One More Bit Is Enough

Yong Xia* Lakshminarayanan Subramanian+ Ion Stoica++ Shivkumar Kalyanaraman**

* NEC Labs China + NYU ++ U.C.Berkeley ** RPI

ABSTRACT

Achieving efficient and fair bandwidth allocation while minimizing packet loss and bottleneck queue in high bandwidth-delay product networks has long been a daunting challenge. Existing end-to-end congestion control (*e.g.,* TCP) and traditional congestion notification schemes (*e.g.,* TCP+AQM/ECN) have significant limitations in achieving this goal. While the XCP protocol addresses this challenge, it requires multiple bits to encode the congestion-related information exchanged between routers and end-hosts. Unfortunately, there is no space in the IP header for these bits, and solving this problem involves a non-trivial and time-consuming standardization process.

In this paper, we design and implement a simple, low-complexity protocol, called Variable-structure congestion Control Protocol (VCP), that leverages only the existing two ECN bits for network congestion feedback, and yet achieves comparable performance to XCP, *i.e.,* high utilization, negligible packet loss rate, low persistent queue length, and reasonable fairness. On the downside, VCP converges significantly slower to a fair allocation than XCP. We evaluate the performance of VCP using extensive ns2 simulations over a wide range of network scenarios and find that it significantly outperforms many recently-proposed TCP variants, such as HSTCP, FAST, and CUBIC. To gain insight into the behavior of VCP, we analyze a simplified fluid model and prove its global stability for the case of a single bottleneck shared by synchronous flows with identical round-trip times.

1. INTRODUCTION

The Additive-Increase-Multiplicative-Decrease (AIMD) [10] congestion control algorithm employed by TCP [24] is known to be ill-suited for high Bandwidth-Delay Product (BDP) networks. With rapid advances in the deployment of very high bandwidth links in the Internet, the need for a viable replacement of TCP in such environments has become increasingly important.

Several research efforts have proposed different approaches for this problem, each with their own strengths and limitations. These can be broadly classified into two categories: *end-to-end* and *network feedback* based approaches. Pure end-to-end congestion control schemes such as HSTCP [14], FAST [30], STCP [35], BIC/CUBIC [64, 57], and HTCP [43], although being attractive short-term solutions (due to a lesser deployment barrier), may not be suitable for the long-term. For congestion control purpose, the end-to-end schemes artificially introduce packet loss or queuing delay, which should be avoided in the first place. Further, in high BDP networks, using loss and/or delay as the only congestion signal(s) poses fundamental limitations on achieving high utilization and fairness while maintaining low bottleneck queue length and minimizing congestion-induced packet loss rate. HSTCP illustrates the limitations of loss-based approaches in high bandwidth optical links with very low bit-error rates [14]. Similarly, it has been shown that delay-based approaches are highly sensitive to delay variations [7], a common case in today's Internet.

To address some of the limitations of end-to-end congestion control schemes, many researchers have proposed the use of *explicit* network feedback. However, while traditional *congestion notification* feedback schemes such as TCP+AQM/ECN proposals [17, 2, 41, 55] are successful in reducing the loss rate and the queue size in the network, they still fall short in achieving high utilization in high BDP networks [23, 47, 34]. XCP [34] addresses this problem by having routers estimate the fair rate and send this rate back to the senders. Congestion control schemes that use *explicit rate* feedback have also been proposed in the context of the ATM Available Bit Rate (ABR) service [39, 9, 32, 26, 33]. However, these schemes are hard to deploy in today's Internet as they require a non-trivial number of bits to encode the rate, bits which are not available in the IP header.

In this paper, we show that it is possible to approximate XCP's performance in high BDP networks by leveraging only the two ECN bits (already present in the IP header) to encode the congestion feedback. The crux of our algorithm, called Variable-structure congestion Control Protocol (VCP), is to dynamically adapt the congestion control policy as a function of the level of congestion in the network. With VCP, each router computes a *load factor* [26], and uses this factor to classify the level of congestion into three regions: low-load, high-load and overload [27]. The router encodes the level of congestion in the ECN bits. As with ECN, the receiver echoes the congestion information back to the sender via acknowledgement (ACK) packets. Based on the load region reported by the network, the sender uses

An earlier version of this paper was previously presented at SIGCOMM'05, August 2005, Philadelphia, PA.

one of the following policies: Multiplicative Increase (MI) in the low-load region, Additive Increase (AI) in the high-load region, and Multiplicative Decrease (MD) in the overload region. By using MI in the low-load region, flows can exponentially ramp up their bandwidth to improve network utilization quickly. Once high utilization is attained, AIMD provides long-term fairness amongst the competing flows.

Using extensive packet-level ns2 [50] simulations that cover a wide range of network scenarios, we show that VCP can approximate the performance of XCP by achieving high utilization, negligible packet drop rate, low persistent queue length and reasonable fairness. One limitation of VCP (as is the case for other end-host based approaches including TCP and its many variants) is that it converges significantly slower to a fair allocation than XCP.

To better understand VCP, we analyze its stability and fairness properties using a simplified fluid model that approximates VCP's behavior. For the case of a single bottleneck link shared by flows with identical round-trip delays, we prove that the model asymptotically achieves *global* stability independent of the link capacity, the feedback delay and the number of flows. For more general multiple-bottleneck topologies, we show that the equilibrium rate allocation of this model is max-min fair [4]. While this model may not accurately reflect VCP's dynamics, it does reinforce the stability and fairness properties that we observe in our simulations and provides a good theoretical grounding for VCP.

From a practical point of view VCP has two advantages. First, VCP does not require any modifications to the IP header since it can reuse the two ECN bits in a way that is compatible with the ECN proposal [55]. Second, it is a simple protocol with low algorithmic complexity. The complexity of VCP's end-host algorithm is similar to that of TCP. The router algorithm maintains no per-flow state, and it has very low computation complexity. We believe that these benefits largely offset VCP's limitation of having a much slower fairness convergence speed than XCP.

The rest of the paper is organized as follows. In Section 2, we describe the guidelines that motivate the design of VCP and in Section 3, we provide a detailed description of VCP. In Section 4, we evaluate the performance of VCP using extensive simulations and compare with many other recently-proposed TCP variants. In Section 5, we develop a fluid model that approximates VCP's behavior and characterize its stability, fairness and convergence properties, with the detailed proofs presented in our technical report [63]. Section 6 addresses concerns on the stability of VCP under heterogeneous delays and the influence of switching between MI, AI and MD on efficiency and fairness, and ???. We review related work in Section 7 and summarize our findings in Section 8.

2. FOUNDATIONS

In this section, we first review why XCP scales to high BDP networks while TCP+AQM does not. Then, we present two guidelines that form the basis of the VCP design.

2.1 Why XCP outperforms TCP+AQM?

There are two main reasons for why TCP does not scale to high BDP networks. First, packet loss is a *binary* congestion signal that conveys no information about the *degree* of congestion. Second, due to stability reasons, relying only on packet loss for congestion indication requires TCP to use a conservative window increment policy and an aggressive window decrement policy [24, 34]. In high BDP networks, every loss event forces a TCP flow to perform an MD, followed by the slow convergence of the AI algorithm to reach high utilization. Since the time for each individual AIMD epoch is proportional to the per-flow BDP, TCP flows remain in low utilization regions for prolonged periods of time thereby resulting in poor link utilization. Using AQM/ECN in conjunction with TCP does not solve this problem since the (one-bit) ECN feedback, similar to a packet loss, is not indicative of the degree of congestion either.

XCP addresses this problem by precisely measuring the fair share of a flow at a router and providing explicit rate feedback to end-hosts. One noteworthy aspect of XCP is the decoupling of efficiency control and fairness control *at each router*. XCP uses MIMD to control the flow aggregate and to converge exponentially fast to any available bandwidth and uses AIMD to fairly allocate the bandwidth among competing flows. As a consequence, XCP requires multiple bits in the packet header to carry bandwidth allocation information ($\Delta cwnd$) from network routers to end-hosts, and congestion window ($cwnd$) and Round-Trip Time (RTT) information (rtt) from the end-hosts to the network routers.

2.2 Design Guidelines for VCP

The main goal of our work is to develop a *simple* congestion control mechanism that can scale to high BDP networks. By "simple" we mean an AQM-style approach where routers merely provide feedback on the level of network congestion, and end-hosts perform congestion control actions using this feedback. Furthermore, to maintain the compatibility with the existing IP header format, we restrict ourselves to using only two bits to encode the congestion information. To address these challenges, our solution builds around two design guidelines:

#1, Decouple efficiency control & fairness control.

Like XCP, VCP decouples efficiency and fairness control. However, unlike XCP where routers run the efficiency and fairness control algorithms and then explicitly communicate the fair rate to end-hosts, VCP routers compute only a congestion level, and end-hosts run one of the two algorithms as a function of the congestion level. More precisely, VCP classifies the network utilization into different utilization regions [27] and determines the controller that is suitable for each region. Efficiency and fairness have different levels of relative importance in different utilization regions. When network utilization is low, the goal of VCP is to improve efficiency more than fairness. On the other hand, when utilization is high, VCP accords higher priority to fairness than efficiency. By decoupling these two issues, end-hosts have only a single objective in each region and thus need to apply only one congestion response. For example, one such choice of congestion response, which we use in VCP, is to perform MI in low utilization regions for improving efficiency, and to apply AIMD in high utilization regions for achieving fairness. The goal then is to switch between these two congestion responses depending on the level of network utilization.

#2, Use link load factor as the congestion signal.

XCP uses spare bandwidth (the difference between capacity and demand) as a measure of the degree of congestion. In VCP, we use load factor as the congestion signal, *i.e.,* the

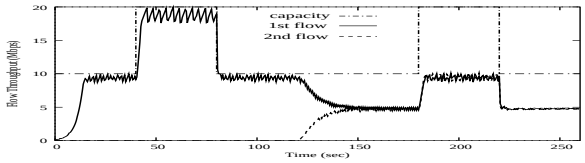


Figure 1: The throughput dynamics of two flows of the same RTT (80ms). They share one bottleneck with the capacity bouncing between 10Mbps and 20Mbps. This simple example unveils VCP's potential to quickly track changes in available bandwidth (with load-factor guided MIMD) and thereafter achieve a fair bandwidth allocation (with AIMD).

relative ratio of demand and capacity [26].

While the load factor conveys less information than spare bandwidth, the fact that the load factor is a *scale-free* parameter allows us to encode it using a small number of bits without much loss of information. In comparison to binary congestion signals such as loss and one-bit ECN, the load factor conveys more information about the degree of network congestion. In this paper, we show that a two-bit encoding of the load factor is sufficient to approximate XCP's performance, which demonstrates the significant marginal performance gain of this one more bit ECN.

2.3 A Simple Illustration

In this subsection, we give a high level description of VCP using a simple example. A detailed description of VCP is presented in Section 3. Periodically, each router measures the load factor for its output links and classifies the load factor into three utilization regions: low-load, high-load or overload. Each router encodes the utilization regions in the two ECN bits in the IP header of each data packet. In turn, the receiver sends back this information to the sender via the ACK packets. Depending on this congestion information, the sender applies different congestion responses. If the router signals low-load, the sender increases its sending rate using MI; if the router signals high-load, the sender increases its sending rate using AI; otherwise, if the router signals overload, the sender reduces its sending rate using MD. The core of the VCP protocol is summarized by the following greatly simplified pseudo code.

---

1) Each router periodically estimates a load factor, and encodes this load factor into the data packets' IP header. This information is then sent back by the receiver to the sender via ACK packets;

2) Based on the load factor it receives, each sender performs one of the following control algorithms:

2.1) For low-load, performs MI;

2.2) For high-load, performs AI;

2.3) For overload, performs MD.

