Stability Analysis of QCN: The Averaging Principle

Mohammad Alizadeh, Abdul Kabbani, Berk Atikoglu, and Balaji Prabhakar
Department of Electrical Engineering, Stanford University
{alizade, akabbani, atikoglu, balaji}@stanford.edu

ABSTRACT

Data Center Networks have recently caused much excitement in the industry and in the research community. They represent the convergence of networking, storage, computing and virtualization. This paper is concerned with the Quantized Congestion Notification (QCN) algorithm, developed for Layer 2 congestion management. QCN has recently been standardized as the IEEE 802.1Qau Ethernet Congestion Notification standard.

We provide a stability analysis of QCN, especially in terms of its ability to utilize high capacity links in the shallow-buffered data center network environment. After a brief description of the QCN algorithm, we develop a delay-differential equation model for mathematically characterizing it. We analyze the model using a linearized approximation, obtaining stability margins as a function of algorithm parameters and network operating conditions. A second contribution of the paper is the articulation and analysis of the Averaging Principle (AP)—a new method for stabilizing control loops when lags increase. The AP is distinct from other well-known methods of feedback stabilization such as higher-order state feedback and lag-dependent gain adjustment. It turns out that the QCN and the BIC-TCP (and CUBIC) algorithms use the AP; we show that this enables them to be stable under large lags. The AP is also of independent interest since it applies to general control systems, not just congestion control systems.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Performance, Theory

Keywords

Data center, Layer 2 congestion control, Ethernet, QCN

SIGMETRICS'11, June 7–11, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0262-3/11/06 ...$10.00.

1. INTRODUCTION

Data centers pose interesting challenges in the areas of computing, storage and networking, and have caused the convergence of these disparate industries. Cloud computing platforms [14, 21, 3] need switching fabrics that simultaneously support latency sensitive high performance computing traffic, loss and latency sensitive storage traffic, and throughput intensive bulk data transfers. Similarly, the FCoE (Fiber Channel over Ethernet) standard [9] enables storage traffic to be carried over Ethernet. In order to facilitate this convergence, the IEEE 802.1 standards body has introduced several enhancements to classical Ethernet, notably the IEEE 802.1Qbb [24] and the IEEE 802.1Qau [23] standards. This work was undertaken by the Data Center Bridging Task Group [7].

The 802.1Qbb standard allows an Ethernet switch to pause transmission at its upstream neighbor switch on a per priority basis. This ensures that packets are not dropped, a feature that is critical for Fiber Channel traffic. However, since this can cause congestion to spread upstream and introduce spurious bottlenecks, the 802.1Qau standard enables an Ethernet switch to directly signal congestion to an Ethernet source in a manner similar to congestion control algorithms in the Internet.

This paper concerns the QCN (Quantized Congestion Notification) algorithm, which the authors have helped to develop as the IEEE 802.1Qau standard. The QCN algorithm has been described in previous work [2]; here, we are interested in analyzing the stability properties of the QCN feedback control loop.

There is an extensive literature on the design and analysis of congestion control algorithms in the Internet, especially for the high bandwidth–delay product regime [10, 18, 8, 27, 19, 28, 13]. In this regime, buffer occupancies become oscillatory (or the congestion control loop becomes "unstable"), causing link underutilization [20, 25]. The data center environment poses similar challenges, making it imperative that QCN ensure stable buffer occupancies. Specifically, data center Ethernet switches typically have buffers which are very small relative to the bandwidth-delay product and, hence, they can easily underflow or overflow. Furthermore, the number of flows which are simultaneously active on a link in a data center network is very small, typically fewer than 10. This makes it difficult to benefit from statistical multiplexing to reduce buffering requirements [4].[1]

[1] We revisit buffer sizing in Section 4 and refer to [2] for more discussion about the operating conditions in a data center.

The following are our main contributions:

(i) We obtain a delay-differential equation fluid model description of the QCN feedback control system. This model differs from conventional fluid models in that a QCN source needs *two* variables to describe its evolution as opposed to just one variable. We analyze this model using standard techniques and obtain the stability margins of the algorithm as a function of its design parameters and network conditions like link speeds, number of flows and round-trip time. These results complement the extensive investigation of QCN via simulation and experimentation conducted during the standardization process [23].

(ii) We describe the Averaging Principle (AP) which is a simple method for improving the stability of a control loop in the presence of increasing feedback delay. The QCN (and the BIC-TCP [28]) algorithm employs the AP, and we show that the AP is the underlying reason for the good stability properties of QCN. We also demonstrate the generality of the AP by applying it to various other feedback systems.

(iii) Finally, we analyze the AP and find that for linear control systems, it is *algebraically equivalent* to a PD (proportional-derivative) controller; that is, the AP controller and the PD controller are input-output equivalent. Since the PD controller is well-known to stabilize control loops when lags increase [11], this equivalence provides a precise characterization of the stability properties of the AP. This result is very useful in practice because the PD controller requires the switch to provide an *additional* derivative of the state—something difficult to achieve in practice since switches implement QCN functionality in hardware. The AP shows how an equivalent effect can be achieved without modifying switches.

Related literature on congestion control. Control theory prescribes two methods of feedback compensation and these have both been used to design stable congestion control algorithms as lags (round-trip times) increase. In one approach, an estimate of the RTT is used to find the correct "gains" for the loop to be stable. For example, this is the approach taken by FAST [27], XCP [18], RCP [8] and HSTCP [10].[2] The second approach improves stability by increasing the order of the feedback by sending higher order derivatives of the queue size. For instance, the active queue management schemes REM [5], and PI [15] compute a weighted sum of the queue size and its derivative (which equals input rate less output rate) as the congestion signal. This is also used in XCP and RCP.

On the other hand, BIC-TCP [28] operates stably in high bandwidth–delay product networks, even though it neither changes loop gains based on RTT nor uses higher order feedback. But it operates in the self-clocked universe of Internet congestion control schemes, where window size changes are made once every RTT. So there is the possibility that it implicitly exploits a knowledge of RTTs to derive stability. As we shall see, the QCN algorithm has no notion of RTTs. This and the similarity of operation of the BIC-TCP and QCN algorithms suggest there may be a more fundamental reason for their good stability. Our attempt to understand this reason has led us to the Averaging Principle.

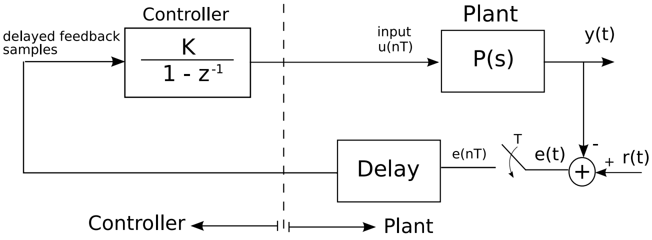


Figure 1: A generic sampled control system.

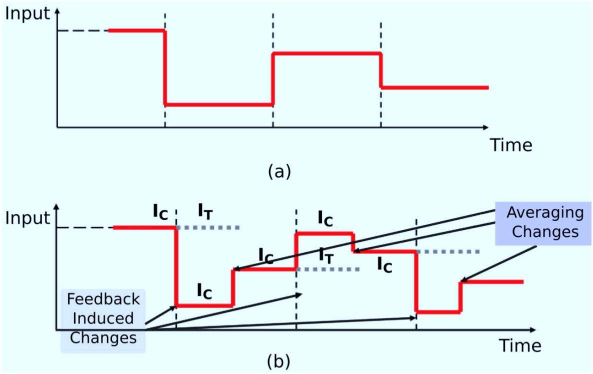


Figure 2: Plant input signal generated by (a) standard and (b) AP controller.

The Averaging Principle (AP). We explain the AP in the context of a generic control system such as the one shown in Fig. 1. The output of the system, $y(t)$, tracks a reference signal, $r(t)$, which is often a constant. The error, $e(t) = r(t) - y(t)$, is sampled with period T and fed back to the controller with some delay. The controller incrementally adjusts the input to the plant as follows:

$$u((n+1)T) = u(nT) + K\, e(nT), \tag{1}$$

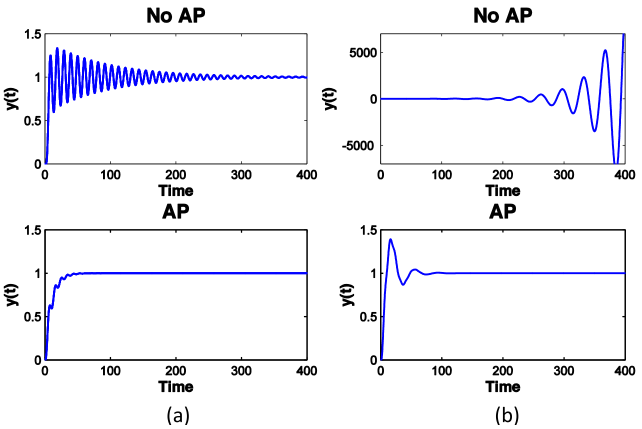
where K is the controller gain. The input is held constant between sampling times; i.e., $u(t) = u(nT)$ for $nT \leq t < (n+1)T$. Fig. 2(a) illustrates the action of the controller just described. To translate this to the congestion control setting, the source (the controller) chooses a packet sending rate (i.e., the input $u(t)$). The output $y(t)$ is the queue size and rate information at a router or a switch. The error $e(t)$ is a deviation of the current queue size and rate from target values. The sampling period T is a function of the packet arrival rate, since most congestion control algorithms sample packets.

