International Journal of Research in Science and Technology
(IJRST) 2014, Vol. No. 4, Issue No. I, January-March
http://www.ijrst.com
ISSN: 2249-0604

A LIGHT ON XCP TO AVOID CONGESTION FOR HIGH BANDWIDTH NETWORK

Syed Nusrat, #Farman Ali, @Agha Salman Haider

**, #Research Scholar, Shri Jjt University @Research Scholar, Manav Bharti University*

ABSTRACT

Technology trends indicate that the future Internet will have a large number of very high-bandwidth links.Less ubiquitous but still commonplace will be satellite and wireless links with high latency. These trends are problematic because TCP reacts adversely to increases in the per-flow bandwidth-delay product.

This extended abstract proposes a novel congestion control protocol, called XCP, that outperforms TCP in both traditional and high bandwidth-delay product environments. The problem facing TCP as the per-flow bandwidth-delay increases is multi-fold. First, mathematical analysis of TCP congestion control reveals that, regardless of the queuing scheme, as the delay-bandwidth product increases, TCP becomes oscillatory and prone to instability. By casting the problem into a control theory framework, Low et al. [10] show that as capacity or delay increases, Random Early Discard (RED) [4], Random Early Marking (REM) [2], Proportional Integral Controller [6], and Virtual Queue [5] all eventually become oscillatory and prone to instability. They further argue that it is unlikely that any Active Queue Management scheme (AQM) can maintain stability over very high-capacity or large-delay links. Furthermore, Katabi and Blake [7] show that Adaptive Virtual Queue (AVQ) [9] also becomes prone to instability when the link capacity is large enough (e.g., gigabit links).

Inefficiency is another problem facing TCP in the future Internet. As the delay bandwidth product increases, performance degrades. TCP's additive increase policy limits its ability to acquire spare bandwidth to one packet per RTT. Since the bandwidth-delay product of a single flow over very-high-bandwidth links may be many thousands of packets, TCP might waste thousands of RTTs ramping up to full utilization following a burst of congestion. To address the above problem, we compared a novel protocol for congestion control that outperforms TCP in conventional environments, and further remains efficient, fair, and stable as the link bandwidth or the round-trip delay increases [8]. This protocol is eXplicit Control Protocol, XCP, generalizes the Explicit Congestion Notification proposal (ECN) [12]. Instead of the one bit congestion indication used by ECN, our routers inform the senders about the degree of congestion at the bottleneck. Another new concept is the decoupling of utilization control from fairness control.

Methodology

In this paper we gives a light to the novel protocol (XCP) for congestion control that outperforms TCP in conventional environments.

Conclusion

To control utilization, this novel protocol adjusts its aggressiveness according to the spare bandwidth in the network and the feedback delay. This prevents oscillations, provides stability in face of high bandwidth or large delay, and ensures efficient utilization of network resources. To control fairness, the protocol reclaims bandwidth from flows whose rate is above their fair share and reallocates it to other flows. By putting the control state in the packets, XCP needs no per-flow state in routers and can scale to any number of flows. Further, implementation of XCP, requires only a few CPU cycles per packet, making it practical even for high-speed routers.

Keywords*: Networks; TCP, XCP; Congestion control*

INTRODUCTION

Our main goal is to solve TCP's limitations in high-bandwidth large-delay environments, XCP design has several additional advantages. First, decoupling fairness control from utilization control provides a flexible framework for integrating differential bandwidth allocations. For example, allocating bandwidth to senders according to their priorities or the price they pay requires changing only the fairness controller and does not affect the efficiency or the congestion characteristics [8].
Second, this novel protocol facilitates distinguishing error losses from congestion losses, which makes it useful for wireless environments. In XCP, drops caused by congestion are highly uncommon (e.g., less than one in a million packets in simulations).
Further, since the protocol uses explicit and precise congestion feedback, a congestion drop is likely to be preceded by an explicit feedback that tells the source to decrease its congestion window. Losses that are preceded and followed by an explicit increase feedback are likely error losses.

SIGNIFICANCE

The problem facing TCP as the per-flow bandwidth-delay increases is multi-fold.