Figure 1 shows the throughput dynamics of two flows sharing one bottleneck link. Clearly, VCP is successful in tracking the bandwidth changes by using MIMD, and achieving fair allocation when the second flow arrives, by using AIMD.

The Internet, however, is much more complex than this simplified example across many dimensions: the link capacities and router buffer sizes are highly heterogeneous, the RTT of flows may differ significantly, and the number of flows is unknown and changes over time. We next describe the details of the VCP protocol, which will be able to handle more realistic environments.

3. THE VCP PROTOCOL

In this section, we provide a detailed description of VCP. We begin by presenting three key issues that need to be addressed in the design of VCP. Then, we describe how we address each of these issues in turn.

3.1 Key Design Issues

To make VCP a practical approach for the Internet-like environments with significant heterogeneity in link capacities, end-to-end RTTs, router buffer sizes and variable traffic characteristics, we need to address the following three issues.

Load factor transition point: VCP separates the network load condition into three regions: low-load, high-load and overload. The load factor transition point in the VCP senders represents the boundary between the low-load and high-load regions, which is also the demarcation between applying MI and AI algorithms. The choice of the transition point represents a trade-off between achieving high link utilization and responsiveness to congestion. Achieving high network utilization requires a high value for the transition point. But this choice negatively impacts responsiveness to congestion, which in turn affects the convergence time to achieve fairness. Additionally, given that Internet traffic is inherently bursty [44, 54], we require a reliable estimation algorithm of the load factor at the VCP routers. We discuss these issues regarding load factor in Section 3.2.

Setting of congestion control parameters: Using MI for congestion control is often fraught with the danger of instability due to its large variations over short time scales. To maintain stability and avoid large queues at routers, we need to make sure that the aggregate rate of the VCP flows using MI does not overshoot the link capacity. Similarly, to achieve fairness, we need to make sure that a flow enters the AI phase before the link gets congested. In order to satisfy these criteria, we need an appropriate choice of MI, AI and MD parameters that can achieve high utilization while maintaining stability, fairness and small persistent queues. To better understand these issues, we first describe our parameter settings for a simplified network model, where all flows have the same RTT and observe the same state of the network load condition, *i.e.,* all flows obtain synchronous load factor feedback (Section 3.3). We then generalize our parameter choice for flows with heterogeneous RTTs.

Heterogeneous RTTs: When flows have heterogeneous RTTs, different flows can run different algorithms (*i.e.,* MI, AI, or MD) at a given time. This may lead to unpredictable behavior. The RTT heterogeneity can have a significant impact even when all flows run the same algorithm, if this algorithm is MI. In this case, a flow with a lower RTT can claim much more bandwidth than a flow with a higher RTT. To address this problem, end-hosts need to adjust their MI parameters according to their observed RTTs, as discussed in Section 3.4.

We now discuss these three design issues in greater detail.

3.2 Load Factor Transition Point

Consider a simple scenario involving a fixed set of long-

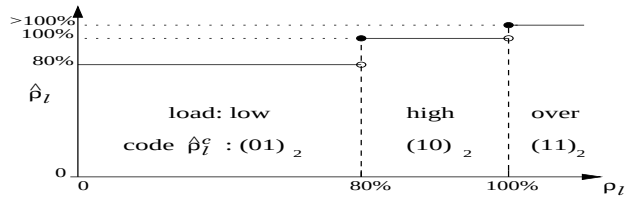


Figure 2: The quantized load factor $\hat{\rho}_l$ at a link l is a non-decreasing function of the raw load factor ρ_l and can be represented by a two-bit code $\hat{\rho}_l^c$.

lived flows. The goal of VCP is to reach a steady state where the system is near full utilization, and the flows use AIMD for congestion control. To achieve this steady state, the choice of the load factor transition point at the VCP senders should satisfy three constraints:

- The transition point should be sufficiently high to enable the system to obtain high overall utilization;
- After the flows perform an MD from an overloaded state, the MD step should force the system to always enter the high-load state, not the low-load state;
- If the utilization is marginally lower than the transition point, a single MI step should only lift the system into the high-load state, but not the overload state.

Let $\beta < 1$ denote the MD factor, *i.e.,* when using the MD algorithm, the sender reduces the congestion window with the factor β (as in Equation (4) in Section 3.3). The first constraint requires a high transition point. This choice coupled with the second condition leads to a high value of β. However, a very high value of β is undesirable as it decreases VCP's response to congestion. For example, if the transition point is 95%, then $\beta > 0.95$, and it takes VCP about 14 RTTs to halve the congestion window. At the other end, if we chose $\beta = 0.5$ (as in TCP [24]), the transition point can be at most 50%, which reduces the overall network utilization. To balance these conflicting requirements, we chose $\beta = 0.875$, the same value used in the DECbit scheme [56]. Given β, we set the load factor transition point to 80%. This gives us a "safety margin" of 7.5%, which allows the system to operate in the AIMD mode in steady state. In summary, we choose the following three ranges to encode the load factor ρ_l (see Figure 2):

- Low-load region: $\hat{\rho}_l = 80\%$ when $\rho_l \in [0\%, 80\%)$;
- High-load region: $\hat{\rho}_l = 100\%$ when $\rho_l \in [80\%, 100\%)$;
- Overload region: $\hat{\rho}_l > 100\%$ when $\rho_l \in [100\%, \infty)$.

Thus, the quantized load factor $\hat{\rho}_l$ can be represented using a two-bit code $\hat{\rho}_l^c$, *i.e.,* $\hat{\rho}_l^c = (01)_2, (10)_2$ and $(11)_2$ for $\hat{\rho}_l = 80\%$, $\hat{\rho}_l = 100\%$ and $\hat{\rho}_l > 100\%$, respectively. The code $(00)_2$ is reserved for ECN-unaware source hosts to signal "not-ECN-capable-transport" to ECN-capable routers, which is needed for incremental deployment [55]. The encoded load factor is embedded in the two-bit ECN field in the IP header.

Estimation of the load factor: Due to the bursty nature of the Internet traffic, we need to estimate the load factor over an appropriate time interval, t_ρ. When choosing t_ρ we need to balance two conflicting requirements. On one hand, t_ρ should be larger than the RTTs experienced by most flows to factor out the burstiness induced by the flows' responses to congestion. On the other hand, t_ρ should be small enough to avoid queue buildup. Internet measurements [53, 29] report that roughly 75%∼90% of flows have RTTs less than 200 ms. Hence, we set $t_\rho = 200$ms. During every time interval t_ρ, each VCP router estimates a load factor ρ_l for each of its output links l as [26, 33, 17, 2, 41]:

$$\rho_l = \frac{\lambda_l + \kappa_q \cdot \widetilde{q}_l}{\gamma_l \cdot C_l \cdot t_\rho}. \quad (1)$$

Here, λ_l is the amount of input traffic during the period t_ρ, $\widetilde{q}_l$ is the persistent queue length during this period, κ_q controls how fast the persistent queue drains [17, 2] (we set $\kappa_q = 0.5$), [1] γ_l is the target utilization [41] (set to a value close to 1), and C_l is the link capacity. The input traffic λ_l is measured using a packet counter. To measure the persistent queue $\widetilde{q}_l$, we use a low-pass filter that samples the instantaneous queue size, $q(t)$, every t_q, where $t_q \ll t_\rho$ (we chose $t_q = 10$ms).

3.3 Congestion Control Parameter Setting

In this section, we discuss the choice of parameters used by VCP to implement the MI/AI/MD algorithms. To simplify the discussion, we consider a single link shared by flows, whose RTTs are equal to the link load factor estimation period, *i.e., rtt* $= t_\rho$. Hence, the flows have synchronous feedback and their control intervals are also in sync with the link load factor estimation. We will discuss the case of heterogeneous RTTs in Section 3.4.

At any time t, a VCP sender performs one of the three actions based on the value of the encoded load factor sent by the network:

$$\texttt{MI}: \quad cwnd(t + rtt) = cwnd(t) \times (1 + \xi) \quad (2)$$

$$\texttt{AI}: \quad cwnd(t + rtt) = cwnd(t) + \alpha \quad (3)$$

$$\texttt{MD}: \quad cwnd(t + \delta t) = cwnd(t) \times \beta \quad (4)$$

where $rtt = t_\rho$, $\delta t \to 0+$, $\xi > 0$, $\alpha > 0$ and $0 < \beta < 1$. Based on the relationship between the choice of the load factor transition point and the MD parameter β, we chose $\beta = 0.875$ (see Section 3.2). We use $\alpha = 1.0$ as is in TCP [24].

Setting the MI parameter: The stability of VCP is dictated by the MI parameter ξ. In network-based rate allocation approaches like XCP, the rate increase of a flow at any time is proportional to the spare capacity available in the network [34]. Translating this into the VCP context, we require the MI of the congestion window to be proportional to $1 - \hat{\rho}_l$ where $\hat{\rho}_l$ represents the current load factor. During the MI phase, the current sending rate of each flow is proportional to the current load factor $\hat{\rho}_l$. Consequently, we obtain

$$\xi(\hat{\rho}) = \kappa \cdot \frac{1 - \hat{\rho}_l}{\hat{\rho}_l}, \quad (5)$$

where κ is a constant that determines the stability of VCP and controls the speed to converge toward full utilization. Based on analyzing the stability properties of this algorithm (see Theorem 1 in Section 5), we set $\kappa = 0.25$. Since end-hosts only obtain feedback on the utilization region as opposed to the exact value of the load factor, they need to

[1] Note even though we explicitly take the router queue length into account, VCP's congestion measurement is essentially a load-based, instead of queue-based, scheme. Adding the queue length into the total amount of traffic only helps drain the queue faster.

make a conservative assumption that the network load is near the transition point. Thus, the end-hosts use the value of $\xi(80\%) = 0.0625$ in the MI phase.

3.4 Handling RTT Heterogeneity with Parameter Scaling

Until now, we have considered the case where competing flows have the same RTT, and this RTT is also equal to the load factor estimation interval, t_ρ. In this section, we relax these assumptions by considering flows with heterogeneous RTTs. To offset the impact of the RTT heterogeneity, we need to scale the congestion control parameters used by the end-hosts according to their RTTs.

Scaling the MI/AI parameters: Consider a flow with a round trip time rtt, and assume that all the routers use the same interval, t_ρ, to estimate the load factor on each link. Let ξ and α represent the *unscaled* MI and AI parameters as described in Section 3.3, where all flows have identical RTTs ($= t_\rho$). To handle the case of flows with different RTTs, we set the scaled MI/AI parameters ξ_s and α_s as follows: [2]

$$\texttt{For MI:} \quad \xi_s \leftarrow (1+\xi)^{\frac{rtt}{t_\rho}} - 1\,, \tag{6}$$

$$\texttt{For AI:} \quad \alpha_s \leftarrow \alpha \cdot \frac{rtt}{t_\rho}\,. \tag{7}$$

An end-host uses the scaled parameters ξ_s and α_s in (2) and (3) to adjust the congestion window after each RTT. The scaling of these parameters emulates the behavior of all flows having an identical RTT, which is equal to t_ρ. The net result is that over any time period, the window increase under either MI or AI is independent of the flows' RTTs. Thus, unlike TCP, VCP flow's throughput is not affected by the RTT heterogeneity [42, 51, 15].