Fig. 2(b) shows the AP controller. This controller reacts to feedback messages it receives from the plant exactly as in (1), at times which are labelled "feedback-induced changes" in the figure. Note that feedback-induced changes occur every T units of time. At any time, let I_C (for current input) denote the value of the input, and let I_T (for target input) denote the value of the input *before* the last feedback induced change.

Precisely $T/2$ time units after every feedback-induced change, the AP controller performs an "averaging change", where the controller changes I_C as follows:

$$I_C \leftarrow \frac{I_C + I_T}{2}.$$

[2] HSTCP does not explicitly use RTT estimates to adjust gains. Rather, it varies gains based on current window size, which implicitly depends on the RTT—the larger the RTT, the larger the current window.



(a) (b)

Figure 3: Unit step responses of the standard and AP controllers: (a) zero delay and (b) 8 secs delay. AP stabilizes the system.

The term "averaging" comes from the above equation: I_C moves to the average of its values before and after the last feedback-induced change. The BIC-TCP and the QCN algorithms perform averaging several times after receiving a congestion signal and we shall see that this results in their more stable operation.

To illustrate the effectiveness of the AP, let us consider an example linear, zero-delay stable control system with plant transfer function

$$P(s) = \frac{s+1}{s^3 + 1.6s^2 + 0.8s + 0.6},$$

controller gain $K = 1/8$, and sampling period $T = 1$s. We use a unit step function as the reference signal, and compare the stability of the control loop with and without AP as the delay increases. As shown in Fig. 3(a), when there is no delay, both schemes are stable. However, the AP controller induces less oscillations and settles faster. As the delay increases to $\tau = 8$s, the standard controller becomes unstable. However, the AP controller is more robust and continues to be stable. In fact, in this example, AP remains stable for delays up to $\tau = 15$s.

Organization of the paper. We briefly describe the salient features of the QCN algorithm and present the corresponding mathematical model (delay-differential equations) in Sections 2.1 and 2.2 respectively. We analyze a linearized approximation of this model in Section 2.3 to find the stability margins of the algorithm. We compare the stability of the linearized model with and without AP in Section 2.4 and find that the AP increases the stability margin. We analyze the AP in Section 3.1 and prove that, for linear control systems, it is algebraically equivalent to a PD (proportional-derivative) controller. We apply the AP to another congestion control algorithm, RCP [8], in Section 3.2 as an illustration of its wide applicability. We revisit the discussion in [4] in Section 4 and show that the AP reduces the buffering requirements at network switchs by reducing the *variance* in the sending rate of a source. We conclude in Section 5.

2. QCN

We begin with a brief overview of the QCN algorithm, focusing on those aspects which are relevant for the mathematical model.

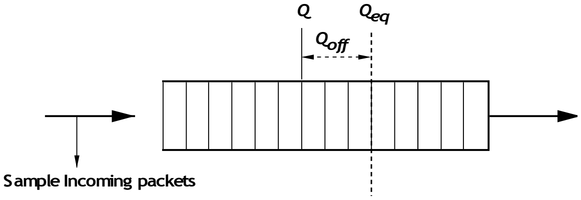


Figure 4: Congestion detection in QCN CP.

2.1 The QCN Algorithm

The QCN algorithm has two components: (i) the switch, or Congestion Point (CP) mechanism, and (ii) the source, or Reaction Point (RP) mechanism. The CP mechanism is concerned with measuring the extent of congestion at the switch buffer, and signaling this information back to the source(s). The RP mechanism is concerned with the actions that need to be taken when a congestion signal is received, and how sources must probe for available bandwidth when there is no congestion.

The CP Algorithm

The CP buffer is shown in Fig. 4. The goal of the CP is to maintain the buffer occupancy at a desired operating point, Q_{eq}. The CP computes a congestion measure F_b (defined below). With a probability p_s (1% by default), it randomly samples[3] an incoming packet and sends the value of F_b in a feedback message to the source of the sampled packet.

Let Q denote the instantaneous queue-size and Q_{old} denote the queue-size when the last packet was sampled. Let $Q_{off} = Q - Q_{eq}$ and $Q_\delta = Q - Q_{old}$. Then F_b is given by the formula:

$$F_b = Q_{off} + wQ_\delta,$$

where w is a positive constant (set to 2 by default).

The interpretation is that F_b captures a combination of queue-size excess (Q_{off}) and rate excess (Q_δ). Thus, when $F_b > 0$, either the buffer or the link or both are oversubscribed. A feedback message containing F_b, quantized to 6 bits, is sent to the source of the sampled packet *only* when $F_b > 0$; nothing is signaled when $F_b \leq 0$.

The RP Algorithm

The basic RP behavior is shown in Fig. 5. The RP algorithm maintains the following quantities:

- *Current Rate* (R_C): The sending rate at any time.
- *Target Rate* (R_T): The sending rate *just before* the arrival of the last feedback message.

Rate decrease. This occurs only when a feedback message is received, in which case R_C and R_T are updated as follows:

$$R_T \leftarrow R_C, \tag{2}$$
$$R_C \leftarrow R_C(1 - G_d F_b), \tag{3}$$

where the constant G_d is chosen so that $G_d F_{bmax} = \frac{1}{2}$; i.e. the sending rate can decrease by at most 50 %.

Rate increase. Since the RP is not given positive rate-increase signals by the network, it needs a mechanism for

[3] In the actual implementation, the sampling probability varies between 1-10% depending on the severity of congestion. We neglect this feature in this paper to keep the model tractable; refer to [2] and [17] for details.

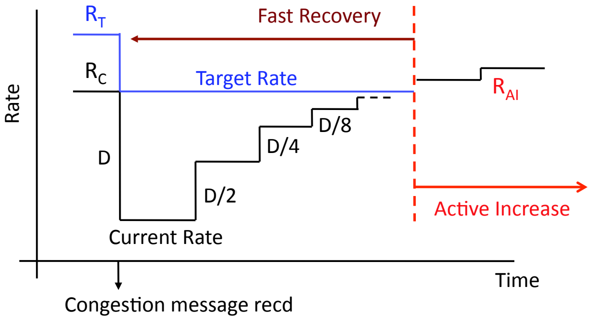


Figure 5: QCN RP operation.

increasing its sending rate on its own. This is achieved by using a *Byte Counter*, which counts the number of bytes transmitted by the RP.[4] Rate increase occurs in two phases: Fast Recovery and Active Increase.

Fast Recovery (*FR*). Immediately following a rate decrease episode, the Byte Counter is reset, and the RP enters the FR state. FR consists of 5 cycles; in each cycle 150 KBytes (100 packets, each 1500 Bytes long) of data are transmitted, as counted by the Byte Counter. At the end of each cycle, R_T remains unchanged while R_C is updated as follows:

$$R_C \leftarrow \frac{1}{2}(R_C + R_T).$$

Active Increase (*AI*). After 5 cycles of FR have completed, the RP enters the AI state where it probes for extra bandwidth on the path. In this phase, the RP increases its sending rate by updating R_T and R_C at the end of each cycle as follows:

$$R_T \leftarrow R_T + R_{AI},$$
$$R_C \leftarrow \frac{1}{2}(R_C + R_T),$$

where R_{AI} is a constant (5 Mbps by default).

Remark 1. Note that during the FR phase QCN performs *averaging.* The BIC-TCP algorithm is the first to use averaging. Indeed, the motives that led to BIC-TCP and QCN employing averaging are quite different and instructive to understand. As stated in [28], the additive increase portion of the TCP algorithm can be viewed as determining the correct window size through a *linear* search process, whereas the BIC-TCP (for Binary Increase TCP) algorithm performs the more efficient *binary* search. QCN takes a control-theoretic angle: a congestion control algorithm is zero-delay stable if the amount of rate increase after a drop is less than the amount of decrease during the drop. The rate before the last drop is check-pointed as the R_T. Since averaging ensures that $R_C < R_T$ throughout the FR phase, QCN is zero-delay stable (Section 2.3). In fact, Section 2.4 shows that averaging (or binary increase) is much more stable than simple additive increase in the face of large feedback delays.

Remark 2. The duration of Byte Counter cycles measured in seconds depends on the current sending rate, and can therefore become unacceptably large when R_C is small, jeopardizing the speed of bandwidth recovery (or responsiveness). Therefore, a *Timer* is also included in the standards implementation of QCN. The Byte Counter and Timer jointly determine rate increase times. The Timer is primarily used during transience, and since we are mainly interested in the steady state stability properties of QCN in this paper, we do not consider the Timer.

2.2 QCN Fluid Model

The fluid model presented below corresponds with the simplified version of QCN from the previous section. The derivation of the equations, for the most part, is part of the research literature [25]. The main difference is in our use of two variables, R_C and R_T, to represent source behavior. This is a necessary step, since although R_C and R_T are inter-dependent variables, neither can be derived from the other.