First, mathematical analysis of TCP congestion control reveals that, regardless of the queuing scheme, as the delay-bandwidth product increases, TCP becomes oscillatory and prone to instability. By casting the problem into a control theory framework, Low et al. [10] show that as capacity or delay increases, Random Early Discard (RED) [4], Random Early Marking (REM) [2], Proportional Integral Controller [6], and Virtual Queue [5] all eventually become oscillatory and prone to instability. They further argue that it is unlikely that *any* Active Queue Management scheme (AQM) can maintain stability over very high-capacity or large-delay links. Furthermore, Katabi and Blake [7] show that Adaptive Virtual Queue (AVQ) [9] also becomes prone to instability when the link capacity is large enough (e.g., gigabit links).

Inefficiency is another problem facing TCP in the future Internet. As the delay bandwidth product increases, performance degrades. TCP's additive increase policy limits its ability to acquire spare

bandwidth to one packet per RTT. Since the bandwidth-delay product of a single flow over very-high-bandwidth links may be many thousands of packets, TCP might waste thousands of RTTs ramping up to full utilization following a burst of congestion.

Further, the increase in link capacity does not improve the transfer delay of short flows (the majority of the flows in the Internet). Short TCP flows cannot acquire the spare bandwidth faster than "slow start" and will waste valuable RTTs ramping up even when bandwidth is available. Additionally, since TCP's throughput is inversely proportional to the RTT, fairness too might become an issue as more flows in the Internet traverse satellite links or wireless WANs [11]. As users with substantially different RTTs compete for the same bottleneck capacity, considerable unfairness will result. Although the full impact of large delay-bandwidth products is yet to come, we can see the seeds of these problems in the current Internet. For example, TCP over satellite links has revealed network utilization issues and TCP's undesirable bias against long RTT flows [1]. Currently, these problems are mitigated using ad hoc mechanisms such as ack spacing, split connection [1], or performance enhancing proxies [3].

OBJECTIVES AND HYPOTHESIS

Our initial objective is to step back and rethink Internet congestion control without caring about backward compatibility or deployment. Our objective is to find a better mechanism to avoid congestion
For future high bandwidth delay network, which work as TCP but avoid the problems with TCP for high bandwidth delay network.

RESEARCH METHODOLOGY

In this paper we gives a light to the novel protocol (XCP) for congestion control that outperforms TCP in conventional environments.

An Overview over XCP

We started with the goal of solving TCP's limitations in high-bandwidth large-delay environments, XCP design has several additional advantages.

First, decoupling fairness control from utilization control opens new avenues for service differentiation using schemes that provide desired bandwidth apportioning, yet are too aggressive or too weak for controlling congestion. We present a simple scheme that implements the shadow prices model [21].
Second, the protocol facilitates distinguishing error losses from congestion losses, which makes it useful for wireless environments.
In XCP, drops caused by congestion are highly uncommon (e.g., less than one in a million packets in simulations). Further, since the protocol uses explicit and precise congestion feedback, a congestion drop is likely to be preceded by an explicit feedback that tells the source to decrease its congestion

window. Losses that are preceded and followed by an explicit increase feedback are likely error losses.

Third, XCP facilitates the detection of misbehaving sources.

Finally, XCP's performance provides an incentive for both end users and network providers to deploy the protocol.

DESIGN RATIONALE

Our initial objective is to step back and rethink Internet congestion control without caring about backward compatibility or deployment. If we were to build a new congestion control architecture from scratch, what might it look like? The first observation is that packet loss is a poor signal of congestion. While we do not believe a cost-effective network can always avoid loss, dropping packets should be a congestion signal of last resort. As an implicit signal, loss is bad because congestion is not the only source of loss, and because a definite decision that a packet was lost cannot be made quickly. As a binary signal, loss only signals whether there is congestion (a loss) or not (no loss). Thus senders must probe the network to the point of congestion before backing off. Moreover, as the feedback is imprecise, the increase policy must be conservative and the decrease policy must be aggressive. Tight congestion control requires explicit and precise congestion feedback. Congestion is not a binary variable, so congestion signaling should reflect the degree of congestion. We propose using precise congestion signaling, where the network explicitly tells the sender the state of congestion and how to react to it. This allows the senders to decrease their sending windows quickly when the bottleneck is highly congested, while performing small reductions when the sending rate is close to the bottleneck capacity. The resulting protocol is both more responsive and less oscillatory.

Second, the aggressiveness of the sources should be adjusted according to the delay in the feedback-loop. The dynamics of congestion control may be abstracted as a control loop with feedback delay. A fundamental characteristic of such a system is that it becomes unstable for some large feedback delay. To counter this destabilizing effect, the system must slow down as the feedback delay increases. In the context of congestion control, this means that as delay increases, the sources should change their sending rates more slowly. This issue has been raised by other researchers [23, 6], but the important question is how exactly feedback should depend on delay to establish stability. Using tools from control theory, we conjecture that congestion feedback based on rate-mismatch should be inversely proportional to delay, and feedback based on queue-mismatch should be inversely proportional to the square of delay.

