASQM's active sensing mechanism detected queuing delay across the link regardless of the buffering displacement. Because of this ASQM took corrective action when the queuing delay exceeded its

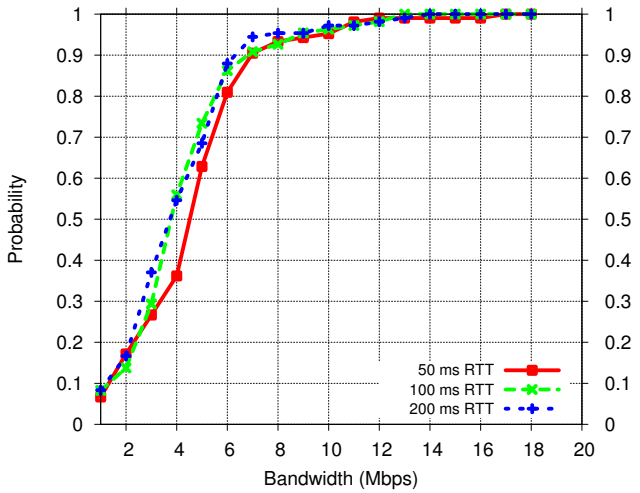

Fig. 13. PIE Bandwidth CDF (Peak Hours)

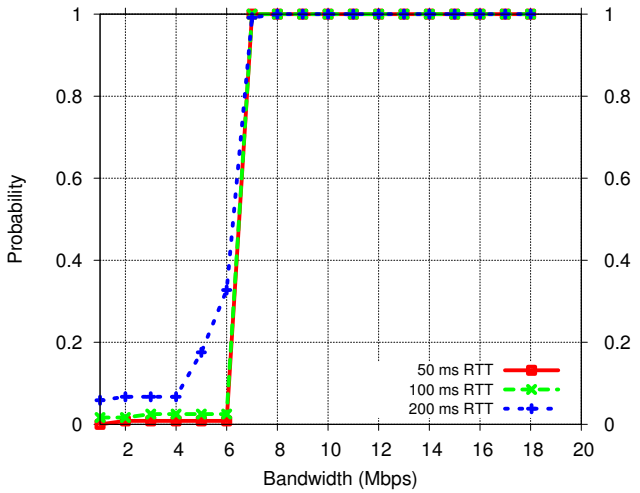

Fig. 15. ASQM Bandwidth CDF (Peak Hours

target value. Figure 14 we see that ASQM's delay curves were virtually unaffected by the buffering displacement. Figure 15 shows the bandwidth curves for ASQM during peak traffic hours. The bandwidth is stable at the full available rate because ASQM was controlling the queuing delay.

This evaluation fulfilled both of its goals. We have shown that all three algorithms perform well in terms of bandwidth and RTT during non-peak traffic hours when the access link provides 100% or more of the rated bandwidth. Additionally we have shown that during peak traffic hours when bufferbloat displacement commonly occurs that the other algorithms are unable to control RTT. Of the three only ASQM is able to provide a stable bandwidth at the full available rate with a managed RTT when bufferbloat displacement caused by peak traffic hours occurs.

VII. Summary, Conclusions and Future Work

In this work we have presented ASQM. ASQM is a new class of AQM algorithm using an active sensing mechanism to detect queuing delay across the entire access link. Traditional AQM algorithms can only detect queuing delay in the IP layer. We have presented experiments demonstrating how ASQM is able to manage queuing delay even when bufferbloat displacement occurs during peak traffic hours.

We have conducted thousands of experiments (besides the few presented in this paper). We have found that it is irrelevant how much less than 100% of the committed rate is provided. Even the slightest bit less than 100% causes bufferbloat displacement and defeats PIE and CoDel's ability to sense bufferbloat. We chose to present a range of RTTs from 50 ms to 200 ms. We chose these values because they provide a good range around the 100 ms target delay. Also we wanted to avoid the loss of bandwidth that occurs in all algorithms at larger delays. This is a well known problem in the AQM field called the Long Delay Flow problem.

PIE and CoDel both purport an operating range of 10 ms to 500 ms, ASQM's operating range is about the same. However, all three algorithms begin losing bandwidth due to the Long Delay Flow problem at around 250 ms. After 500 ms the algorithms become unusable due to severe bandwidth loss. We reserve this problem for future work.

References

[1] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. *SIGCOMM*, 34(4):281–292, Aug. 2004.

[2] S. Bauer, R. Beverly, and A. Berger. Measuring the State of ECN Readiness in Servers, Clients, and Routers. In *Internet Measurement Conference*, pages 171–180, 2011.