Consider a "dumb-bell topology" with N sources sharing a single link of capacity C. The RTT is assumed to be the same for all sources, equal to τ seconds. The source variables evolve according to the following differential equations:

$$\begin{aligned}\frac{dR_C}{dt} &= -G_d F_b(t-\tau) R_C(t) R_C(t-\tau) p_r(t-\tau) \\ &+ \left(\frac{R_T(t)-R_C(t)}{2}\right) \frac{R_C(t-\tau)p_r(t-\tau)}{(1-p_r(t-\tau))^{-100}-1}, \qquad (4)\\ \frac{dR_T}{dt} &= -\left(R_T(t)-R_C(t)\right) R_C(t-\tau) p_r(t-\tau) \\ &+ R_{AI} R_C(t-\tau) \frac{(1-p_r(t-\tau))^{500} p_r(t-\tau)}{(1-p_r(t-\tau))^{-100}-1}, \qquad (5)\end{aligned}$$

where $p_r(t)$ is the "reflection"—not sampling, see equation (8)—probability at the switch, and $F_b(t)$ is the congestion measure. These quantities are related to the queue size, $Q(\cdot)$, at the switch and evolve as follows:

$$\frac{dQ}{dt} = \begin{cases} NR_C(t) - C & \text{if } q(t) > 0, \\ \max\left(NR_C(t) - C, 0\right) & \text{if } q(t) = 0, \end{cases} \qquad (6)$$

$$F_b(t) = Q(t) - Q_{eq} + \frac{w}{Cp_s}(NR_C(t) - C), \qquad (7)$$

$$p_r(t) = p_s \mathbb{1}_{[F_b(t)>0]}, \qquad (8)$$

where p_s is the sampling probability.

Equations (4) and (5) each consist of a negative (rate decrease) term, and a positive (rate increase) term. Let us first consider the simpler negative terms. These terms model the decrease in R_T and R_C due to negative feedback signals, corresponding to (2) and (3). Observing that feedback signals to each source arrive at rate $R_C(t-\tau)p_r(t-\tau)$ explains the negative terms.

Now consider the positive term in (4). A rate increase occurs each time 100 packets are sent and no negative feedback message is received.[5] The change in $R_C(t)$ at each such event is given by:

$$\begin{aligned}\Delta R_C(t) &= \frac{R_C(t)+R_T(t)}{2} - R_C(t) \\ &= \frac{R_T(t)-R_C(t)}{2}. \qquad (9)\end{aligned}$$

[4] Recall that due to the absence of ACKs in Ethernet, packet transmission isn't self-clocked like in TCP.

[5] The actual algorithm sets increases to occur every 50 packets in the Active Increase phase. For simplicity we use the 100 packet increment in both the FR and AI phases in the model.

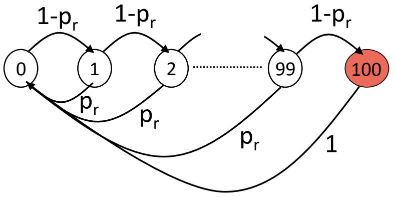


Figure 6: Markov chain corresponding to R_C rate increases.

To compute the rate at which such events occur, we consider the Markov Chain shown in Fig. 6. It is easy to see that if each packet is reflected with probability p_r, then the average number of packets that must be sent before an increase event occurs precisely equals the expected number of steps it takes to hit state 100 starting from state 0. This is easily computed:

$$\mathbb{E}_0(T_{100}) = \frac{(1-p_r)^{-100}-1}{p_r}.$$

Therefore, because packets from a source arrive at the switch with rate $R_C(t-\tau)$ at time t, the average time between increase events is:

$$\Delta T = \frac{(1-p_r(t-\tau))^{-100}-1}{R_C(t-\tau)p_r(t-\tau)}. \tag{10}$$

Dividing (9) by (10), we obtain the positive term in (4). The positive term in (5) is derived similarly.

Equations (6) and (7) are self-explanatory. Equation (8) captures the fact that sampled packets result in a reflected feedback message only when $F_b > 0$.[6]

Model validation. We have verified the fidelity of the model against packet-level simulations using the ns2 simulator [22]. An example run is shown in Fig. 7. As can be seen, the model and simulations match quite well.

2.3 Stability Analysis of Linearized Model

We now use the fluid model to analyze the stability of the QCN control loop in the presence of feedback delay. Define:

$$\eta(p_s) \triangleq \frac{p_s}{(1-p_s)^{-100}-1}, \quad \zeta(p_s) \triangleq \frac{(1-p_s)^{500}p_s}{(1-p_s)^{-100}-1}.$$

It is easily verified that the fluid model (4)–(8) has the following unique fixed point:

$$\begin{aligned} R_C^* &= \frac{C}{N}, \\ R_T^* &= \frac{C}{N} + \frac{\zeta(p_s)R_{AI}}{p_s}, \\ Q^* &= Q_{eq} + \frac{\eta(p_s)\zeta(p_s)NR_{AI}}{2p_s^2 G_d C}. \end{aligned}$$

We are interested in understanding if, and under what conditions, is this fixed point locally stable. The standard approach we undertake is to linearize the system around the fixed point, and use tools from linear control theory to study its stability.

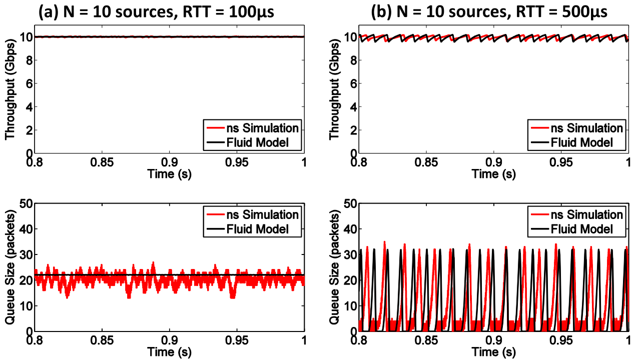


Figure 7: Comparison of QCN fluid model and ns2 simulation

The linearization of the differential equations is straightforward. Omitting the algebra, the linearized system describing the evolution of $\delta R_C(t) \triangleq R_C(t) - R_C^*$, $\delta R_T(t) \triangleq R_T(t) - R_T^*$, and $\delta Q(t) \triangleq Q(t) - Q^*$ is given by:

$$\frac{d\delta R_C}{dt} = -a_1\delta R_C(t) + a_2\delta R_T(t) - a_3\delta R_C(t-\tau) - a_4\delta Q(t-\tau), \tag{11}$$

$$\frac{d\delta R_T}{dt} = b\delta R_C(t) - b\delta R_T(t), \tag{12}$$

$$\frac{d\delta Q}{dt} = N\delta R_C(t), \tag{13}$$

where:

$$\begin{aligned} a_1 &= \frac{\eta(p_s)}{2}R_C^* + \frac{\eta(p_s)\zeta(p_s)}{2p_s}R_{AI}, \\ a_2 &= \frac{\eta(p_s)}{2}R_C^*, \quad a_3 = G_d w R_C^*, \quad a_4 = p_s G_d {R_C^*}^2, \\ b &= p_s R_C^*. \end{aligned}$$

We have obtained a linear time-delayed system and can now study its stability through its characteristic equation whose roots constitute the poles of the system. The characteristic equation of (11)–(13), derived in Appendix A, is given by:

$$1 + G(s) = 0, \tag{14}$$

where

$$G(s) = e^{-s\tau}\frac{a_3(s+b)(s+\gamma)}{s(s^2+\beta s+\alpha)}, \tag{15}$$

with $\gamma = Cp/w$, $\beta = b + a_1$, and $\alpha = b(a_1 - a_2)$.

Theorem 1. *Let*

$$\tau^* = \frac{1}{\omega^*}\left(\arctan(\frac{\omega^*}{b}) - \arctan(\frac{\omega^*}{\beta}) + \arctan(\frac{\omega^*}{\gamma})\right), \tag{16}$$

where

$$\omega^* = \sqrt{\frac{a_3^2}{2} + \sqrt{\frac{a_3^4}{4} + \gamma^2 a_3^2}}. \tag{17}$$

Then $\tau^* > 0$, *and the system* (11)–(13) *is stable for all* $\tau \le \tau^*$.