Robustness to congestion should be independent of unknown and quickly changing parameters, such as the number of flows. A fundamental principle from control theory states that a controller must react as quickly as the dynamics of the controlled signal; otherwise the controller will always lag behind the controlled system and will be ineffective. In the context of current proposals for congestion control, the controller is an Active Queue Management scheme (AQM). The controlled

signal is the aggregate traffic traversing the link. The controller seeks to match input traffic to link capacity. However, this objective might be unachievable when the input traffic consists of TCP flows, because the dynamics of a TCP aggregate depend on the number of flows. The aggregate rate increases by packets per RTT, or decreases proportionally to. Since the number of flows in the aggregate is not constant and changes over time, no AQM controller with constant parameters can be fast enough to operate with an arbitrary number of TCP flows. Thus, a third objective of our system is to make the dynamics of the aggregate traffic independent from the number of flows.
This leads to the need for decoupling *efficiency control* (i.e., control of utilization or congestion) from *fairness control*. Robustness to congestion requires the behavior of aggregate traffic to be independent of the number of flows in it. However, any fair bandwidth allocation intrinsically depends on the number of flows traversing the bottleneck. Thus, the rule for dividing bandwidth among individual flows in an aggregate should be independent from the control law that governs the dynamics of the aggregate.

Traditionally, efficiency and fairness are coupled since the same control law (such as AIMD in TCP) is used to obtain both fairness and efficiency simultaneously [3, 9, 17, 18, 16]. Conceptually, however, efficiency and fairness are independent. Efficiency involves only the aggregate traffic's behavior. When the input traffic rate equals the link capacity, no queue builds and utilization is optimal. Fairness, on the other hand, involves the relative throughput of flows sharing a link. A scheme is fair when the flows sharing a link have the same throughput irrespective of congestion. In our new paradigm, a router has both an efficiency controller (EC) and a fairness controller (FC). This separation simplifies the design and analysis of each controller by reducing the requirements imposed. It also permits modifying one of the controllers without redesigning or re-analyzing the other. Furthermore, it provides a flexible framework for integrating differential bandwidth allocations. For example, allocating bandwidth to senders according to their priorities or the price they pay requires changing only the fairness controller and does not affect the efficiency or the congestion characteristics.

Sender's current cwnd (filled by sender and remains unmodified)
Sender's RTT estimate (filled by sender and remains unmodified)
Feedback (initialized to sender's demands; can be modified by therouters)

Figure 1: Congestion header.

PROTOCOL

XCP provides a joint design of end-systems and routers. Like TCP, XCP is a window-based congestion control protocol intended for best effort traffic. However, its flexible architecture can easily support differentiated services as explained in_6. The description of XCP in this section assumes a pure XCP network. We show that XCP can coexist with TCP in the same Internet and be TCP-friendly.

First we give an overview of how control information flows in the network, then we explain feedback computation.
Senders maintain their congestion window cwnd and round trip time rtt1 and communicate these to the routers via a congestion header in every packet. Routers monitor the input traffic rate to each of their output queues. Based on the difference between the link bandwidth and its input traffic rate, the router tells the flows sharing that link to increase or decrease their congestion windows. It does this by annotating the congestion header of data packets. Feedback is divided between flows based on their cwnd and rtt values so that the system converges to fairness. A more congested router later in the path can further reduce the feedback in the congestion header by overwriting it. Ultimately, the packet will contain the feedback from the bottleneck along the path. In this document, the notation RTT refers to the physical round trip time, rtt refers to the variable maintained by the source's software, and cwnd refers to a field in the congestion headers back reaches the receiver, it is returned to the sender in an acknowledgment packet, and the sender updates its cwnd accordingly.

The Congestion Header

Each XCP packet carries a congestion header (Figure 1), which is used to communicate a flow's state to routers and feedback from the routers on to the receivers. The field H_cwnnd is the sender's current congestion window, whereas H_rtt is the sender's current RTT estimate. These are filled in by the sender and never modified in transit.The remaining field, H_feedback, takes positive or negative values and is initialized by the sender. Routers along the path modify this field to directly control the congestion windows of the sources.