Handling MD: MD is an impulse-like operation that is not affected by the length of the RTT. Hence, the value of β in (4) needs not to be scaled with the RTT of the flow. However, to avoid over reaction to the congestion signal, a flow should perform an MD at most once during an estimation interval t_ρ. Upon getting the first load factor feedback that signals congestion (*i.e.,* $\hat{\rho}_l^c = (11)_2$), the sender immediately reduces its congestion window *cwnd* using MD, and then freezes the *cwnd* for a time period of t_ρ for a new load factor to be generated at the routers. After this period, the end-host runs AI for one RTT, which is the time needed to obtain the new load factor.

Scaling for fair rate allocation: RTT-based parameter scaling, as described above, only ensures that the congestion *windows* of two flows with different RTTs converge to the same value in steady state. However, this does not guarantee fairness as the rate of the flow is still inversely proportional to its RTT, *i.e., rate* $= cwnd/rtt$. To achieve fair rate allocation, we need to add an additional scaling factor to the AI algorithm. To illustrate why this is the case, consider the simple AIMD control mechanism applied to two competing flows where each flow $i\,(= 1, 2)$ uses a separate AI parameter α_i but a common MD parameter β. At the end of the M-th congestion epoch that includes $n > 1$ rounds of AI and one round of MD in each epoch, we have

$$cwnd_i(M) = \beta \cdot [\, cwnd_i(M-1) + n \cdot \alpha_i \,].$$

Eventually, each flow i achieves a congestion window that is proportional to its AI parameter, α_i. Indeed, the ratio of the congestion windows of the two flows approaches α_1/α_2 for large values of M, and $n > 1$:

$$\begin{aligned}\frac{cwnd_1(M)}{cwnd_2(M)} &= \frac{cwnd_1(M-1)/n + \alpha_1}{cwnd_2(M-1)/n + \alpha_2} \\ &= \frac{\beta \cdot cwnd_1(M-2)/n + \beta\alpha_1 + \alpha_1}{\beta \cdot cwnd_2(M-2)/n + \beta\alpha_1 + \alpha_2} \\ &= \cdots \rightarrow \frac{\alpha_1}{\alpha_2}.\end{aligned}$$

Hence, to allocate bandwidth fairly among two flows, we need to scale each flow's AI parameter α_i using its own RTT. For this purpose, we use t_ρ as a common-base RTT for all the flows. Thus, the new AI scaling parameter, α_{rate}, becomes

$$\texttt{For AI:} \quad \alpha_{rate} \leftarrow \alpha_s \cdot \frac{rtt}{t_\rho} = \alpha \cdot (\frac{rtt}{t_\rho})^2. \tag{8}$$

3.5 Protocol Description

Putting all the above-discussed building blocks together, now we present the complete VCP protocol.

The router: The VCP router computes and encodes a load factor based on the number of incoming packets and the average queue for each output link. It tags the encoded load factor $\hat{\rho}_p^l$ into the IP header of the outgoing data packet if it is larger than the one, $\hat{\rho}_p^c$, carried by the packet from the upstream router. The VCP router also runs *two priority queues* with the high priority queue for the ACK packets (to minimize the feedback delay) and the low priority queue for the data packets. The VCP router algorithm is described as follows.

R.1) For each incoming packet p of size s_p, update a counter λ_l:

$$\lambda_l \leftarrow \lambda_l + s_p \qquad \textit{// Count}$$

R.2) When the queue sampling timer t_q fires at time t:

$$\widetilde{q}_l \leftarrow a \cdot \widetilde{q}_l + (1-a) \cdot q(t) \qquad \textit{// Average}$$

where $a = 0.875$; i.e., like RED, maintain a low-pass filtered queue length $\widetilde{q}_l$ using exponentially weighted moving average (EWMA). We set $t_q = 10$ms;

R.3) When the load factor measurement timer t_ρ fires:

$$\begin{aligned}\rho_l &= \frac{\lambda_l + \kappa_q \cdot \widetilde{q}_l}{\gamma_l \cdot c_l \cdot t_\rho} && \textit{// Measure} \\ \hat{\rho}_l^c &\leftarrow \texttt{encode}(\rho_l) && \textit{// Encode} \\ \lambda_l &\leftarrow 0 && \textit{// Reset}\end{aligned}$$

where $\kappa_q = 0.5$, $t_\rho = 200$ms, link target utilization $\gamma_l = 0.98$, c_l is the link capacity, and the encoding function is described in Section 3.2;

R.4) For each dequeued data packet p that bears an encoded load factor $\hat{\rho}_p^c$ from upstream:

$$\hat{\rho}_p^c \leftarrow \max(\hat{\rho}_l^c, \hat{\rho}_p^c) \qquad \textit{// Tag}$$

---

[2] Equation (6) is the solution for $1 + \xi = (1 + \xi_s)^{\frac{t_\rho}{rtt}}$ where the right-hand side is the MI amount of a flow with the RTT value rtt, during a time interval t_ρ. Similarly, Equation (7) is obtained by solving $1 + \alpha = 1 + \frac{t_\rho}{rtt}\,\alpha_s$.

Table 1: VCP Parameter Settings

Para	Value	Meaning
t_ρ	200 ms	the link load factor measurement interval
t_q	10 ms	the link queue sampling interval
γ_l	0.98	the link target utilization
κ_q	0.5	how fast to drain the link steady queue
ξ	0.0625	the MI parameter
α	1.0	the AI parameter
β	0.875	the MD parameter
σ_{mi}	2.5	the MI scaling limiter
σ_{ai}	10.0	the AI scaling limiter

The end hosts: The VCP receiver is the same as the TCP Reno receiver, except that it copies the encoded two-bit load factor $\hat{\rho}_p^c$ ECN from the data packet to its corresponding ACK packet.

The VCP sender builds upon the TCP Reno sender. It behaves like TCP Reno when packet loss happens. The VCP sender initializes the encoded load factor $\hat{\rho}_p^c$ in the data packet IP header to the smallest one, i.e, $(00)_2$. It switches its window-based control among MI/AI/MD according to the encoded load factor $\hat{\rho}_p^c$ carried back by the ACK packet. This switching is performed as follows.

S.1) For the low-/medium-load factors $\hat{\rho}_p^c = (00)_2$ or $(01)_2$, per ACK:

$$inc \leftarrow (1.0+\xi)^{\min(\frac{srtt}{t_\rho},\ \sigma_{mi})} - 1.0 \quad // \ Scaling$$
$$cwnd \leftarrow cwnd + inc \quad // \ MI$$

where $\xi = 0.0625$, $srtt$ is the smoothed RTT measurement in ms, $t_\rho = 200$ms, and the MI scaling limiter $\sigma_{mi} = 2.5$ (to bound the bursty traffic introduced due to the MI scaling);

S.2) For the high-load factor $\hat{\rho}_p^c = (10)_2$, per ACK:

$$inc \leftarrow \alpha \cdot \min((\frac{srtt}{t_\rho})^2 \cdot w,\ \sigma_{ai}) \quad // \ Scaling$$
$$cwnd \leftarrow cwnd + inc\,/\,cwnd \quad // \ AI$$

where $\alpha = 1.0$, the AI scaling limiter $\sigma_{ai} = 10.0$ (to bound the bursty traffic introduced due to the AI scaling), and the weight w is settable with a default value 1.0; (Section 4.5 has details on setting different weights for flows to achieve weighted bandwidth sharing.)

S.3) For the overload factor $\hat{\rho}_p^c = (11)_2$, do the following cut once, then firstly freeze $cwnd$ for one t_ρ and secondly follow S.2 for one $srtt$, regardless of the remaining load factor feedbacks during these two time periods:

$$cwnd \leftarrow \max(1.0,\ \beta \cdot cwnd) \quad // \ MD$$

where $\beta = 0.875$.

Table 1 summarizes the VCP router (upper part) and end-host (lower part) parameters and their typical values.

4. PERFORMANCE EVALUATION

In this section, we use extensive ns2 simulations to evaluate the performance of VCP for a wide range of network scenarios [18] including varying the link capacities in the range [500Kbps, 5Gbps], round trip times in the range [1ms, 1.5s], numbers of long-lived, FTP-like flows in the range [1, 1000], and arrival rates of short-lived, web-like flows in the range $[1s^{-1}, 1000s^{-1}]$. We always use two-way traffic with congestion resulted in the reverse path. The bottleneck buffer size is set to the bandwidth-delay product, or two packets per flow, whichever is larger. The data packet size is 1000 bytes, while the ACK packet is 40 bytes. All simulations are run for at least 120s to ensure that the system has reached its steady state. The average utilization statistics neglect the first 20% of simulation time. For all the time-series graphs, utilization and throughput are averaged over 500ms interval, while queue length and congestion window are sampled every 10ms. Throughout all the simulations in this paper, we use the *same* set of parameter values listed in Table 1. This suggests that VCP is robust in a large variety of environments.

For comparison purpose, we also run simulations for other schemes including TCP Reno [1], SACK [49], HSTCP [14], STCP [35], FAST [30], BIC [64], CUBIC [57], HTCP [43], and XCP [34], under the same network and traffic settings. Except for XCP which has its own router algorithm, we run RED [17] with ECN enabled in the bottleneck routers for each of the above schemes (except REM [2] for FAST). The protocol parameter settings of these schemes are those recommended by their respective authors. The simulation results demonstrate that, for a wide range of scenarios, VCP achieves comparable performance to XCP, *i.e.*, exponential convergence to high utilization, negligible packet drop rate, low persistent queue and reasonable fairness, except its significantly slower fairness convergence speed than XCP. VCP and XCP significantly outperforms the other eight schemes.

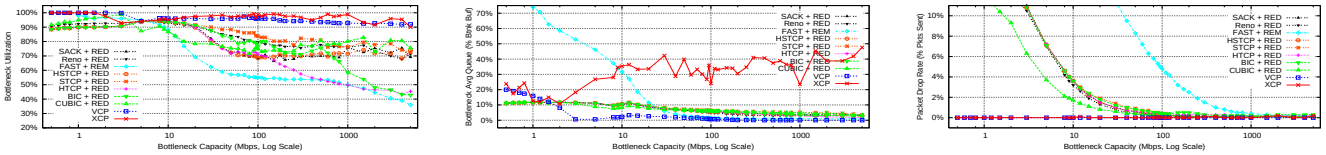
4.1 One Bottleneck

We first evaluate the performance of all the schemes for the simple case of a single bottleneck link shared by multiple flows. We study the effect of varying the link capacity, the round-trip times, the number of flows on the performance of VCP. The basic setting is a 150Mbps link with 80ms RTT where the forward and reverse path each has 30 FTP flows. We evaluate the impact of each network parameter in isolation while retaining the others as the basic setting. Each simulation result is averaged over five simulation runs with random flow start times.

Varying Bottleneck Capacity: As illustrated in Figure 3, we observe that, among all the schemes, only VCP and XCP achieve high utilization ($> 90\%$) and no packet drops across the whole range of bottleneck link capacities varying from 500Kbps to 5Gbps. The utilization gap between VCP and XCP is at most 6% across the entire capacity range. However, VCP maintains much lower persistent bottleneck queue length (less than 20% bottleneck buffer size, mostly below 2%) than XCP ($10\% \sim 47\%$ buffer size). This is because that the VCP router gives higher priority to ACK packets than data packets, as described in Section 3.5. We believe that XCP should be able to achieve the same low-queue performance if it does the same.