Proof. Using $\beta > b$, we have:

$$\arctan(\frac{\omega^*}{\beta}) < \arctan(\frac{\omega^*}{b}),$$

[6] It must be noted that when $p_r = 0$, the positive increase terms in (4) and (5) are to be interpreted as the resulting limits as $p_r \to 0$.

which implies $\tau^* > 0$. The proof of stability follows by applying the Bode stability criterion [11] to G(s). Define

$$r(\omega) = |G(j\omega)|, \quad \theta(\omega) = -\angle G(j\omega),$$

so that $G(j\omega) = r(\omega)e^{-j\theta(\omega)}$. We upper bound $r(\omega)$ as follows:

$$\begin{aligned} r(\omega)^2 &= \frac{a_3^2(\omega^2+b^2)(\omega^2+\gamma^2)}{\omega^2((\omega^2-\alpha)^2+\beta^2\omega^2)}, \\ &< \frac{a_3^2(\omega^2+b^2)(\omega^2+\gamma^2)}{\omega^4(\omega^2+\beta^2-2\alpha)}, \\ &< \frac{a_3^2(\omega^2+\gamma^2)}{\omega^4}. \end{aligned} \tag{18}$$

The last inequality holds because $\beta^2 - 2\alpha > b^2$, which is easily checked by plugging in $\beta = b + a_1$, $\alpha = b(a_1 - a_2)$. Now with ω^* given by (17), the bound (18) implies $r(\omega^*) < 1$. In particular, the 0-dB crossover frequency, ω_c, at which $r(\omega_c) = 1$ occurs for some $\omega_c < \omega^*$. Hence, the Bode stability criterion implies that if $\theta(\omega) < \pi$ for all $0 \le \omega < \omega^*$, the system is stable. But for $0 \le \omega < \omega^*$, using $\alpha > 0$ we have:

$$\begin{aligned} \theta(\omega) &= \pi + \omega\tau + \arctan(\frac{\omega^2-\alpha}{\beta\omega}) - \arctan(\frac{\omega}{b}) - \arctan(\frac{\omega}{\gamma}), \\ &< \pi + \omega\tau + \arctan(\frac{\omega}{\beta}) - \arctan(\frac{\omega}{b}) - \arctan(\frac{\omega}{\gamma}), \\ &= \pi + \omega\tau - \arctan\left(\frac{(\beta-b)\omega}{\beta b+\omega^2}\right) - \arctan(\frac{\omega}{\gamma}), \\ &\le \pi + \omega\tau - \arctan\left(\frac{(\beta-b)\omega}{\beta b+\omega^{*2}}\right) - \arctan(\frac{\omega}{\gamma}), \end{aligned} \tag{19}$$

where in the last inequality, we use the fact that arctan(.) is an increasing function. Now let

$$\Psi(\omega) = \pi + \omega\tau - \arctan\left(\frac{(\beta-b)\omega}{\beta b+\omega^{*2}}\right) - \arctan(\frac{\omega}{\gamma}).$$

Note that $\Psi(0) = \pi$ and for $\tau \le \tau^*$, $\Psi(\omega^*) \le \pi$. Moreover, since $\arctan(x)$ is concave for $x \ge 0$, $\Psi(\omega)$ is convex on $\omega \in [0, \omega^*]$. Therefore $\Psi(\omega) \le \pi$ for all $0 \le \omega \le \omega^*$, and (19) implies $\theta(\omega) < \pi$, completing the proof. □

Corollary 1 (Zero-Delay Stabilty). *If $\tau = 0$, system* (11)*–*(13) *is stable .*

Proof. This follows because $\tau^* > 0$ in Theorem 1. □

Corollary 1 confirms the intuitive argument given for zero-delay stability of QCN in Remark 1 of Section 2.1.

2.4 Averaging in QCN

As we have seen, the QCN Reaction Point averages R_C and R_T during the Fast Recovery phase. We now use fluid models to show that this averaging improves the robustness of QCN to increasing lags in the control loop.

Consider a modified QCN RP algorithm, henceforth called QCN-AIMD, where Active Increase begins *immediately* following a rate decrease, i.e. there is no Fast Recovery (see Fig 8 for an illustration). In Active Increase, the QCN-AIMD RP increases its sending rate each time the Byte Counter counts out 100 packets:

$$R_C \leftarrow R_C + R_{AI}.$$

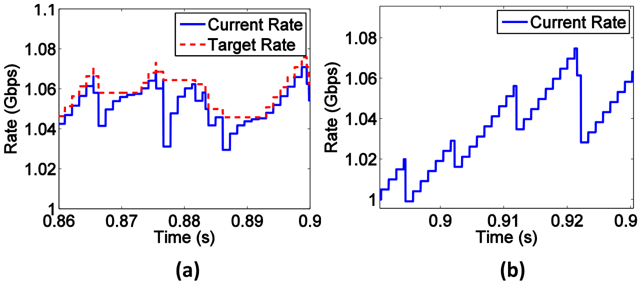


Figure 8: Rate evolutions for (a) QCN (b) QCN-AIMD

The CP algorithm remains the same as QCN.[7]

A fluid model for QCN-AIMD can be derived similarly as for QCN. In fact, since QCN-AIMD has no R_T variable, we only need to change the R_C equation (4) to the following:

$$\begin{aligned} \frac{dR_C}{dt} &= -G_d F_b(t-\tau) R_C(t) R_C(t-\tau) p_r(t-\tau) \\ &+ R_{AI} \frac{R_C(t-\tau)p_r(t-\tau)}{(1-p_r(t-\tau))^{-100}-1}. \end{aligned} \tag{20}$$

This, along with (6)–(8) constitute the QCN-AIMD fluid model. We can now linearize the QCN-AIMD model around its fixed point, and analyze its stability. This is done in Appendix B, where the following Theorem is proven:

Theorem 2. *Let*

$$\hat{\tau} = \frac{1}{\hat{\omega}}\left(\arctan(\frac{\hat{\omega}}{\gamma}) + \arctan(\frac{\hat{a}}{\hat{\omega}})\right), \tag{21}$$

where

$$\hat{\omega} = \sqrt{\frac{a_3^2-\hat{a}^2}{2} + \sqrt{\frac{(a_3^2-\hat{a}^2)^2}{4} + \gamma^2 a_3^2}}. \tag{22}$$

Here a_3 and γ are the same constants found in the QCN model, and $\hat{a} = \eta(p_s)R_{AI}$. The linearized QCN-AIMD model (35)*–*(36) *is stable if and only if $\tau < \hat{\tau}$.*

Theorems 1 and 2 provide the largest feedback delay for which the linear models of QCN and QCN-AIMD control loops retain stability. The following theorem compares these two and proves that under mild conditions, the QCN system—with its use of averaging—remains stable for larger lags.

Theorem 3. *Let τ^* and $\hat{\tau}$ be given by* (16) *and* (21) *respectively. If*

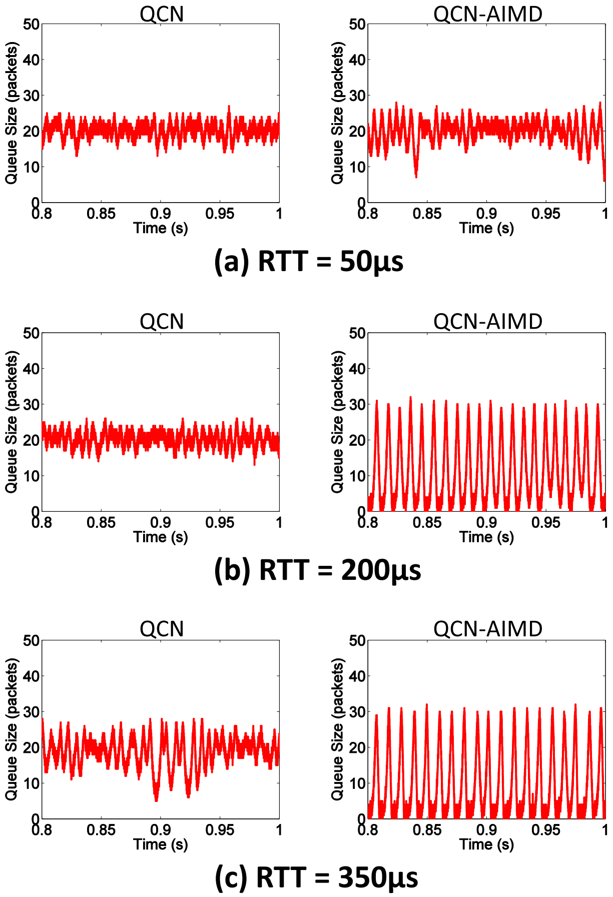
$$\frac{R_{AI}}{C}\max\left(\frac{\eta(p_s)^2/p_s}{G_d}, \frac{2\eta(p_s)+4p_s}{G_d}, \frac{\eta(p_s)w}{p_s}\right) < 0.1, \tag{23}$$

$$\frac{NR_{AI}}{C} < 0.2, \tag{24}$$

then $\tau^* > \hat{\tau}$.

Proof. See Appendix C. □

[7] QCN-AIMD and QCN are analogous to TCP and BIC-TCP respectively. However, an important distinction is that QCN-AIMD and QCN get multi-bit feedback from the network, allowing them to cut their rates by different factors corresponding to the amount of congestion.



(a) RTT = 50μs

(b) RTT = 200μs

(c) RTT = 350μs

Figure 9: Queue size for QCN and AIMD-QCN with RTT of (a) 50μs (b) 200μs (c) 350μs. $N = 10$ sources share a single 10Gbps bottleneck. The desired operating point at the switch buffer, Q_{eq}, is set to 22 packets.