The XCP Sender

As with TCP, an XCP sender maintains a congestion window of the outstanding packets, cwnd, and an estimate of the round trip time rtt. On packet departure, the sender attaches a congestion header to the packet and sets the H_cwnnd field to its current cwnd and H_rtt to its current rtt. In the first packet of a flow, H_rtt is set to zero to indicate to the routers that the source does not yet have a valid estimate of the RTT. The sender initializes the H_feedback field to request its desired window increase. For example, when the application has a desired rate r, the sender sets H_feedback to the desired increase in the congestion window (r.rtt - cwnd) divided by the number of packets in the current congestion window. If bandwidth is available, this initialization allows the sender to reach the desired rate after one RTT. Whenever a new acknowledgment arrives, positive feedback

increases the sender's cwnd and negative feedback reduces it:

cwnd = max(cwnd+ H_feedback.s),

where s is the packet size.

In addition to direct feedback, XCP still needs to respond to losses although they are rare. It does this in a similar manner to TCP.

The XCP Receiver

An XCP receiver is similar to a TCP receiver except that when acknowledging a packet, it copies the congestion header from the data packet to its acknowledgment.

The XCP Router: The Control Laws

The job of an XCP router is to compute the feedback to cause the system to converge to optimal efficiency and min-max fairness. Thus, XCP does not drop packets. It operates on top of a dropping policy such as Drop Tail, RED, or AVQ. The objective of XCP is to prevent, as much as possible, the queue from building up to the point at which a packet has to be dropped. To compute the feedback, an XCP router uses an *efficiency controller* and a *fairness controller*. Both of these compute estimates over the average RTT of the flows traversing the link, which smoothes the burstiness of a window-based control protocol. Estimating parameters over intervals longer than the average RTT leads to sluggish response, while estimating parameters over shorter intervals leads to erroneous estimates. The average RTT is computed using the information in the congestion header.
XCP controllers make a single control decision every average RTT (the control interval). This is motivated by the need to observe the results of previous control decisions before attempting a new control. For example, if the router tells the sources to increase their congestion windows, it should wait to see how much spare bandwidth remains before telling them to increase again. The router maintains a per-link estimation-control timer that is set to the most recent estimate of the average RTT on that link.
Upon timeout the router updates its estimates and its control decisions. In the remainder of this paper, we refer to the router's current estimate of the average RTT as d to emphasize this is the feedback delay.

The Efficiency Controller (EC)

The efficiency controller's purpose is to maximize link utilization while minimizing drop rate and persistent queues. It looks only at aggregate traffic and need not care about fairness issues, such as which flow a packet belongs to. As XCP is window-based, the EC computes a desired increase or decrease in the number of bytes that the aggregate traffic transmits in a control interval (i.e., an average RTT).

The Fairness Controller (FC)
The job of the fairness controller (FC) is to apportion the feedback to individual packets to achieve fairness. The FC relies on the same principle TCP uses to converge to fairness, namely Additive-Increase Multiplicative-Decrease (AIMD).

Notes on the Efficiency and Fairness Controllers

This section summarizes the important points about the design of the efficiency controller and the fairness controller. As mentioned earlier, the efficiency and fairness controllers are decoupled. Specifically, the efficiency controller uses a Multiplicative-Increase Multiplicative-Decrease law (MIMD), which increases the traffic rate proportionally to the spare bandwidth in the system (instead of increasing by one packet/RTT/flow as TCP does). This allows XCP to quickly acquire the positive spare bandwidth even over high capacity links. The fairness controller, on the other hand, uses an Additive-Increase Multiplicative-Decrease law (AIMD), which converges to fairness [10]. Thus, the decoupling allows each controller to use a suitable control law. The particular control laws used by the efficiency controller (MIMD) and the fairness controller (AIMD) are not the only possible choices. For example, in [20] we describe a fairness controller that uses a binomial law similar to those described in [6]. We chose the control laws above because our analysis and simulation demonstrate their good performance.

STABILITY ANALYSIS

We use a fluid model of the traffic to analyze the stability of XCP. Our analysis considers a single link traversed by multiple XCP flows. For the sake of simplicity and tractability, similarly to previous work [22, 15, 23, 24], our analysis assumes all flows have a common, finite, and positive round trip delay, and neglects boundary conditions (i.e., queues are bounded, rates cannot be negative). Later, we demonstrate through extensive simulations that even with larger topologies, different RTTs, and boundary conditions, our results still hold.