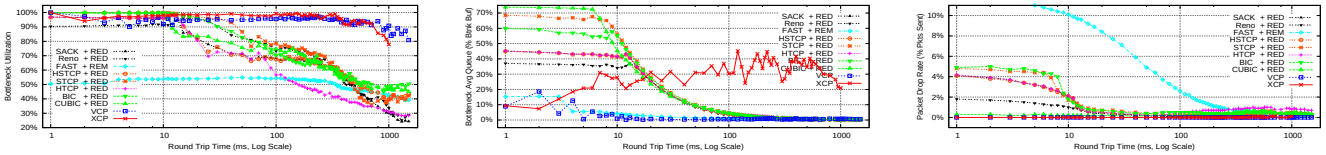
For all the other schemes, as we scale the bottleneck capacity to beyond 200Mbps, the bottleneck utilization mostly drops to around $70\% \sim 80\%$ (even less than 60% for HTCP and FAST). When the capacity is below 5Mbps, all the schemes result in higher than 4% packet loss rate.

Varying Feedback Delay: We fix the bottleneck capacity at 150Mbps and vary the round-trip propagation delay



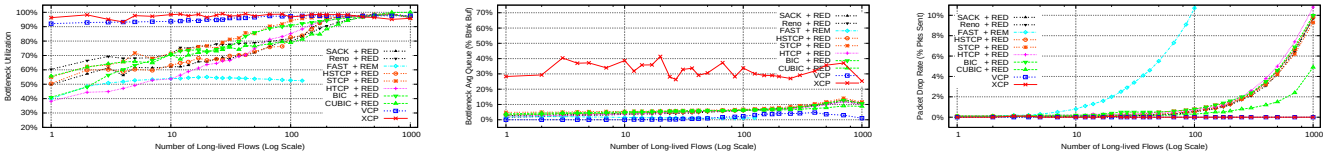
(a) Bottleneck Average Utilization (b) Bottleneck Average Queue (c) Packet Drop Rate

Figure 3: One bottleneck with the capacity varying from 500Kbps to 5Gbps. Note the logarithmic scale of the x-axis.



(a) Bottleneck Average Utilization (b) Bottleneck Average Queue (c) Packet Drop Rate

Figure 4: One bottleneck with the round-trip propagation delay ranging from 1ms to 1500ms.



(a) Bottleneck Average Utilization (b) Bottleneck Average Queue (c) Packet Drop Rate

Figure 5: One bottleneck with the number of long-lived, FTP-like flows increasing from 1 to 1000.

from 1ms to 1.5s. As shown in Figure 4, we notice that, in most cases, VCP's bottleneck utilization is higher than 90%, and the average bottleneck queue is less than 2% buffer size. We also observe that VCP's RTT parameter scaling is more sensitive to very low values of RTT (*e.g.,* <5ms), thereby causing the average queue length to grow to about 10% ∼ 20% buffer size. For the RTT values larger than 800ms, VCP obtains lower utilization (80% ∼ 90%) since the link load factor measurement interval t_ρ = 200ms is much less than the RTTs of the flows. As a result, the load condition measured in each t_ρ shows variations due to the bursty nature of window-based control. This can be compensated by increasing t_ρ; but the trade-off is that the link load measurement will be less responsive causing the queue length to grow. In all these cases of wide RTT variation, we did not observe any packet drops in VCP.

Comparing to the other schemes, VCP's performance is comparable to XCP's (lower utilization but also lower queue length) and it performs significantly better than all the other TCP variants, which achieve lower than 60% bottleneck utilization when the RTT is higher than 400ms.

Varying The Number of Long-lived Flows: With an increase in the number of forward FTP flows, we notice that the traffic gets more bursty, as shown by the increasing trend of the bottleneck average queue in Figure 5. However, even when the network is very heavily multiplexed by more than 500 flows (*i.e.,* the average per-flow BDP is no more than 3 packets), the 90-percentile queue is still less than 20% of the buffer size. The average queue is consistently less than 5% buffer size across all the simulation cases.

For high per-flow BDP scenarios, where there is no more than 10 flows on the forward path (*i.e.,* the per-flow BDP is 150 ∼ 1500 packets), only VCP and XCP achieve higher than 90% bottleneck utilization. The other schemes' bottleneck utilization is only between 40% and 70%, far less than that of VCP and XCP.

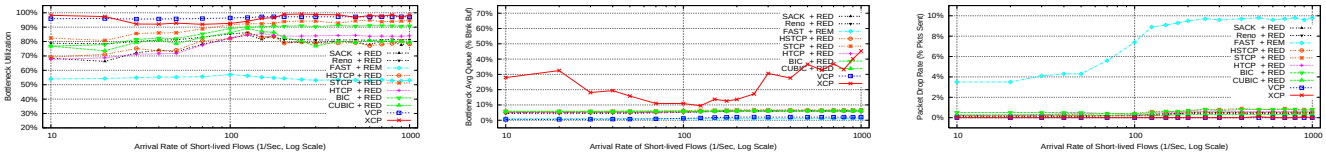
Varying Short-lived Traffic: To study VCP's performance in the presence of variability and burstiness in flow arrivals, we add short-lived traffic into the network. These flows arrive according to the Poisson process, with the average arrival rate varying from 1/s to 1000/s. Their transfer size obeys the Pareto distribution with an average of 30 packets. As shown in Figure 6, VCP always maintains high utilization (>95%) with small queue lengths (less than 3% bottleneck buffer size) and no packet drops, similar to XCP (which has higher queue lengths).

In summary, we note that across a wide range of network configurations with a single bottleneck, VCP can achieve high utilization, low persistent queue, and negligible packet drops. VCP's performance is comparable to XCP's and is significantly better than that of all the other schemes.

4.2 Multiple Bottlenecks

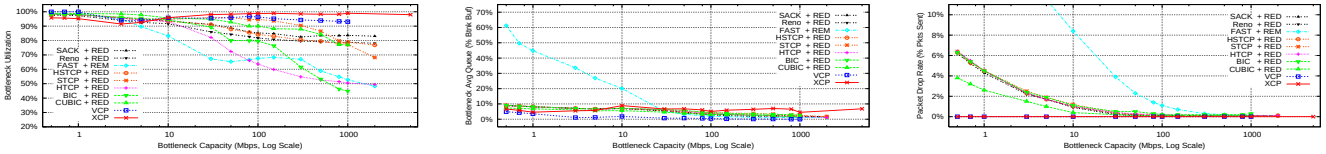
Next, we study the performance of VCP with a more complex topology of multiple bottlenecks. For this purpose, we use a typical parking-lot topology with seven links. There are 30 long FTP flows traversing all the links in the forward direction, and 30 long FTP flows in the reverse direction. In addition, each individual link has 5 cross FTP flows traversing in the forward direction. We run two sets of simulations by varying link bandwidth and path RTT, respectively, in a range of three to four orders of magnitudes.

Varying Bottleneck Capacity: First, we set all the bottleneck links' one-way propagation delay to 5ms. The longest path's round-trip propagation delay is 80ms. We vary all the bottleneck links' bandwidth from 500Kbps to 5Gbps. Figure 7 shows that, for all the cases, VCP performs as good as in the single-bottleneck scenarios. It achieves achieves at least 93% average bottleneck utilization (averaged among all the seven bottlenecks), less than 5%-buffer-size average queue length and no packet drops at all the bottlenecks.



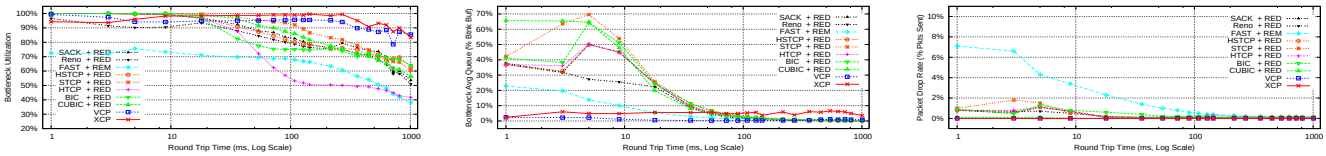
(a) Bottleneck Average Utilization (b) Bottleneck Average Queue (c) Packet Drop Rate

Figure 6: One bottleneck with short-lived, web-like flows arriving/departing at a rate from 1/ s to 1000/ s.



(a) Bottleneck Average Utilization (b) Bottleneck Average Queue (c) Packet Drop Rate

Figure 7: Seven bottlenecks with the capacity varying from 500Kbps to 5Gbps.



(a) Bottleneck Average Utilization (b) Bottleneck Average Queue (c) Packet Drop Rate

Figure 8: Seven bottlenecks with the longest path's round-trip propagation delay ranging from 1ms to 1000ms.

In comparison to XCP, one key difference is that *VCP penalizes flows that traverse more bottlenecks.* For example, when the bottlenecks' capacity is 150Mbps, VCP allocates ??Mbps to each long flow that traverses all the seven bottlenecks, and ??Mbps to each cross flow that passes one bottleneck; while all these flows get about ??Mbps under XCP. We discuss the reason behind this in Section 5. The other TCP variants exhibit similar behavior as VCP, except FAST.??

Varying Path RTT: Second, we fix all the bottlenecks' bandwidth to 150Mbps and vary the longest path's round-trip propagation delay from 10ms to 1s. Again, VCP and XCP outperform all the other schemes, as shown in Figure 8. Comparing to XCP, VCP trades a few percent of bottleneck utilization for lower bottleneck queue length. Both VCP and XCP drop no packet for all the simulation cases.

In brief summary, the simulation results we obtain for the multiple-bottleneck scenarios are consistent with the single-bottleneck cases. VCP's performance is close to XCP's; both significantly outperforms the other TCP variants.

4.3 RTT Fairness

TCP flows with different RTTs achieve bandwidth allocations that are proportional to $1/rtt^z$ where $1 \leq z \leq 2$ [42]. VCP alleviates this issue to some extent. Here we look at the RTT-fairness of VCP and the other schemes. We have 30 FTP flows sharing a single 150Mbps bottleneck, with 30 FTP flows on the reverse path. Each forward flow i's RTT value $rtt_i = 40 + i * rtt_{delta}$ ms for $i = 0, 1, ..., 29$. We perform eleven sets of simulations with rtt_{delta} increasing from 0, 1, 2, ..., to 10 ms. When rtt_{delta} = 0ms, all the flows' RTTs equal to 40ms; As rtt_{delta} increases to 4ms, the RTTs are in the range of [40ms, 156ms] with the RTT ratio of about 4; When rtt_{delta} = 10ms, the RTTs are in the range of [40ms, 330ms] with the RTT ratio of more than 8.

Figure 9 shows that, in terms of RTT fairness, XCP performs the best by achieving Jain's fairness index [28] [3] of 1.0 across tho whole set of simulations, closely followed by VCP ($0.94 \sim 1.0$) and CUBIC ($0.94 \sim 0.98$). All the other seven schemes fall short of distributing bandwidth fairly among flows with heterogenous RTTs.[4] Among the three RTT-fair schemes, XCP's average bottleneck utilization is 98% and its 90-percentile bottleneck queue length is on average 30% bottleneck buffer size. In contrast, VCP's average bottleneck utilization is slightly less (94%) and the 90-percentile bottleneck queue length is only 5% bottleneck buffer size, while CUBIC achieves 88% average bottleneck utilization and 10%-bottleneck-buffer-size 90-percentile queue.