Remark 3. The required conditions of Theorem 3 are easily satisfied in practice. For instance, for the baseline QCN parameters $C = 10$Gbps, $R_{AI} = 5$Mbps, $G_d = 1/128$, $w = 2$, $p_s = 0.01$, (23) is satisfied and (24) is equivalent to $N < 400$, which is far more than the number of sources typically active on each path in the data center [2].

2.5 Simulations

We have verified the theoretical predictions of the previous sections using ns2 simulations. We briefly present a representative example.

We compare the stability of QCN-AIMD and QCN as RTT increases. Fig. 9 shows the queue size for the two schemes when 10 sources share a single 10Gbps bottleneck link. Note that according to (16) and (21), the *linearized* QCN and QCN-AIMD control loops are stable for RTTs less than $\tau^* = 249\mu$s and $\hat{\tau} = 189\mu$s respectively.

As shown, when the RTT is small (50μs), both schemes are able to keep the queue stable around Q_{eq}. But when RTT is increased to 200μs, QCN-AIMD can no longer control the oscillations and the queue underflows, while the queue size for QCN continues to be stable as predicted. Even with RTT equal to 350μs, which is beyond the stability margin, τ^*, of the linearized model, QCN continues to gracefully keep the queue occupancy closely hovering around 22 packets (albeit with an increase in the amplitude of oscillations). It is only after the RTT increases beyond 500μs that the queue size with QCN begins to underflow.

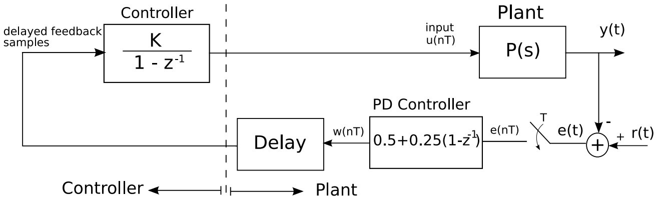


Figure 10: Equivalent PD control scheme to Averaging Principle.

In the next section, we formally define the Averaging Principle for a generic sampled control system, and show that averaging improves the robustness of control loops to increasing lags in much more general settings.

3. THE AVERAGING PRINCIPLE

3.1 Analysis

Recall the sampled control system in Fig. 1, and the basic form of AP (see Fig. 2) composed of periodic *feedback-induced changes* given by:

$$I_T \leftarrow I_C,$$
$$I_C \leftarrow I_C + Ke(nT),$$

and *averaging changes* at the midpoints of the sampling periods:

$$I_C \leftarrow \frac{I_C + I_T}{2}.$$

Note: In the generic setting of Fig. 2, the feedback signal can take both positive and negative values. However, in some cases of interest, the feedback might be restricted to negative values only; QCN is such an example. The definition of AP is the same in either case: following every feedback-induced change, make (one or more) averaging changes.

We now address the question: Why does the AP improve the stability of the basic controller? To anticipate the answer, we find the AP controller behaves like a PD controller—which is well-known to improve the stability of control loops [11]—without explicitly computing a derivative. This last feature is very important since it avoids the switch having to compute derivatives, which can be very cumbersome to do in hardware.

Before proving the claimed equivalence of the AP and PD controllers, we demonstrate it using our example. Consider the PD control scheme of Fig. 10. Here the error samples are not directly fed back. Instead, the feedback samples are computed as:

$$\begin{aligned} w(nT) &= \frac{1}{2}e(nT) + \frac{1}{4}(e(nT) - e((n-1)T)), \\ &\approx \frac{1}{2}e(nT) + \frac{T}{4}\frac{d}{dt}e(nT). \end{aligned}$$

Here, $w(nT)$ is a weighted sum of two terms: a *proportional* term, and a (discrete) *derivative* term. Hence this

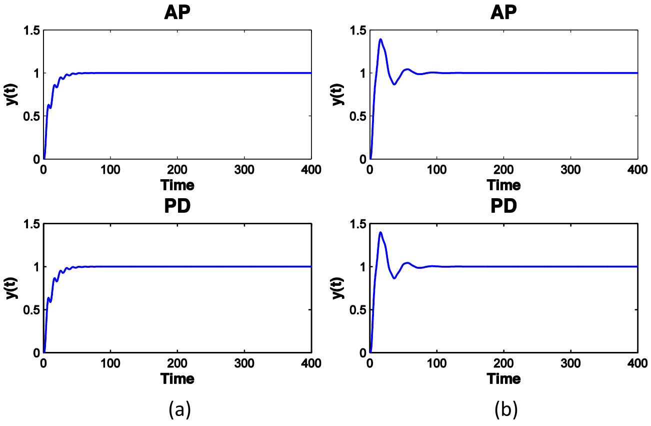


Figure 11: Step response for AP and PD controllers, with (a) zero delay (b) $\tau = 8$s delay.

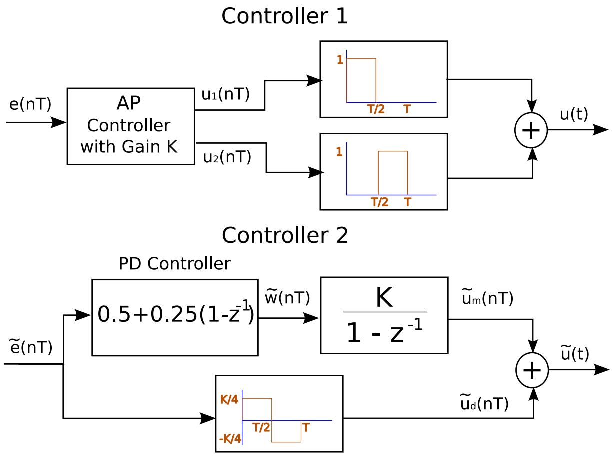


Figure 12: Two equivalent controllers.

is discrete-time PD control. The step response of this control loop is compared to the output of the AP controller in Fig. 11. As shown in the figure, the outputs of the AP and PD controllers are essentially identical.

Formally, consider the two controllers shown in Fig. 12. Both controllers map a sequence of error samples to an input signal driving the plant. Controller 1 is the AP controller, and Controller 2 is (essentially) the PD controller. The input-output relationships of the two controllers are given by the following equations:

Controller 1:

$$u_1(nT) = u_2((n-1)T) + Ke(nT)$$
$$u_2(nT) = \frac{u_1(nT) + u_2((n-1)T)}{2}$$

$$u(t) = \begin{cases} u_1(nT) & nT \le t < nT + \frac{T}{2} \\ u_2(nT) & nT + \frac{T}{2} \le t < nT + T \end{cases}$$

Controller 2:

$$\tilde{u}_m(nT) = \tilde{u}_m((n-1)T) + K\tilde{w}(nT) \qquad (25)$$
$$\tilde{w}(nT) = \frac{1}{2}\tilde{e}(nT) + \frac{1}{4}(\tilde{e}(nT) - \tilde{e}((n-1)T)) \qquad (26)$$

$$\tilde{u}_d(t) = \begin{cases} \frac{K}{4}\tilde{e}(nT) & nT \le t < nT + \frac{T}{2} \\ -\frac{K}{4}\tilde{e}(nT) & nT + \frac{T}{2} \le t < nT + T \end{cases} \qquad (27)$$

$$\tilde{u}(t) = \tilde{u}_m(t) + \tilde{u}_d(t) \qquad (28)$$

It should be noted $\tilde{u}_m(t) = \tilde{u}_m([\frac{t}{T}]T)$ in (28). For simplicity, we will not explicitly point out this conversion between discrete and continuous time signals each time.

Theorem 4. *Controllers 1 and 2 are algebraically equivalent; i.e., if they are given the same input sequences $e(nT) = \tilde{e}(nT)$ for all $n \ge 0$, and have the same initial condition $u(0) = \tilde{u}(0)$, then $u(t) = \tilde{u}(t)$ for all $t \ge 0$.*

Proof. For $n \ge 0$, let $u_m(nT) = (u_1(nT) + u_2(nT))/2$. From the definition of Controller 1, we get that

$$u_m(nT) = u_2((n-1)T) + \frac{3K}{4}e(nT), \qquad (29)$$

and, in turn, that

$$u_1(nT) = u_m(nT) + \frac{K}{4}e(nT), \qquad (30)$$
$$u_2(nT) = u_m(nT) - \frac{K}{4}e(nT). \qquad (31)$$

Comparing (30) and (31) with (27) and (28), it suffices to establish: $u_m(nT) = \tilde{u}_m(nT)$ for all $n \ge 0$. We do this by showing that $u_m(\cdot)$ and $\tilde{u}_m(\cdot)$ satisfy the same recursion.