[3] J. Bennett and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. *Networking*, 5(5):675–689, Oct. 1997.

[4] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine. A Framework for Integrated Services Operation over Diffserv Networks. RFC 2998, Nov. 2000.

[5] L. Brakmo and L. Peterson. TCP Vegas: end to end congestion avoidance on a global Internet. *Communications*, 13(8):1465–1480, Oct. 1995.

[6] C. Caini and R. Firrincieli. TCP Hybla: a TCP enhancement for heterogeneous networks. *Satellite Communications and Networking*, 22(5):547–566, 2004.

[7] N. Dukkipati, M. Mathis, Y. Cheng, and M. Ghobadi. Proportional rate reduction for TCP. In *IMC*, pages 155–170, 2011.

[8] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *SIGCOMM*, 26(3):5–21, July 1996.

[9] W. Feng, K. G. Shin, D. D. Kandlur, and D. Saha. The BLUE active queue management algorithms. *Networking*, 10(4):513–528, Aug. 2002.

[10] S. Floyd, R. Gummadi, S. Shenker, and Others. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. *Preprint, available at http://www. icir. org/floyd/papers. html*, 2001.

[11] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *Networking*, 1(4):397–413, Aug. 1993.

[12] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *Networking*, 3(4):365–386, Aug. 1995.

[13] C. Fu and S. Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. *Communications*, 21(2):216–228, Feb. 2003.

[14] T. Hamann and J. Walrand. A new fair window algorithm for ECN capable TCP (new-ECN). In *INFOCOM*, volume 3, pages 1528–1536, Mar. 2000.

[15] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348, Sept. 2008.

[16] S. Hassayoun, J. Iyengar, and D. Ros. Dynamic Window Coupling for multipath congestion control. In *ICNP*, pages 341–352, Oct. 2011.

[17] D. Hayes and G. Armitage. Revisiting TCP Congestion Control Using Delay Gradients. In *Networking*, volume 6641 of *Lecture Notes in Computer Science*, pages 328–341. Springer Berlin Heidelberg, 2011.

[18] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it still possible to extend TCP? In *IMC*, pages 181–194, 2011.

[19] R. Jain. *Design and implementation of split tcp in the linux kernel*. PhD thesis, 2007.

[20] W. Kim and B. G. Lee. FRED fair random early detection algorithm for TCP over ATM networks. *Electronics Letters*, 34(2):152–154, Jan. 1998.

[21] M. Kühlewind, S. Neuner, and B. Trammell. On the State of ECN and TCP Options on the Internet. In *Passive and Active Measurement*, pages 135–144, 2013.

[22] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. In *SIGCOMM*, pages 123–134, 2001.

[23] S. Liu, T. Ba\csar, and R. Srikant. TCP Illinois: a loss and delay-based congestion control algorithm for high-speed networks. In *Performance Evaluation Methodolgies and Tools*, 2006.

[24] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. In *Mobile Computing and Networking*, pages 287–297, 2001.

[25] K. Nichols and V. Jacobson. Controlling queue delay. *ACM Communications*, 55(7):42–50, July 2012.

[26] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *Networking*, 8(2):133–145, Apr. 2000.

[27] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg. PIE: A lightweight control scheme to address the bufferbloat problem. In *High Performance Switching and Routing*, pages 148–155, July 2013.

[28] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ecn) to ip. RFC 3168, Sept. 2001.

[29] R. Stewart and C. Metz. SCTP: new transport protocol for TCP/IP. *IEEE Internet Computing*, 5(6):64–69, 2001.

[30] D. Stiliadis and A. Varma. Efficient fair queueing algorithms for packet-switched networks. *Networking*, 6(2):175–185, Apr. 1998.

[31] I. Stoica, H. Zhang, and T. Ng. A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services. *Networking*, 8(2):185–199, Apr. 2000.

[32] J. Sundararajan, D. Shah, M. MeÌĄdard, S. Jakubczak, M. Mitzenmacher, and J. Barros. Network Coding Meets TCP: Theory and Implementation. *Proceedings of the IEEE*, 99(3):490–512, Mar. 2011.

[33] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A Compound TCP Approach for High-Speed and Long Distance Networks. In *INFOCOM*, pages 1–12, Apr. 2006.

[34] A. Tang, J. Wang, S. Hegde, and S. Low. Equilibrium and Fairness of Networks Shared by TCP Reno and Vegas/FAST. *Telecommunication Systems*, 30(4):417–439, 2005.

[35] C. Villamizar and C. Song. High Performance TCP in ANSNET. *SIGCOMM*, 24(5):45–60, Oct. 1994.