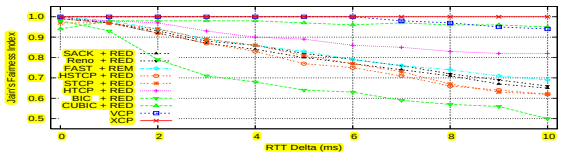
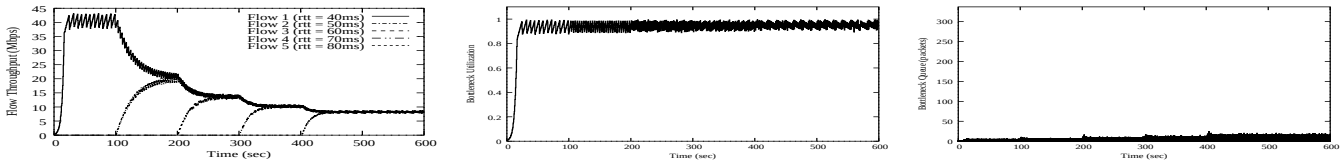


Figure 9: Jain's fairness index under scenarios of one bottleneck shared by flows with RTTs in the ranges from [40ms, 40ms] to [40ms, 330ms].

The fairness discrepancy of VCP for large rtt_{delta} cases occurs due to the following reason. A flow with very high RTT is bound to have high values for their MI and AI parameters due to parameter scaling as described in Section 3.4. To prevent sudden traffic bursts from such VCP flows which can cause the bottleneck instantaneous queue to increase substantially, we place bounds on the actual values

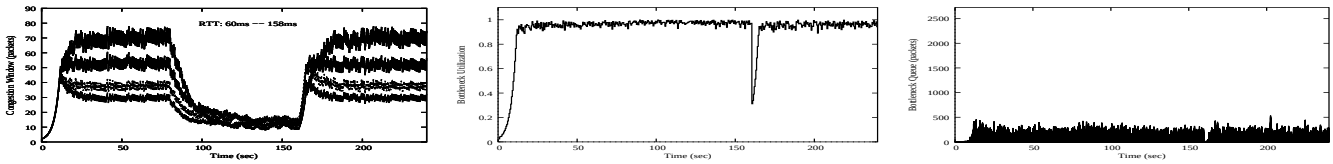
[3] It is defined by $\frac{(\sum_{i=1}^{N} x_i)^2}{N \cdot \sum_{i=1}^{N} x_i^2}$ for flow rates $x_i, i \in [1, N]$.

[4] It is debatable if we should allocate bandwidth equally regardless of flow RTT. Here we assume we should do so. It is easy to tailor VCP (by using (7) instead of (8) in Section 3.4) to achieve bandwidth allocation proportional to RTT^{-1}.



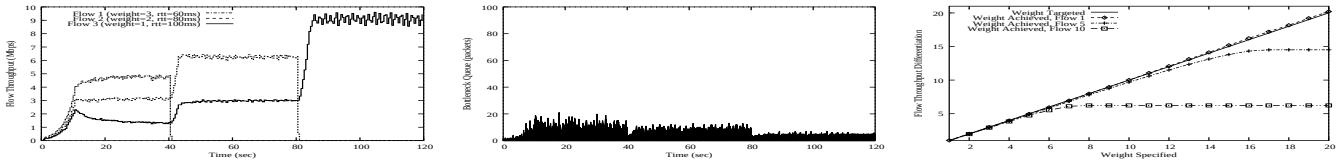
(a) Flow Throughput (b) Bottleneck Utilization (c) Bottleneck Queue

Figure 10: VCP converges onto good fairness, high utilization and small queue. However, its fairness convergence takes significantly longer time than XCP.



(a) Flow Congestion Window (b) Bottleneck Utilization (c) Bottleneck Queue

Figure 11: VCP is robust against and responsive to sudden, considerable traffic demand changes, and at the same time maintains low persistent bottleneck queue.



(a) Flow Throughput (b) Bottleneck Queue (c) Achieved Weight

Figure 12: VCP is able to differentiate bandwidth allocation among competing flows of heterogeneous RTTs. However, the RTT heterogeneity limits the largest achievable wight.

of these parameters (see Section 3.5). These bounds restrict the throughput of flows with very high RTTs.

4.4 Dynamics

All the previous simulations focus on comparing the steady-state behavior of VCP and the other schemes. Now, we investigate the short-term dynamics of VCP.

Convergence Behavior: To study the convergence behavior of VCP, we revert to the single bottleneck link with a bandwidth of 45Mbps where we introduce 5 flows into the system, one after another, with starting times separated by 100s. We also set the RTT values of the five flows to different values. The reverse path has 5 flows that are always active. Figure 10 illustrates that VCP reallocates bandwidth to new flows whenever they come in without affecting its high utilization or causing large instantaneous queue. However, VCP takes a much longer time than XCP to converge to the fair allocation. We theoretically quantify the fairness convergence speed for VCP in Theorem 4 in Section 5, where we also show that VCP's susb-linear fairness convergence can be significantly improved to exponential speed using a larger number (*e.g.,* eight) of ECN bits.

Sudden Demand Change: We illustrate how VCP reacts to sudden changes in traffic demand using a simple simulation. Consider an initial setting of 50 forward FTP flows with varying RTTs (uniformly chosen in the range [60ms, 160ms]) sharing a 200Mbps bottleneck link. There are 50 FTP flows on the reverse path. At t=80s, 150 new forward FTP flows become active; then they leave at 160s. Figure 11 clearly shows that VCP can adapt sudden fluctuations in the traffic demand. (The left figure draws the congestion window dynamics for four randomly chosen flows.) When the new flows enter the system, the flows adjust their rates to the new fair share while maintaining the link at high utilization. At t=160s, when three-fourths of the flows depart creating a sudden drop in the utilization, the system quickly discovers this and ramps up to 95% utilization in about 5 seconds. Notice that during the adjustment period, the bottleneck queue remains much lower than its full size. This simulation shows that VCP is responsive to sudden, significant decreases/increases in the available bandwidth. This is no surprise because VCP switches to the MI mode which by nature can track any bandwidth change in logarithmic time (see Theorem 3 in Section 5).

4.5 Bandwidth Differentiation

The analysis in Section 3.4 shows that the steady state bandwidth allocation of an AIMD scheme is proportional to the AI parameter α, given the same MD parameter β. Thus, by simply scaling α_{rate} in Equation (8) with a weight ω_i for each flow i:

$$\alpha_i \leftarrow \omega_i \cdot \alpha_{rate} = \omega_i \cdot \alpha \cdot (\frac{rtt}{t_\rho})^2, \quad (9)$$

VCP can provide differentiated bandwidth to competing flows that are on the same path.

Weighted Bandwidth Sharing: Figure 12 shows VCP's bandwidth differentiation capability. In our first simulation, a 10Mbps bottleneck is shared by 3 FTP flows with different RTTs: $rtt_1 = 60$ms, $rtt_2 = 80$ms, $rtt_3 = 100$ms, and different weights: $\omega_1 = 3$, $\omega_2 = 2$, $\omega_3 = 1$. They all start at 0s but stop at 40s, 80s, and 120s, respectively. There are also 3 reverse FTP flows that are always on and all with weight 1. The left and middle figures in Figures 12 clearly demon-

strate that the bottleneck bandwidth is distributed among the three flows according to their specified weights without introducing large queue in the bottleneck.

Achievable Weight Range: In our second set of simulations, we evaluate the range of weights achievable. We have a bottleneck of capacity 100Mbps shared by 10 FTP flows of heterogeneous RTT ranging from 60ms to 150ms ($rtt_i = 50 + 10 \times i$ ms for $i = 1, 2, \ldots, 10$). One flow has a varying weight in $[1, 20]$ while others have the same unit weight. There are also 10 reverse FTP flows all with unit weight. We simulate three cases with the weighted flow being flow #1 (60ms RTT), #5 (100ms RTT) and #10 (150ms RTT), respectively. Each simulation runs for 150s. The rightmost Figure 12 shows the achieved bandwidth ratio between the weighted flow and the average of all the others. The achieved weight range for flow #1 is [1.0, 20.4], closely matching the specified wights, while flows #5 and #10 achieve only [1.0, 15.2] and [1.0, 6.6], respectively, indicating the influence of their larger RTTs. All these results are achieved while the bottleneck is highly utilized with low persistent queue.

The effect of RTT heterogeneity on the achievable weight range is due to the following reason. Note the AI parameter scaling in Equation (9) might introduce a significant amount of bursty traffic within one RTT if $rtt \gg t_\rho$ or ω is huge and therefore cause packet losses. To limit this effect, similar to the previous discussion in Section 4.3, we impose an upper bound on α_i such that the burstiness can be effectively absorbed by the router buffer. This upper bound causes the large-RTT flows to receive a lower bandwidth share than otherwise.

5. A FLUID MODEL

To obtain insight into the VCP protocol, in this section, we analyze its stability, fairness, and convergence properties using a simplified fluid model. We start our analysis by considering a single bottleneck with infinite buffer traversed by N flows that have the same RTT, T.

To make the analysis tractable, we use the following load-factor guided algorithm to *approximate* the behavior of VCP as defined by (2), (3), and (4) in Section 3.3:

$$\dot{w}_i(t) = \frac{1}{T} \cdot [\, w_i(t) \cdot \xi(\rho(t)) \,+\, \alpha \,] \tag{10}$$

with the MI parameter

$$\xi(\rho(t)) = \kappa \cdot \frac{1 - \rho(t)}{\rho(t)}, \tag{11}$$

where $\kappa > 0$ is the stability coefficient of the MI parameter. In the remainder of this section we will refer to this model as the MIAIMD model. This model makes three simplifications comparing to the VCP protocol. First, it uses the exact load factor value $\rho(t)$, while VCP uses a quantized value of the load factor. Second, in the MI and AI phases, VCP uses either the multiplicative factor or the additive factor term, but not both as the MIAIMD model does. Third, in the overload region, VCP applies a constant MD parameter β instead of $\xi(\rho(t))$.

As shown in Figure 13, the load factor $\rho(t)$ received by the *source* at a time t is computed based on the sender's rate at time $t - T$,

$$\rho(t) = \frac{\sum_{i=1}^{N} w_i(t - T)}{\gamma CT}, \tag{12}$$

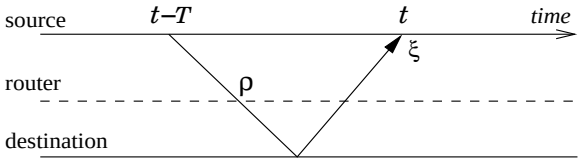


Figure 13: A simplified VCP model. The source sending rate at time $t - T$ is used by the router to calculate a load factor ρ, which is echoed back from the destination to the source at time t. Then the source adjusts its MI parameter $\xi(\rho(t))$ based on the load factor $\rho(t)$.

where $w_i(t)$ is the flow i's congestion window at time t, C is the link capacity, and $0 < \gamma \leq 1$ is the target link utilization. We assume that $w_i(t)$ is a positive, continuous and differentiable function, and T is a constant.

Since $\xi(\rho(t))$ is proportional to the available bandwidth, the MIAIMD algorithm tracks the available bandwidth exponentially fast and thus achieves efficiency. It also converges to fairness as we will show in Theorem 2. [5]

Using (10) to sum over all N flows yields

$$\dot{w}(t) = \frac{1}{T} \cdot [\, w(t) \cdot \xi(\rho(t)) \,+\, N\alpha \,] \tag{13}$$

where $w(t) = \sum_{i=1}^{N} w_i(t)$ is the sum of all the congestion windows. This result, together with (11) and (12), leads to

$$\dot{w}(t) = \frac{1}{T} \cdot \{\, \kappa \cdot w(t) \cdot [\, \frac{\gamma CT}{w(t - T)} - 1 \,] \,+\, N\alpha \,\} \tag{14}$$

where $w(t) > 0$. We assume the initial condition $w(t) = N$ (*i.e.*, $w_i(t) = 1$), for all $t \in [-T, 0]$. In [63], we prove the following global stability result.