PERFORMANCE

In this section, we demonstrate through extensive simulations that XCP outperforms TCP both in conventional and high bandwidth delay environments. Our simulations also show that XCP has the unique characteristic of almost never dropping packets.

Comparison with TCP and AQM SchemesImpact of Capacity:

We show that an increase in link capacity (with the resulting increase of per-flow bandwidth) will cause a significant degradation in TCP's performance, irrespective of the queuing scheme. In this experiment, 50 long-lived FTP flows share a bottleneck. The round trip propagation delay is 80 ms. additionally, there are 50 flows traversing the reverse path and used merely to create a 2-way traffic environment with the potential for ack compression. Since XCP is based on a fluid model and

estimates some parameters, the existence of reverse traffic, with the resulting burstiness, tends to stress the protocol. As capacity increases, TCP's bottleneck utilization decreases significantly. This happens regardless of the queuing scheme.

In contrast, XCP's utilization is always near optimal independent of the link capacity. Furthermore, XCP never drops any packet, whereas TCP drops thousands of packets despite its use of ECN. Although the XCP queue increases with the capacity, the queuing delay does not increase because the larger capacity causes the queue to drain faster.

The Dynamics of XCP

While the simulations presented above focus on long term average behavior, this section shows the short term dynamics of XCP. In particular, we show that XCP's utilization, queue size, and throughput exhibit very limited oscillations. Therefore, the average behavior presented in the section above is highly representative of the general behavior of the protocol.

SECURITY

Similarly to TCP, in XCP security against misbehaving sources requires an additional mechanism that polices the flows and ensures that they obey the congestion control protocol. This may be done by policing agents located at the edges of the network. The agents maintain per-flow state and monitor the behavior of the flows to detect network attacks and isolate unresponsive sources. Unlike TCP, XCP facilitates the job of these policing agents because of its explicit feedback. Isolating the misbehaving source becomes faster and easier because the agent can use the explicit feedback to test a source. More precisely, in TCP isolating an unresponsive source requires the agent/router to monitor the average rate of a suspect source over a fairly long interval to decide whether the source is reacting according to AIMD. Also, since the source's RTT is unknown, it's correct sending rate is not specified, which complicates the task even further. In contrast, in XCP, isolating a suspect flow is easy. The router can send the flow a test feedback requiring it to decrease its congestion window to a particular value. If the flow does not react in a single RTT then it is unresponsive. The f» act that the flow specifies its RTT in the packet makes the monitoring easier. Since the flow cannot tell when an agent/router is monitoring its behavior, it has to always follow the explicit feedback.

A TCP friendly XCP

In this section, we describe a mechanism allowing end-to-end XCP to compete fairly with TCP in the same network. This design can be used to allow XCP to exist in a multi-protocol network, or as a mechanism for incremental deployment.

To start an XCP connection, the sender must check whether the receiver and the routers along the path are XCP-enabled. If they are not, the sender reverts to TCP or another conventional protocol. These checks can be done using simple TCP and IP options. We then extend the design of an XCP

router to handle a mixture of XCP and TCP flows while ensuring that XCP flows are TCP-friendly. The router distinguishes XCP traffic from non-XCP traffic and queues it separately. TCP packets are queued in a conventional RED queue (the *T-queue*). XCP flows are queued in an XCP-enabled queue. To be fair, the router should process packets from the two queues such that the average throughput observed by XCP flows equals the average throughput observed by TCP flows, irrespective of the number of flows.

CONCLUSION

Theory and simulations suggest that current Internet congestion control mechanisms are likely to run into difficulty in the long term as the per-flow bandwidth-delay product increases. This motivated us to step back and re-evaluate both control law and signalling for congestion control. Motivated by CSFQ, we chose to convey control information between the end-systems and the routers using a few bytes in the packet header. The most important consequence of this explicit control is that it permits a decoupling of *congestion control* from *fairness control*. In turn, this decoupling allows more efficient use of network resources and more flexible bandwidth allocation schemes. Based on these ideas, we devised XCP, an explicit congestion control protocol and architecture that can control the dynamics of the aggregate traffic independently from the relative throughput of the individual flows in the aggregate. Controlling congestion is done using an analytically tractable method that matches the aggregate traffic rate to the link capacity, while preventing persistent queues from forming. The decoupling then permits XCP to reallocate bandwidth between individual flows without worrying about being too aggressive in dropping packets or too slow in utilizing spare bandwidth. We demonstrated a fairness mechanism based on *bandwidth shuffling* that converges much faster than TCP does, and showed how to use this to implement both min-max fairness and the differential bandwidth allocation. Our extensive simulations demonstrate that XCP maintains good utilization and fairness, has low queuing delay, and drops very few packets. We evaluated XCP in comparison with TCP over RED, REM, AVQ, and CSFQ queues, in both steady-state and dynamic environments with web-like traffic and with impulse loads. We found no case where XCP performs significantly worse than TCP. In fact when the per-flow delay-bandwidth product becomes large, XCP's performance remains excellent whereas TCP suffers significantly. We believe that XCP is viable and practical as a congestion control scheme. It operates the network with almost no drops, and substantially increases the efficiency in high bandwidth-delay product environments.