Equations (29) and (31) give

$$\begin{aligned} u_m(nT) &= u_m((n-1)T) - \frac{K}{4}e((n-1)T) + \frac{3K}{4}e(nT) \\ &= u_m((n-1)T) + Kw(nT), \end{aligned} \qquad (32)$$

where

$$w(nT) = \frac{1}{2}e(nT) + \frac{1}{4}(e(nT) - e((n-1)T)). \qquad (33)$$

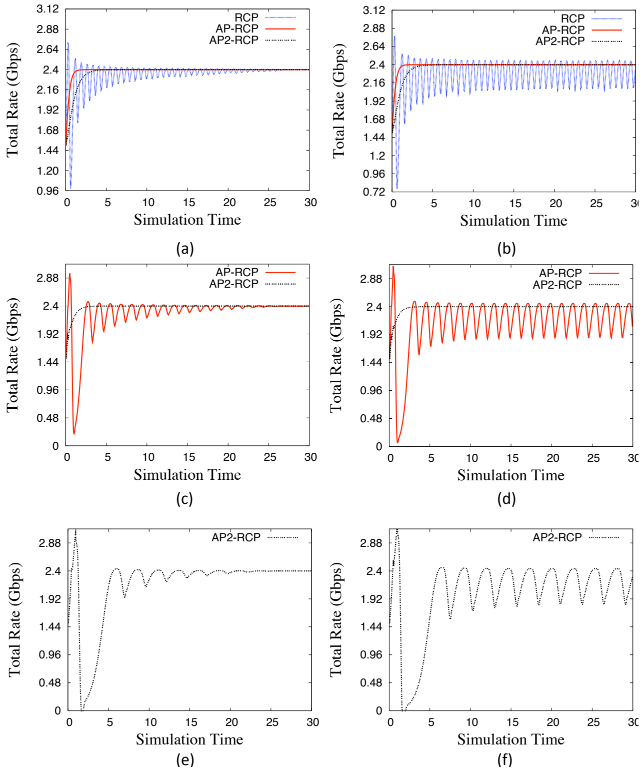
The recursion defined by (32) and (33) for $u_m(\cdot)$ in terms of $e(\cdot)$, is identical to the recursion defined by (25) and (26) for $\tilde{u}_m(\cdot)$ in terms of $\tilde{e}(\cdot)$. But, by hypothesis, $e(nT) = \tilde{e}(nT)$ for all $n \ge 0$ and $u_m(0) = \tilde{u}_m(0)$, and the proof is complete. □

Theorem 4 states that the AP controller is exactly equivalent to Controller 2. Since the effect of $\tilde{u}_d(t)$ is typically negligible (see below), we conclude that the AP controller is essentially equivalent to the top path of Controller 2. But, this is the same as the PD controller in Fig. 10.

The effect of $\tilde{u}_d(t)$ is negligible if the sampling time T is small compared to the rise time of the plant.[8] Intuitively, such a plant only reacts to the *mean* value of the input and not to the detailed variations of the input within a sampling interval. Since the mean of $\tilde{u}_d(t)$ in a sampling interval is zero, it doesn't affect the output of the plant. Thus, the AP and PD controllers are essentially identical, as seen in Fig. 11.

Remark 4. Extensions of the basic AP scheme are possible and are amenable to a complete theoretical analysis. These

[8] This is a fairly standard design criterion in digital control systems. For example, a simple rule of thumb is that the sampling time should be smaller than 10 % of the dominant time constant.



(a) (b) (c) (d) (e) (f)

Figure 13: Total sending rate for RCP, AP-RCP and AP2-RCP with RTT of (a) 110ms (b) 115ms (c) 240ms (d) 250ms (e) 480ms (f) 490ms.

include: averaging by factors other than 1/2, for example

$$I_C \leftarrow (1-\alpha)I_C + \alpha I_T \qquad \text{(for } 0<\alpha<1\text{)},$$

applying the averaging changes at points other than the midpoint, applying averaging more than once, etc.

Remark 5. The common tradeoff between stability and responsiveness exists with the AP as well: an AP-enhanced control scheme is more stable than the original, but it is also more sluggish. In practice, various techniques are used to improve the transient behavior of a system. Typically, these take the form of a temporary deviation from the normal controller behavior during times of transience. Some examples of this are BIC-TCP's *Fast Convergence* [28] and QCN's *Timer*, *Extra Fast Recovery*, and *Target Rate Reduction* [2].

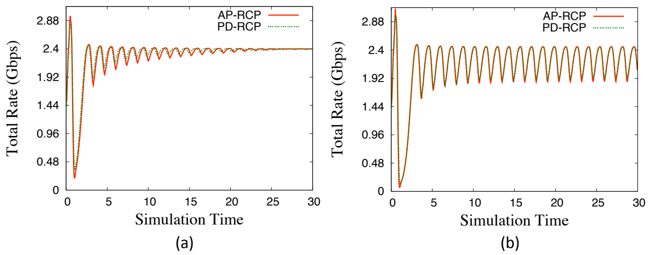
3.2 The AP applied to another congestion control algorithm: RCP

As mentioned, the AP is a general technique and can be applied to any control loop. We now apply it to the congestion control scheme RCP [8]. In RCP, each router offers a rate, $R(t)$, to each flow passing through it. $R(t)$ is updated every T seconds:

$$R(t) = R(t-T)(1+\frac{T}{d_0}F_b(t)), \tag{34}$$

where d_0 is a moving average of the RTT measured across all packets, and:

$$F_b(t) = \frac{\alpha(C-y(t)) - \beta\frac{q(t)}{d_0}}{C} \approx \frac{-\alpha\dot{q}(t) - \beta\frac{q(t)}{d_0}}{C}.$$



(a) (b)

Figure 14: Comparison of AP-RCP and PD-RCP for (a) RTT = 240ms (b) 250ms.

Here $y(t)$ is the measured input traffic rate during the last update interval, $q(t)$ is the instantaneous buffer size, C is the link capacity, and α and β are non-negative constants. We refer the reader to [8] for details of RCP.

As seen in (34), RCP automatically adjusts the loop gain, d_0, based on the average RTT in the control loop. This automatic gain adjustment stabilizes RCP as RTT varies. To demonstrate the effect of the AP on RCP's stability, we disable the automatic gain adjustment of RCP; i.e., we consider the case where d_0 is held constant at the router while the actual RTT increases.[9]

In what follows, we compare RCP with two AP-enhanced versions of the algorithm: AP-RCP and AP2-RCP. AP-RCP is just the application of the basic form of AP to RCP. AP2-TCP makes two averaging changes within each update interval: at $T/3$ and $2T/3$ after the start of interval.

Stability. Using ns2 [22], we simulate RCP controlling 10 long-lived flows passing through a 2.4Gbps link. The RCP parameters are set to $\alpha = 0.1$, $\beta = 1$, the sampling period $T = 50$ms, and d_0 is fixed at 20ms.

The RTT is varied and the results are shown in Fig. 13. When the RTT is smaller than 110ms, both RCP and its AP modifications are stable. RCP becomes unstable when the RTT increases to 115ms, but AP-RCP and AP2-RCP both continue to be stable for RTTs up to 240ms. AP-RCP becomes unstable at an RTT of 250ms, but AP2-RCP remains stable even when the RTT equals 480 ms.

PD Equivalence to AP. We now apply a PD controller to RCP (PD-RCP), and compare it with AP-RCP. The rate update equation for PD-RCP is given by:

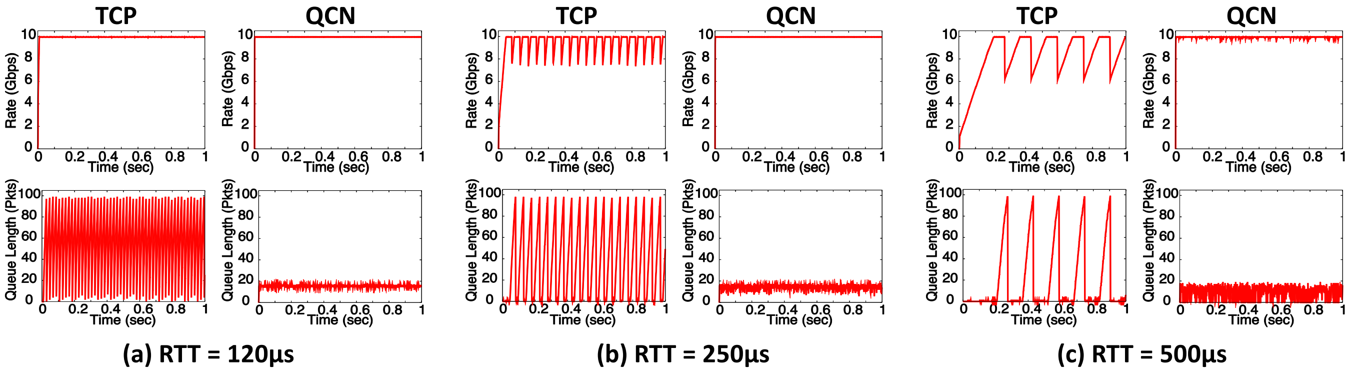
$$R(t) = R(t-T)\left(1+\frac{T}{d_0}\left(\frac{1}{2}F_b(t)+\frac{1}{4}(F_b(t)-F_b(t-T))\right)\right).$$

Fig. 14 shows that AP-RCP and PD-RCP are both closely matched: both are stable at RTT = 240ms, unstable at RTT = 250ms, and exhibit very similar queue/rate oscillations.