Theorem 1. Under the model (10), (11) and (12) where a single bottleneck is shared by a set of synchronous flows with the same RTT, if $\kappa \leq \frac{1}{2}$, then the delayed differential equation (14) is globally asymptotically stable with a unique equilibrium $w^* = \gamma CT + N\frac{\alpha}{\kappa}$, and all the flows have the same steady-state rate $r_i^* = \frac{\gamma C}{N} + \frac{\alpha}{\kappa T}$.

This result has two implications. First, the sufficient condition $\kappa \leq \frac{1}{2}$ holds for any link capacity, any feedback delay, and any number of flows. Furthermore, the global stability result does *not* depend on the network parameters. Second, this result is optimal in that at the equilibrium, the system achieves all the design goals: high utilization, fairness, zero steady-state queue length, and zero packet loss rate—this is because we can always adjust γ such that the system stabilizes at a steady-state utilization slightly less than 1.

Importance of γ: While (12) defines γ as the target utilization, the actual utilization is $\frac{w^*}{CT} = \gamma + \frac{\alpha}{\kappa P}$ where $P = \frac{CT}{N}$ is the per-flow BDP. To achieve a certain target utilization γ^*, γ should be treated as a *control variable* and set to $\gamma = \gamma^* - \frac{\alpha}{\kappa P}$. To make this adjustment process automatic without even knowing α, κ and P, we vary γ at a time scale that is much larger than one RTT, i.e.,

$$\gamma(t + T_\gamma) = \gamma(t) + \mathtt{sgn}\,(\gamma^* - \widetilde{\gamma}^*(t)) \cdot \delta\gamma \tag{15}$$

where $T_\gamma \gg T$, sgn$(\cdot)$ is the sign function, $\widetilde{\gamma}^*(t)$ is a low-pass

[5] Theorem 2 actually proves the max-min fairness for a general multiple-bottleneck topology. For a single link, max-min fairness means each flow gets an equal share of the link capacity.

filtered link utilization which is very easy to measure at the router, and $0 < \delta\gamma \ll \gamma^*$ is a constant stepsize (set to 0.01). This adjustment process will stop if and only if the target utilization γ^* has been achieved.

For the case of heterogeneous delays, suppose flow i's RTT is T_i, for all $i \in [1, N]$. *If* we still have a unique equilibrium, then it is straight-forward to show that the equilibrium window size $w^* = \gamma C \hat{T}_h + N\frac{\alpha}{\kappa}$ where $\hat{T}_h = N/\sum_{i=1}^{N}\frac{1}{T_i}$, which is the harmonic average of all the RTTs, and flow i has a steady-state rate $r_i^* = \frac{\gamma C}{N}\frac{\hat{T}_h}{T_i} + \frac{\alpha}{\kappa}\frac{1}{T_i}$.

Next, we consider a more general multiple-bottleneck network topology. Let $\rho^i(t)$ denote the *maximal* link load factor on flow i's path L_i that includes a subset of links, *i.e.,* $L_i = \{\, l \mid \text{flow } i \text{ traverses link } l\}$. The MI parameter of flow i is then

$$\xi(\rho^i(t)) = \kappa \cdot [\,\frac{1}{\rho^i(t)} - 1\,], \tag{16}$$

where $\rho^i(t) = \max_{l\in L_i}\rho_l(t)$, $\rho_l(t) = \frac{\sum_{i\in I_l} w_i(t-T)}{\gamma C_l T}$, and the subset of flows $I_l = \{\, i \mid \text{flow } i \text{ traverses link } l\}$. We prove the following fairness result in [63].

Theorem 2. In a multiple-bottleneck topology where all flows have the same round-trip time T, if there exists a unique equilibrium, then the algorithm defined by (10) and (16) allocates a set of max-min fair rates $r_i^* = \frac{\alpha}{\kappa T(1-\frac{1}{\max_{l\in L_i}\rho_l^*})}$ where $\rho_l^* = \frac{\sum_{i\in I_l} w_i^*}{\gamma C T}$.

To better understand this result note that a flow's sending rate is determined by the most congested bottleneck link on its path. Thus, the flows traversing the most congested bottleneck links in the system will naturally experience the lowest throughputs.

Having established the stability and fairness properties of the MIAIMD model, we now turn our attention on the convergence of the VCP protocol. The following two theorems, proved in [63], give the convergence properties.

Theorem 3. The VCP protocol takes $O(\log C)$ RTTs to claim (or release) a major part of any spare (or over-used) capacity C.

Theorem 4. The VCP protocol takes $O(P\log\Delta P)$ RTTs to converge onto fairness for any link, where P is the per-flow bandwidth-delay product, and $\Delta P > 1$ is the largest congestion window difference between flows sharing that link.

Not surprisingly, due to the use of MI in the low-load region, VCP converges exponentially fast to high utilization. On the other hand, VCP's convergence time to fairness is similar to other AIMD-based protocols, such as TCP+AQM. In contrast, explicit feedback schemes like XCP require only $O(\log\Delta P)$ RTTs to converge to fairness. This is because the end-host based AIMD algorithms improve fairness per AIMD *epoch*, which includes $O(P)$ rounds of AI and one round of MD, while the equivalent operation in XCP takes only one RTT.

The comparison between the simulation results of VCP and the analytical results of the MIAIMD model suggests that the two differ most notably in terms of the fairness model. While in the case of multiple bottleneck links, the MIAIMD model achieves max-min fairness [4], VCP tends to allocate more bandwidth to flows that traverse fewer bottleneck links (see Section 4.2). This is because VCP relies on the quantized representation of the load factor instead of the exact value.

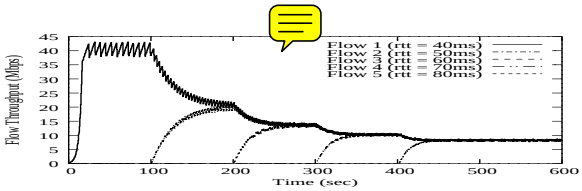


Figure 14: The sub-linear fairness convergence of the VCP protocol (shown in Figure 10) can be improved significantly if the load factor information is encoded with 8 ECN bits.

Given a larger number of bits to encode the load factor value $\rho(t)$, within one AIMD epoch VCP will be able to do *multiple* MD cuts with an adaptive MD parameter $\xi(\rho(t))$, as oppose to the constant β, and thus can potentially improve its fairness convergence speed. To demonstrate this we use eight ECN bits to encode the value of the load factor (with finer than 1% precision) and repeat the first simulation in Section 4.4. Figure 14 shows that, comparing to the sub-linear convergence speed in Figure 10, VCP now converges onto fairness exponentially.

6. DISCUSSIONS

Since VCP switches between MI, AI, and MD algorithms based on the load factor feedback, there are natural concerns with respect to the impact of these switches on the system stability, efficiency, and fairness, particularly in systems with highly heterogeneous RTTs. We discuss these concerns as well as VCP's TCP-friendliness and incremental deployment in this section.

6.1 Stability under Heterogeneous Delays

Although the MIAIMD model presented in Section 5 is provably stable, it assumes synchronous feedback. To accommodate heterogeneous delays, VCP scales the MI/AI parameters such that flows with different RTTs act as if they were having the same RTT. This scaling mechanism is also essential to achieving fair bandwidth allocation, as discussed in Section 3.4.

In normal circumstances, VCP makes a transition to MD only from AI. However, even if VCP switches directly from MD to MI, if the demand traffic at the router does not change significantly, VCP will eventually slide back into AI.

Finally, to prevent the system from oscillating between MI and MD, we set the load factor transition point $\hat{\rho}_l$ to 80%, and set the MD parameter β to $0.875 > \hat{\rho}_l$. This gives us a safety margin of 7.5%.

The extensive simulation results presented in Section 4 suggest that VCP is indeed stable over a large variety of network scenarios including per-flow bandwidths from 2Kbps to 100Mbps and RTTs from 1ms to 1.5s.

6.2 Influences of Mode Sliding

From an efficiency perspective, VCP's goal is to bring and maintain the system into the high utilization region. While MI enables VCP to quickly reach the high link utilization, VCP needs also to make sure that the system remains in this state. The main mechanisms employed by VCP to achieve this goal is the scaling of the MI/AI parameters for flows with different RTTs. In addition to improving fairness, this

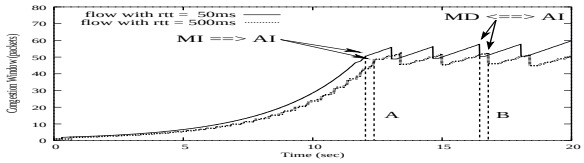


Figure 15: The congestion window dynamics of two flows with dramatically different RTTs (50ms vs. 500ms). Due to its longer delay, the larger-RTT flow always slides its mode later than the one with smaller RTT (see the regions labeled as A and B). However, the effect of this asynchronous switching is accommodated by VCP and does not prevent it from maintaining stability and achieving efficiency and fairness.

scaling is essential to avoid oscillations. Otherwise, a flow with a low RTT may apply MI several times during the estimation interval, t_ρ, of the link load factor. Other mechanisms employed by VCP to maintain high efficiency include choosing an appropriate value of the MD parameter to remain in the high utilization region, using a safety margin between MI and AI, and bounding the burstiness (Section 4.3).

As discussed in Section 3.4, there are two major concerns with respect to fairness. First, a flow with a small RTT probes the network faster than a flow with a large RTT. Thus, the former may increase its bandwidth much faster than the latter. Second, it will take longer for a large-RTT flow to switch from MI to AI than a small-RTT flow. This may give the large-RTT flow an unfair advantage. VCP addresses the first issue by using the RTT scaling mechanism (see (6)-(7)). To address the second issue, VCP bounds the MI gain, as discussed in Section 4.3. To illustrate the effectiveness of limiting the MI gain, Figure 15 shows the congestion window evolution for two flows with RTTs of 50ms and 500ms, respectively, traversing a single 10Mbps link. At time 12.06s, the 50ms-RTT flow switches from MI to AI. In contrast, due to its larger RTT, the 500ms-RTT flow keeps performing MI until time 12.37s. However, because VCP limits the MI gain of the 500ms-RTT flow, the additional bandwidth acquired by this flow during the 0.31s interval is only marginal when compared to the bandwidth acquired by the 50ms-RTT flow.

6.3 TCP-Friendliness

We define a VCP flow to be *TCP-friendly* with a competing TCP flow if the steady state throughput of the TCP flow matches what it would when competing with a normal TCP flow [51, 15]. However in high BDP networks, a VCP flow should be able to leverage the additional bandwidth unused by the TCP flows while not affecting the throughput of TCP flows. Because VCP operates with AIMD in steady state, it is straight-forward to tailor it to exhibit TCP-friendly behavior. At the end host, to match TCP's AI parameter, we need to change the VCP AI parameter to $\alpha = \frac{3(1-\beta)}{1+\beta} = 0.2$ according to the TCP-friendly general AIMD formula [65]. At the router, when the encoded load factor $\hat{\rho}_l^c = (11)_2$, we need to replace the original deterministic ECN marking with a probabilistic one similar to RED [17]. For TCP sources, in accordance with the ECN proposal, the encoded load factors $(01)_2$ and $(10)_2$ correspond to no congestion, while $(11)_2$ to congestion.