ACKNOWLEDGEMENTS

This research is part of my PhD work. I am doing PhD in Computer Science. This project will help to find the better protocol to control the congestion. Congestion control is very necessary for future high bandwidth-delay network

REFERENCES

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

[2] *Red parameters. http://www.icir.org/floyd/red.html#parameters.*

[3] *Y. Afek, Y. Mansour, and Z. Ostfeld. Phantom: A simple and effective flow control scheme. In Proc. of ACM SIGCOMM, 1996.*

[4] *M. Allman, D. Glover, and L. Sanchez. Enhancing tcp over satellite channels using standard mechanisms, Jan. 1999.*

[5] *S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. Rem: Active queue management. IEEE Network, 2001.*

[6] *D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In Proc. of IEEE INFOCOM '01, Apr. 2001.*

[7] *D. Bansal, H. Balakrishnan, and S. S. S. Floyd. Dynamic behavior of slowly-responsivecongestion control algorithms. In Proc. of ACM SIGCOMM, 2001.*

[8] *J. Border, M. Kojo, J. Griner, and G. Montenegro. Performance enhancing proxies, Nov. 2000.*

[9] *A. Charny. An algorithm for rate allocation in a packet-switching network with feedback, 1994.*

[10] *D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidancein computer networks.In Computer Networks and ISDN Systems 17, page 1-14, 1989.*

[11] *M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In IEEE/ACM Transactions on Networking, 5(6):835–846, Dec.1997.*

[12] *S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In Proc. of ACM SIGCOMM, Aug. 2000.*

[13] *S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. In IEEE/ACM Transactions on Networking, 1(4):397–413, Aug. 1993.*

[14] *R. Gibbens and F. Kelly. Distributed connection acceptance control for a connectionlessnetwork. In Proc. of the 16th Intl. Telegraffic Congress, June 1999.*

[15] *C. Hollot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for aqm routers supporting tcp flows. In Proc. of IEEE INFOCOM, Apr. 2001.*

[16] V. Jacobson. Congestion avoidance and control. ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium, 18, 4:314–329, Aug. 1988.R. Jain, S. Fahmy, S. Kalyanaraman, and R. Goyal. The Erica switch algorithm for abr traffic management in atm networks: Part ii: Requirements and performance evaluation. In The Ohio State University, Department of CIS, Jan. 1997.

[17] R. Jain, S. Kalyanaraman, and R. Viswanathan. The osu scheme for congestion avoidance in atm networks: Lessons learnt and extensions. In Performance Evaluation Journal, Vol. 31/1-2, Dec. 1997.

[18] D. Katabi and C. Blake. A note on the stability requirements of adaptive virtual queue. MIT Technichal Memo, 2002.

[19] D. Katabi and M. Handley. Precise feedback for congestion control in the internet. MITTechnical Report, 2001.

[20] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability.

[21] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue. In Proc. of ACM SIGCOMM, 2001.

[22] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle. Dynamics of tcp/aqm and a scalable control. In Proc. Of IEEE INFOCOM, June 2002.

[23] V. Misra, W. Gong, and D. Towsley. A fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. Aug. 2000.

[24] G. Montenegro, S. Dawkins, M. Kojo, V. Magret, and N. Vaidya. Long thin networks, Jan. 2000.

[25] F. Paganini, J. C. Doyle, and S. H. Low. Scalable laws for stable network congestion control. In

IEEE CDC, 2001.

[26] K. K. Ramakrishnan and S. Floyd. Proposal to add explicit congestion notification (ecn) to ip. RFC 2481, Jan. 1999.

[27] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queuing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In Proc. Of ACM SIGCOMM, Aug. 1998.