4. BUFFER SIZING

The performance of congestion control algorithms crucially depends on the size of buffers at switches: if the buffers are small relative to the bandwidth–delay product of the flows they support, buffer occupancies will oscillate and can cause under-utilization of links. The TCP buffer

[9] It must be noted that this is only intended as a demonstration of how the AP can improve stability without gain adjustments; we are not suggesting a change to the RCP algorithm.



(a) RTT = 120µs

(b) RTT = 250µs

(c) RTT = 500µs

Figure 15: TCP vs. QCN utilization and queue occupancy as RTT increases. Switch buffer is 100 packets deep.

sizing "rule of thumb" states that a single TCP flow requires a bandwidth-delay product ($C \times RTT$) amount of buffering to fully utilize a link of capacity C [26]. For example, a 10Gbps network with a 500μs round-trip time (RTT) necessitates that switches have 625 Kbytes of buffering per priority per port.

On the other hand, the amount of buffering that *can be provided* in a data center switch is limited by cost (at 10Gbps line rates the low access time allowed per packet necessitate that expensive SRAMs be used to build switch buffers) and latency (having shallow buffers bounds the worst-case latency of a packet through a switch and, hence, end-to-end). Moreover, since these buffers are usually placed on-chip (to ensure low-latency), they will consume a large die area and dissipate a lot of heat. Thus, data center Ethernet switches typically have small buffers, usually around 2-10MB for 24-48 ports.

The above discussion illustrates there can be a mismatch in data centers between the amount of buffering that is needed and that which can be provided, and this is only likely to worsen as link speeds increase. (The Ethernet industry road map calls for the deployment of 40Gbps switched Ethernet in the near future.)

One way the buffering requirement at a switch can be smaller is if several flows are simultaneously traversing it: The paper by Appenzeller et. al. [4] shows that when N flows share the link, the required buffer size for TCP goes down by a factor of $\sqrt{N}$ ($C \times RTT/\sqrt{N}$). Essentially, as [4] explains, the variance of the total flow arrival rate is reduced due to statistical multiplexing, and this leads to a reduction in the required queue size. More precisely, if N flows are multiplexed, the variance of their aggregate sending rate equals

$$Var(\sum_{i=1}^{N} R_i) = \frac{Var(\widetilde{R})}{\sqrt{N}},$$

where $Var(\widetilde{R})$ is the variance in the sending rate when only 1 flow traverses the link. Note that the mean value of each R_i is C/N, but the mean value of $\tilde{R}$ equals C.

However, the number of simultaneously active flows at a link in a data center network is very small, typically fewer than 10. Indeed, great pains are taken to ensure that flows are load balanced onto multiple paths so that they may fully utilize network bandwidth [16, 1, 12]. Therefore, it is hard to obtain the benefit of statistical multiplexing in the data center environment and it appears that large buffers will be required.

Perhaps not. The amount of buffering a single source needs to occupy the link is dependent on the variance of the sending rate which, in turn, depends on the congestion control algorithm. Table 1 shows that over a 10Gbps link the standard deviation in the sending rate of a QCN source is significantly smaller than that of a TCP source as the RTT increases. QCN reduces the variance of the sending rate in two ways: (i) using a multi-bit feedback allows QCN sources to cut their sending rates by factors smaller than 1/2 (as small as 1/128), and (ii) employing the AP reduces the sending rate variance via averaging changes. Note that BIC-TCP also derives the benefit of (ii) (see also [6]), but is constrained to cut window sizes by a constant factor of 0.125 during congestion. So its variance, while smaller than TCP's, will be larger than QCN's.

Fig. 15 compares the throughput and queue occupancy of a TCP and a QCN flow traversing a 10Gbps link. The buffer size at the bottleneck switch is 150KBytes, which exactly equals the BDP for an RTT of 120μs. Hence, as seen in part (a), both algorithms fully utilize the link. At higher RTTs (250μs in part (b) and 500μs in part (c)), large queue oscillations cause TCP to lose throughput, whereas QCN remains stable.

Table 1: Standard deviation of sending rate

RTT	120μs	250μs	500 μs
TCP	265 Mbps	783 Mbps	1250 Mbps
QCN	14 Mbps	33 Mbps	95 Mbps

5. CONCLUSION

Data center networks present new opportunities and challenges for developing new networking technology. In this paper we studied one such technology: the L2 congestion control algorithm, QCN, developed for the IEEE 802.1Qau standard. We described the salient features of QCN and developed a high-fidelity fluid model corresponding to it. We determined the stability margins of the QCN algorithm using a linearization of the fluid model. We articulated the Averaging Principle (AP), and showed that it is the underlying reason for QCN's good stability properties. The AP applies to general control systems, not just congestion control systems. This aspect is worth pursuing further. The AP controller was analyzed from a control-theoretic point-of-view and found to be equivalent to a PD controller in a strong algebraic sense. Finally, we showed that QCN's use of the AP and of a multi-bit feedback signal allows it to use much smaller buffers when compared to TCP.

Acknowledgments

Mohammad Alizadeh is supported by a Caroline and Fabian Pease Stanford Graduate Fellowship. We thank our shepherd Yuliy Baryshnikov and the anonymous reviewers whose comments helped us improve the paper.

6. REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 63–74, New York, NY, USA, 2008. ACM.

[2] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman. Data center transport mechanisms: Congestion control theory and IEEE standardization. In *Allerton*, 2008.

[3] Amazon Web Services. `http://aws.amazon.com`.

[4] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. *SIGCOMM Comput. Commun. Rev.*, 34(4):281–292, 2004.

[5] S. Athuraliya, S. Low, V. Li, and Q. Yin. REM: active queue management. *Network, IEEE*, 15(3):48 –53, May 2001.

[6] H. Cai, D. Y. Eun, S. Ha, I. Rhee, and L. Xu. Stochastic Ordering for Internet Congestion Control and its Applications. In *INFOCOM*, May 2007.

[7] Data Center Bridging Task Group. `http://www.ieee802.org/1/pages/dcbridges.html`.

[8] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor sharing flows in the internet. In *IWQoS*, pages 271–285, 2005.

[9] Fiber Channel standard. `http://www.t11.org/ftp/t11/pub/fc/bb-5/09-056v5.pdf`.

[10] S. Floyd. HighSpeed TCP for Large Congestion Windows, 2003.

[11] G. F. Franklin, D. J. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[12] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 51–62, New York, NY, USA, 2009. ACM.

[13] S. Ha, I. Rhee, and L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, 2008.

[14] B. Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.

[15] C. Hollot, V. Misra, D. Towsley, and W.-B. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *INFOCOM 2001*, volume 3, pages 1726 –1734 vol.3, 2001.

[16] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm, 2000.

[17] QCN pseudo code. `http://www.ieee802.org/1/files/public/docs2008/au-rong-qcn-serial-hai-pseudo-code\%20rev2.0.pdf`.

[18] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 89–102, New York, NY, USA, 2002. ACM.

[19] T. Kelly. Scalable TCP: improving performance in highspeed wide area networks. *SIGCOMM Comput. Commun. Rev.*, 33(2):83–91, 2003.

[20] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle. Dynamics of TCP/RED and a Scalable Control. In *IN PROCEEDINGS OF IEEE INFOCOM 2002*, pages 239–248, 2002.

[21] Microsoft Corporation. An Overview of Windows Azure. `http://www.microsoft.com/downloads/en/details.aspx?displaylang=en&FamilyID=96d08ded-bbb9-450b-b180-b9d1f04c3b7f`.

[22] The Network Simulator NS-2. `http://www.isi.edu/nsnam/ns/`.

[23] IEEE 802.1Qau. `http://www.ieee802.org/1/pages/802.1au.html`.

[24] IEEE 802.1Qbb. `http://www.ieee802.org/1/pages/802.1bb.html`.

[25] R. Srikant. *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*. SpringerVerlag, 2004.

[26] C. Villamizar and C. Song. High performance TCP in ANSNET. *SIGCOMM Comput. Commun. Rev.*, 24(5):45–60, 1994.

[27] D. X. Wei, C. Jin, S. H. Low, and S. Hegde. FAST TCP: motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Netw.*, 14(6):1246–1259, 2006.

[28] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. In *INFOCOM*, 2004.

APPENDIX

A. CHARACTERISTIC EQUATION OF LINEARIZED QCN FLUID MODEL

Taking the Laplace transform of (11)–(13), we have:

$$\begin{aligned} sR_C(s) - \delta R_C(0) &= -a_1 R_C(s) + a_2 R_T(s) \\ &\quad - e^{-s\tau}\left(a_3 R_C(s) + a_4 Q(s)\right), \\ sR_T(s) - \delta R_T(0) &= bR_C(s) - bR_T(s), \\ sQ(s) - \delta Q_0 &= NR_C(s). \end{aligned}$$

Solving for $Q(s)$ and $R_T(s)$ in terms of $R_C(s)$ and plugging into the first equation, we have:

$$\begin{aligned} &\Big(s + a_1 - \frac{a_2 b}{s+b} + e^{-s\tau}(a_3 + \frac{Na_4}{s})\Big) R_C(s) \\ &\qquad = \delta R_C(0) + \frac{a_2 \delta R_T(0)}{s+b} - e^{-s\tau}\frac{a_4 \delta Q_0}{s}. \end{aligned}$$

Multiplying both sides by $s(s+b)$, we see that the poles of the system are the given by the roots of:

$$s^2 + \beta s + \alpha + e^{-s\tau} a_3 (s+b)(s+\gamma) = 0,$$

where $\beta = b + a_1$, and $\alpha = b(a_1 - a_2)$, and $\gamma = Na_4/a_3 = Cp/w$, which can equivalently be written as (14), (15).