6.4 Incremental Deployment

If VCP is to be gradually deployed on the Internet, the deployment could follow the similar path as CSFQ [58] and XCP on an island-by-island basis. Therefore, even though VCP looks simpler than XCP, the deployment cost is quite similar, *not* much less. The deployment, however, will still benefit from VCP's simplicity: It does not need a new field in the IP header; the needed two-bit space has been standardized for congestion control purposes by the current ECN proposal and VCP uses it in a way that is a natural generalization of ECN. From the end hosts perspective, VCP can be made TCP-friendly, as described earlier. On the network side, as we have shown, the VCP router is scalable in that it does not keep any per-flow state and its algorithm complexity is very low. This makes it deployable in high speed core networks. The traffic inside an VCP island will immediately enjoy VCP's capability of maintaining high utilization, low persistent queue and minimal packet drop. The cross traffic that passes an VCP island between two border routers will be mapped onto an VCP flow from the ingress router to the egress router. These border routers do need to keep per-VCP-flow state. However, since the VCP flow is aggregated from the passing micro-flows, this will not cause scalability problems.

7. RELATED WORK

This paper builds upon a great body of related work, particularly XCP [34], TCP [24, 1, 16, 49], AIMD [10, 28], AQM [17, 2, 41] and ECN [55, 56]. Congestion control is pioneered by TCP and AIMD. The research on AQM starts from RED [17, 45], followed by Blue [13], REM [2], PI controller [22], AVQ [20, 41], and CHOKe [52], etc. Below we relate VCP to three categories of congestion control schemes and a set of analytical results.

Explicit rate based schemes: XCP regulates source sending rate with decoupled efficiency control and fairness control and achieves excellent performance. ATM ABR service (*e.g.,* see [39, 9, 32, 26, 33]) previously proposes explicit rate control. VCP learns from these schemes. In contrast, VCP is primarily an end-host based protocol. This key difference brings new design challenges not faced by XCP (and the ATM ABR schemes) and thus VCP is not just a "two-bit" version of XCP. The idea of classifying network load into different regions is originally presented in [27]. The link load factor is suggested as a congestion signal in [26], based on which VCP quantizes and encodes it for a more compact representation for the degree of congestion. MaxNet [62] uses the maximal congestion information among all the bottlenecks to achieve max-min fairness. QuickStart [25] occasionally uses several bits per packet to quickly ramp up source sending rates. VCP is complementary to QuickStart as it constantly uses two bits per packet.

Congestion notification based schemes: For high BDP networks, according to [34], the performance gap between XCP and TCP+RED/REM/AVQ/CSFQ [58] with one-bit ECN support seems large. VCP generalizes one-bit ECN and applies some ideas from these AQM schemes. For example, RED's queue-averaging idea, REM's match-rate-clear-buffer idea and AVQ's virtual-capacity idea obviously find themselves in VCP's load factor calculation in Equation (1). This paper demonstrates that the marginal performance gain from one-bit to two-bit ECN feedback could be significant. On the end-host side, two-bit ECN is also used to choose

different decrease parameters for TCP in [12], which is very different from the way VCP uses. GAIMD [65] and the binomial control [3] generalize the AIMD algorithm, while VCP goes even further to combine MIMD with AIMD.

Pure end-to-end schemes: Recently there have been many studies on the end-to-end congestion control for high BDP networks. HSTCP [14] extends the standard TCP by adaptively setting the increase/decrease parameters according to the congestion window size. H-TCP [43] employs an adaptive AIMD with its parameters set as functions of the elapsed time since the last congestion event. Adaptive TCP [37] also applies dynamic AIMD parameters with respect to the changing network conditions. STCP [35] changes to a fixed MIMD algorithm. FAST [30] uses queueing delay, like TCP Vegas [6], instead of packet loss, as its primary congestion signal and improves Vegas' Additive-Increase-Additive-Decrease policy with a proportional controller. BIC [64, 57] adds a binary search phase into the standard TCP to probe the available bandwidth in a logarithmic manner. LTCP [5] layers congestion control of two scales for high speed, large RTT networks. TCP Westwood [8] enhances the loss-based congestion detector using more robust bandwidth estimation techniques. All these end-to-end schemes do not need explicit feedback. Therefore, it is hard for them to achieve *both* low persistent bottleneck queue length and almost zero congestion-caused packet loss rate. VCP does need explicit two-bit ECN but is able to maintain low queue and almost zero loss. The extensive simulations in Section 4 show that, even with AQM/ECN support from network, these schemes still can not achieve similar performance as VCP in high BDP networks.

Analytical Results: The nonlinear optimization framework [36, 46, 40] provides the above schemes a unified theoretic underpin and proposes a class of control algorithms. The local stability of the algorithms when homogeneous delay is present is considered by [31, 59] and then extended to the case of heterogeneous delays by [48]. The local stability of a modified algorithm for the case of heterogeneous delays is proved by [67], which establishes a model that is similar to what we show in Section 5. In contrast, a global stability result is obtained in this paper for the case of a single bottleneck with homogeneous delays. The global stability of more general congestion controllers are considered by other researchers, *e.g.,* in [60, 11, 66].

8. SUMMARY AND FUTURE WORK

In this paper, we propose VCP, a simple, low-complexity congestion control protocol for high BDP networks. Using extensive ns2 simulations, we show that VCP achieves high utilization, reasonable fairness, low persistent bottleneck queue, and negligible packet loss rate. VCP achieves all these desirable properties while requiring only two bits to encode the network congestion information. Since it can leverage the two ECN bits to carry this information, VCP requires no changes of the IP header. In this respect, VCP can be seen as an extension of the TCP+AQM/ECN proposals that scales to high BDP networks.

To better understand the behavior of VCP, we propose a fluid model, and use this model to analyze the efficiency, fairness, and convergence properties of a simplified version of VCP. Particularly, we prove that the model is globally stable for the case of a single bottleneck link shared by long-lived flows with identical RTTs.

As future work, it would be interesting to study if we can design a pure end-to-end VCP without any explicit congestion information from network. One possibility would be to use packet loss to differentiate between overload and high-load regions and to use RTT variation to differentiate between high-load and low-load regions. [6] While in this paper we evaluate VCP through extensive simulations, ultimately, only a real implementation and deployment will allow us to asses the strengths and limitations of VCP.

The ns2 implementation and simulation code of VCP is available at http://networks.ecse.rpi.edu/∼xiay.

9. ACKNOWLEDGEMENTS

Funding ack.

10. REFERENCES

[1] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. *IETF RFC 2581*, April 1999.
[2] S. Athuraliya, V. Li, S. Low, and Q. Yin. REM: Active Queue Management. *IEEE Network*, 15(3):48-53, May 2001.
[3] D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms. *INFOCOM'01*, April 2001.
[4] D. Bertsekas and R. Gallager. Data Networks. 2nd Ed., Simon & Schuster, December 1991.
[5] S. Bhandarkar, S. Jain, and A. Reddy. Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control. *PFLDNet'05*, February 2005.
[6] L. Brakmo and L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE J. Selected Areas in Communications*, 13(8):1465-1480, October 1995.
[7] H. Bullot and R. Les Cottrell. Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks. Available at http://www.slac.stanford.edu/grp/scs/net/talk03/tcp-slac-nov03.pdf.
[8] C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi, and R. Wang. TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks. *Wireless Networks Journal*, 8(5):467-479, September 2002.
[9] A. Charny, D. Clark, and R. Jain. Congestion Control with Explicit Rate Indication. *IEEE ICC'95*, June 1995.
[10] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *J. of Computer Networks and ISDN*, 17(1):1-14, June 1989.
[11] S. Deb and R. Srikant. Global Stability of Congestion Controllers for the Internet. *IEEE Trans. Automatic Control*, 48(6):1055-1060, June 2003.
[12] A. Durresi, M. Sridharan, C. Liu, M. Goyal, and R. Jain. Multilevel Explicit Congestion Notification. *SCI'01*, July 2001.
[13] W. Feng, K. Shin, D. Kandlur, and D. Saha. The BLUE active queue management algorithms. *IEEE/ACM Trans. Networking*, 10(4):513-528, August 2002.
[14] S. Floyd. HighSpeed TCP for Large Congestion Windows. *IETF RFC 3649*, December 2003.
[15] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. *SIGCOMM'00*, August 2000.
[16] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. *IETF RFC 2582*, April 1999.
[17] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Trans. Networking*, 1(4):397-413, August 1993.
[18] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Trans. Networking*, 9(4):392-403, August 2001.
[19] E. Gafni and D. Bertsekas. Dynamic Control of Session Input Rates in Communication Networks. *IEEE Trans. Automatic Control*, 29(11):1009-1016, November 1984.
[20] R. Gibbens and F. Kelly. Resource Pricing and the Evolution of Congestion Control. *Automatica*, 35:1969-1985, 1999.

[6] While an end-to-end measured increase in RTT is hardly a reliable indicator of congestion in today's Internet, no such increase might very probably mean no congestion.