B. STABILITY ANALYSIS OF THE QCN-AIMD FLUID MODEL

Recall the QCN-AIMD fluid model given by (20) and (6)–(8). We linearize the system around it unique fixed point:

$$\hat{R_C} = \frac{C}{N}, \quad \hat{Q} = Q_{eq} + \frac{\eta(p_s) N R_{AI}}{p_s G_d C}.$$

This leads to the following linear time-delay system:

$$\frac{d\delta R_C}{dt} = -\hat{a}\delta R_C(t) - a_3\delta R_C(t-\tau) - a_4\delta Q(t-\tau), \quad (35)$$

$$\frac{d\delta Q}{dt} = N\delta R_C(t). \quad (36)$$

The constants $a_3 = G_d w\hat{R_C}$, $a_4 = p_s G_d \hat{R_C}^2$, are the same as in the QCN linear model, and $\hat{a} = \eta(p_s)R_{AI}$. After taking the Laplace transform and rearranging, the characteristic equation of (35)–(36) is found to be:

$$1 + \hat{G}(s) = 0,$$

where

$$\hat{G}(s) = e^{-s\tau}\frac{a_3(s+\gamma)}{s(s+\hat{a})}.$$

To prove Theorem 2, we proceed by applying the Bode stability criterion to $\hat{G}(s)$. Let $\hat{G}(j\omega) = \hat{r}(\omega)\exp(-j\hat{\theta}(\omega))$. The 0-dB crossover frequency is found by solving the equation:

$$\hat{r}(\omega) = \frac{a_3\sqrt{\omega^2+\gamma^2}}{\omega\sqrt{\omega^2+\hat{a}^2}} = 1.$$

After squaring both sides, we have a quadratic equation in ω^2 which is easily solved, and we find that $\hat{\omega}$, given by (22), is the 0-dB crossover frequency. Therefore, we require $\hat{\theta}(\hat{\omega}) < \pi$ for the system to be stable. But:

$$\begin{aligned}\hat{\theta}(\omega) &= \frac{\pi}{2} + \omega\tau + \arctan(\frac{\omega}{\hat{a}}) - \arctan(\frac{\omega}{\gamma}),\\ &= \pi + \omega\tau - \arctan(\frac{\hat{a}}{\omega}) - \arctan(\frac{\omega}{\gamma}).\end{aligned}$$

Therefore, $\hat{\theta}(\hat{\omega}) < \pi$ is equivalent to $\tau < \hat{\tau}$, completing the proof. □

C. PROOF OF THEOREM 3

To prove $\tau^* > \tau$, we need to show:

$$\begin{aligned}&\hat{\omega}\left(\arctan(\frac{\omega^*}{b}) - \arctan(\frac{\omega^*}{\beta}) + \arctan(\frac{\omega^*}{\gamma})\right)\\ &\quad > \omega^*\left(\arctan(\frac{\hat{\omega}}{\gamma}) + \arctan(\frac{\hat{a}}{\hat{\omega}})\right), \quad (37)\end{aligned}$$

where ω^* and $\hat{\omega}$ are given by (17) and (22) respectively. We begin with two simple Lemmas.

Lemma 1. *If* (23) *and* (24) *are satisfied, then:*

$$\max\left(\frac{\hat{a}}{\gamma}, \frac{2\hat{a}b\beta}{\gamma a_3 a_1}\right) < 0.1, \quad (38)$$

$$\frac{\hat{a}}{a_1} < 0.4, \quad (39)$$

$$\frac{\hat{a}^2}{\gamma a_3} < 0.02. \quad (40)$$

Proof. By direct substitution for $\hat{a}$, γ, a_3, b, β, and using $a_1 > \eta(p_s)C/(2N)$, it is easy to verify that (38) results from (23), and (39) results from (24). For (40), note that:

$$\frac{\hat{a}^2}{\gamma a_3} = \frac{NR_{AI}}{C} \times \frac{\eta(p_s)^2 R_{AI}}{p_s G_d C} < 0.2 \times 0.1 = 0.02 \quad \square$$

Lemma 2. *Let* ω^* *and* $\hat{\omega}$ *be given by* (17) *and* (22) *respectively. If* (23) *and* (24) *are satisfied, then:*

$$1 < \frac{\omega^*}{\hat{\omega}} < 1 + \epsilon,$$

where

$$\epsilon = \frac{3\hat{a}^2}{4\hat{\omega}^2} < 0.015.$$

Proof. From (22) we have:

$$\begin{aligned}\hat{\omega}^2 &= \frac{a_3^2 - \hat{a}^2}{2} + \sqrt{\frac{a_3^2}{4} + \gamma^2 a_3^2 + \frac{\hat{a}^4 - 2\hat{a}^2 a_3^2}{4}},\\ &> \frac{a_3^2 - \hat{a}^2}{2} + \sqrt{\frac{a_3^4}{4} + \gamma^2 a_3^2}\sqrt{1 - \frac{2\hat{a}^2 a_3^2}{a_3^4 + 4\gamma^2 a_3^2}}.\end{aligned}$$

Therefore, using $\sqrt{1-x} > 1 - x$ for $0 < x < 1$, we have:

$$\begin{aligned}\hat{\omega}^2 &> \frac{a_3^2 - \hat{a}^2}{2} + \sqrt{\frac{a_3^4}{4} + \gamma^2 a_3^2}\left(1 - \frac{2\hat{a}^2 a_3^2}{a_3^4 + 4\gamma^2 a_3^2}\right),\\ &= \frac{a_3^2}{2} + \sqrt{\frac{a_3^4}{4} + \gamma^2 a_3^2} - \frac{\hat{a}^2 a_3^2}{\sqrt{a_3^4 + 4\gamma^2 a_3^2}} - \frac{\hat{a}^2}{2},\\ &> {\omega^*}^2 - \frac{3\hat{a}^2}{2}.\end{aligned}$$

Therefore, using $\sqrt{1+x} < 1 + x/2$:

$$\frac{\omega^*}{\hat{\omega}} < \sqrt{1 + \frac{3\hat{a}^2}{2\hat{\omega}^2}} < 1 + \frac{3\hat{a}^2}{4\hat{\omega}^2}.$$

Let $\epsilon \triangleq 3\hat{a}^2/(4\hat{\omega}^2)$. Since $\hat{\omega}^2 > \gamma a_3$, using (40), we have $\epsilon < 0.015$. □

We are now ready to prove (37). Using Lemma (2), it is enough to show:

$$\begin{aligned}&\arctan(\frac{\omega^*}{b}) - \arctan(\frac{\omega^*}{\beta}) + \arctan(\frac{\omega^*}{\gamma})\\ &\quad > (1+\epsilon)\left(\arctan(\frac{\hat{\omega}}{\gamma}) + \arctan(\frac{\hat{a}}{\hat{\omega}})\right),\end{aligned}$$

and since $\omega^* > \hat{\omega}$, it is enough to show:

$$\arctan(\frac{\omega^*}{b}) - \arctan(\frac{\omega^*}{\beta}) > \epsilon\arctan(\frac{\hat{\omega}}{\gamma}) + (1+\epsilon)\arctan(\frac{\hat{a}}{\hat{\omega}}).$$

However, using $\arctan(x) \le x$ for $x \ge 0$ and (38):

$$\epsilon\arctan(\frac{\hat{\omega}}{\gamma}) < \frac{\hat{a}^2}{\hat{\omega}^2}\left(\frac{\hat{\omega}}{\gamma}\right) < 0.1\left(\frac{\hat{a}}{\hat{\omega}}\right) < 0.2\arctan(\frac{\hat{a}}{\hat{\omega}}),$$

where the last inequality uses the fact: $x \le 2\arctan(x)$ for $0 \le x \le 1$. Therefore, it is enough to show:

$$\arctan(\frac{\omega^*}{b}) - \arctan(\frac{\omega^*}{\beta}) > 2\arctan(\frac{\hat{a}}{\hat{\omega}}).$$

After applying $\tan(\cdot)$ to both sides of this inequality and simplifying using $\omega^* > \omega$, we are left with:

$$a_1\hat{\omega}^2 - 2\hat{a}{\omega^*}^2 > a_1\hat{a}^2 + 2\hat{a}b\beta,$$

which can be further simplified using $\omega^* < (1+\epsilon)\hat{\omega}$, and $\hat{\omega} > \gamma a_3$ to arrive at:

$$1 > \frac{\hat{a}^2}{\gamma a_3} + \frac{2\hat{a}b\beta}{\gamma a_3 a_1} + \frac{2\hat{a}}{a_1}(1+\epsilon)^2.$$

This inequality holds because of (38)–(40) and $\epsilon < 0.015$. □