[21] K. Gopalsamy. Stability and Oscillations in Delay Differential Equations of Population Dynamics. Kluwer Academic Publishers, 1992.
[22] C. Hollot, V. Misra, D. Towlsey, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. *INFOCOM'01*, April 2001.
[23] C. Hollot, V. Misra, D. Towsley, and W. Gong. Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. *IEEE Trans. Automatic Control*, 47(6):945-959, June 2002.
[24] V. Jacobson. Congestion Avoidance and Control. *SIGCOMM'88*, August 1988.
[25] A. Jain and S. Floyd. Quick-Start for TCP and IP. *IETF Internet Draft draft-amit-quick-start-02.txt*, October 2002.
[26] R. Jain, S. Kalyanaraman, and R. Viswanathan. The OSU Scheme for Congestion Avoidance in ATM Networks: Lessons Learnt and Extensions. *Performance Evaluation*, 31(1):67-88, November 1997.
[27] R. Jain and K. K. Ramakrishnan. Congestion Avoidance in Computer Networks with A Connectionless Network Layer: Concepts, Goals, and Methodology. *Proc. IEEE Computer Networking Symposium*, April 1988.
[28] R. Jain, K. K. Ramakrishnan, and D. Chiu. Congestion Avoidance in Computer Networks with a Connectionless Network Layer. *DEC-TR-506*, August 1987.
[29] H. Jiang and C. Dovrolis. Passive Estimation of TCP Round-Trip Times. *ACM Computer Communications Review*, 32(3):75-88, July 2002.
[30] C. Jin, D. Wei, and S. Low. FAST TCP: Motivation, Architecture, Algorithms, Performance. *INFOCOM'04*, March 2004.
[31] R. Johari and D. Tan. End-to-End Congestion Control for the Internet: Delays and Stability. *IEEE/ACM Trans. Networking*, 9(6):818-832, December 2001.
[32] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Dynamics of an Explicit Rate Allocation Algorithm for Available Bit-Rate (ABR) Service in ATM Networks. *Proceedings of the IFIP/IEEE Conference on Broadband Communications*, April 1996.
[33] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore. The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks. *IEEE/ACM Trans. Networking*, 8(1), February 2000.
[34] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. *SIGCOMM'02*, August 2002.
[35] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *ACM Computer Communications Review*, 32(2), April 2003.
[36] F. Kelly, A. Maulloo, and D. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 49:237-252, 1998.
[37] A. Kesselman and Y. Mansour. Adaptive TCP Flow Control. *PODC'03*, July 2003.
[38] Y. Kuang. Delay Differential Equations with Applications in Population Dynamics. Academic Press, 1993.
[39] H. Kung, T. Blackwell, and A. Chapman. Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing. *SIGCOMM'94*, August 1994.
[40] S. Kunniyur and R. Srikant. End-To-End Congestion Control: Utility Functions, Random Losses and ECN Marks. *INFOCOM'00*, March 2000.
[41] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. *SIGCOMM'01*, August 2001.
[42] T. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-delay Products and Random Loss. *IEEE/ACM Trans. Networking*, 5(3):336-350, June 1997.
[43] D. Leith and R. Shorten. H-TCP: TCP for High-speed and Long-distance Networks. *PFLDnet'04*, February 2004.
[44] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the Self-Similar Nature of Ethernet Traffic. *SIGCOMM'93*, August 1993.
[45] D. Lin and R. Morris. Dynamics of Random Early Detection. *SIGCOMM'97*, August 1997.
[46] S. Low and D. Lapsley. Optimization Flow Control, I: Basic Algorithm and Convergence. *IEEE/ACM Trans. Networking*, 7(6):861-875, December 1999.
[47] S. Low, F. Paganini, J. Wang, and J. Doyle. Linear Stability of TCP/RED and a Scalable Control. *Computer Networks Journal*, 43(5):633-647, December 2003.
[48] L. Massoule. Stability of Distributed Congestion Control with Heterogeneous Feedback Delays. *IEEE Trans. Automatic Control*, 47(6):895-902, June 2002.
[49] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options. *IETF RFC 2018*, October 1996.
[50] Network Simulator ns-2. Http://www.isi.edu/nsnam/ns/.
[51] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *SIGCOMM'98*, September 1998.
[52] R. Pan, K. Psounis, and B. Prabhakar. CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. *INFOCOM'00*, March 2000.
[53] V. Paxson. End-to-End Internet Packet Dynamics. *SIGCOMM'97*, September 1997.
[54] V. Paxson and S. Flyod. Wide-Area Traffic: The Failure of Poisson Modeling. *SIGCOMM'94*, August 1994.
[55] K. K. Ramakrishnan and S. Floyd. The Addition of Explicit Congestion Notification (ECN) to IP. *IETF RFC 3168*, September 2001.
[56] K. K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *SIGCOMM'88*, August 1988.
[57] I. Rhee and L. Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *PFLDNet'05*, February 2005.
[58] I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. *SIGCOMM'98*, September 1998.
[59] G. Vinnicombe. On the Stability of End-to-end Congestion Control for the Internet. *Univ. of Cambridge Tech Report CUED/F-INFENG/TR.398*, December 2000.
[60] J. Wen and M. Arcak. A Unifying Passivity Framework for Network Flow Control. *INFOCOM'03*, March 2003.
[61] E. Wright. A Non-linear Difference-Differential Equation. *J. Reine Angew. Math.*, 494:66-87, 1955.
[62] B. Wydrowski and M. Zukerman. MaxNet: A Congestion Control Architecture for Maxmin Fairness. *IEEE Comm. Letters*, 6(11):512-514, November 2002.
[63] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit is Enough. *U.C.Berkeley Tech Report*, June 2005. (The version without proof details in Appendix appeared at *SIGCOMM'05*.)
[64] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. *INFOCOM'04*, March 2004.
[65] Y. Yang and S. Lam. General AIMD Congestion Control. *ICNP'00*, November 2000.
[66] L. Ying, G. Dullerud, and R. Srikant. Global Stability of Internet Congestion Controllers with Heterogeneous Delays. *Proc. American Control Conference*, June 2004.
[67] Y. Zhang, S. Kang, and D. Loguinov. Delayed Stability and Performance of Distributed Congestion Control. *SIGCOMM'04*, September 2004.
[68] Y. Zhang, D. Leonard, and D. Loguinov. JetMax: Scalable Max-Min Congestion Control for High-Speed Heterogeneous Networks. *INFOCOM'06*, April 2006.

11. APPENDIX

11.1 Proof of Theorem 1

To analyze the stability of Equation (14), let $y(t) = \frac{\gamma CT}{w(t)}$, then after some manipulation we obtain

$$\dot{y}(t) = k_1\, y(t)\,[1 - y(t-T) - k_2\, y(t)] \tag{17}$$

where $k_1 = \frac{\kappa}{T} > 0$, and $k_2 = \frac{N\alpha}{\kappa\,\gamma CT} > 0$. We also have, for all $t > 0, y(t) \geq 0$ since $w(t) > 0$, and the initial condition $y(t) > 0$ (since we assume $w(t) = N < \infty$) for $t \in [-T, 0]$.

Observing that the trivial solution $y(t) = 0$ is not a stable equilibrium of this equation, since any small perturbation will move the system further off the origin, we therefore

assume $y(t) > 0, \forall t > 0$. Physically, this assumption means that, at any time, the congestion window $w(t)$ is not infinite, which generally holds in reality.

To prove this theorem, we first establish two lemmas, applying the techniques developed in [61, 38, 21], to which we would like to give credit.

Lemma 1. If $\kappa \leq 1 - \frac{N\alpha}{\gamma CT}$, then (17) has a unique equilibrium $y^* = \frac{1}{1+k_2}$ that is globally asymptotically stable.

Proof. Substituting $x(t) = y(t) - y^*$ in (17) we get

$$\dot{x}(t) = -k_1 \, [\, y^* + x(t) \,] \, [\, x(t-T) + k_2 \, x(t) \,] \tag{18}$$

with a solution

$$y^* + x(t) = [\, y^* + x(t_0) \,] \cdot e^{-k_1 \int_{t_0-T}^{t-T} [\, x(\tau) + k_2 \, x(\tau+T) \,] \, d\tau} \tag{19}$$

where t_0 is a constant [38]. To show $\lim_{t\to+\infty} x(t) = 0$, we treat separately the following two cases.

Case #1. If $x(t)$ is not oscillatory, *i.e.,* $|x(t)| > 0$ when $t > t_1$ for some $t_1 > 0$. For $x(t) > 0$, when $t > t_1 + T$, we have $\dot{x}(t) < 0$ due to (17), which means $x(t)$ is strictly decreasing for all $t > t_1 + T$. Because $x(t) > 0$, there exists a constant c such that $\lim_{t\to+\infty} x(t) = c$ and thus $\lim_{t\to+\infty} \dot{x}(t) = 0$. We must have $c = 0$. Otherwise $c > 0$, then $\lim_{t\to+\infty} \dot{x}(t) = -c \, k_1 (y^* + c)(1 + k_2) < 0$ according to (17), resulting in a contradiction. The same analysis applies to when $x(t) < 0$;

Case #2. If $x(t)$ is oscillatory, *i.e.,* there is a sequence $t_l' > t_2$ for some $t_2 > 0$, $t_l' \to +\infty$ as $l \to +\infty$, such that $x(t_l') = 0$ for all t_l'. We firstly prove that $x(t)$ is bounded when $t > t_2$. Obviously $x(t)$ is lower bounded, since $x(t) = y(t) - y^* > -y^*$ as $y(t) > 0$. Now we show that $x(t)$ is also upper bounded. Let $t_3, t_4 \in (t_2, \infty)$, $t_3 < t_4$ be any two consecutive zeros of $x(t)$ such that $x(t) > 0$ for $t_3 < t < t_4$. (If $x(t) < 0$ we get an upper bound 0.) Because $w(t)$ is continuous and differentiable, so does $x(t) = y(t) - y^* = \frac{\gamma CT}{w(t)} - y^*$. Since $x(t_3) = x(t_4) = 0$, $x(t)$ has a local maximum in the interval (t_3, t_4). Denote it as $x(t_m)$ where $t_3 < t_m < t_4$, we have $x(t) \leq x(t_m)$ for all $t \in (t_3, t_4)$. We also have $\dot{x}(t_m) = 0$, which deduces $x(t_m - T) + k_2 \, x(t_m) = 0$ from (18). Substituting this result, $x(t) > -y^*$, and $y^* = \frac{1}{1+k_2}$ in (19) and setting $t_0 = t_m - T$, $t = t_m$, we get

$$-k_1 \int_{t_0-T}^{t-T} [\, x(\tau) + k_2 \, x(\tau+T) \,] \, d\tau < k_1 T$$

and thus

$$x(t) \;\leq\; x(t_m) \;<\; \frac{y^* (e^{k_1 T} - 1)}{1 + k_2 e^{k_1 T}} \tag{20}$$

for all $t \in [t_3, t_4]$. Repeating this process for all the consecutive zero pairs of $x(t)$ in (t_2, ∞), we conclude that $x(t)$ is bounded for $t > t_2$.

Since $x(t)$ is continuous and bounded, denote

$$u = \limsup_{t\to+\infty} x(t), \qquad v = -\liminf_{t\to+\infty} x(t). \tag{21}$$

Obviously we have

$$u \geq -v. \tag{22}$$

We now prove that $u = v = 0$ (therefore $\lim_{t\to+\infty} x(t) = 0$). Let $\epsilon > 0$ be an arbitrarily small constant such that, for all $t > t_5 = t_5(\epsilon) > 0$,

$$-v - \epsilon < x(t) < u + \epsilon. \tag{23}$$

As per the definition of u, we can always find a local maximum $x(t_n) > u - \epsilon$ for $t_n > t_5 + T$. Applying the same technique used to derive (20) on t_n, plus the left half of (23), we get

$$u - \epsilon \;<\; x(t_n) \;<\; \frac{y^* [\, e^{\phi(v+\epsilon)} - 1 \,]}{1 + k_2 e^{\phi(v+\epsilon)}} \tag{24}$$

where $\phi = k_1(1 + k_2)T = \kappa + \frac{N\alpha}{\gamma CT} > 0$. Since (24) holds for all $\epsilon > 0$, we conclude

$$u \leq \frac{y^* (e^{\phi v} - 1)}{1 + k_2 e^{\phi v}}, \tag{25}$$

where $0 < y^* = \frac{1}{1+k_2} < 1$ and $k_2 > 0$. Following similar steps in deriving (25) on a local minimum generates

$$v \leq \frac{y^* (1 - e^{-\phi u})}{1 + k_2 e^{-\phi u}}. \tag{26}$$

Now we discuss the following combinations of u and v and show that the only possibility is $u = v = 0$, if $\phi \leq 1$.

i) If $u < 0$, then $v < 0$ according to (26), so $-v > 0 > u$, violating (22);

ii) If $u = 0$, then $v \geq 0$ from (22). We have $v \leq 0$ as well due to (26). Therefore $v = 0$;

iii) If $u > 0$, then $v > 0$ according to (25). From (26) we have $v < y^* < 1$. If $\phi \leq 1$, we get

$$1 + u < e^{\phi v} \leq e^{v} < e^{1 - e^{-\phi u}} \leq e^{1 - e^{-u}}. \tag{27}$$

However, for $u > 0$, we have

$$1 + u - e^{1 - e^{-u}} = \int_0^u \int_0^\zeta (1 - e^{-\eta}) e^{(1 - e^{-\eta} - \eta)} d\eta d\zeta > 0$$

which is a conflict with (27).

To sum up, we must have $u = v = 0$ if $\phi \leq 1$, *i.e.,* $\kappa \leq 1 - \frac{N\alpha}{\gamma CT}$, which deduces $\lim_{t\to+\infty} x(t) = 0$.

Next we turn to the second lemma.

Lemma 2. If $\kappa < \frac{N\alpha}{\gamma CT}$, then (17) has a unique equilibrium $y^* = \frac{1}{1+k_2}$ that is globally asymptotically stable.

Proof. Consider the following function [21]

$$V(t) = y(t) - y^* \log y(t) + \frac{k_1}{2} \int_{t-T}^{t} [\, y(\tau) - y^* \,]^2 \, d\tau. \tag{28}$$

Again let $x(t) = y(t) - y^*$. Due to (17) and $k_1 > 0$ we have

$$\begin{aligned}
\frac{dV}{dt} &= \dot{y}(t) \, [1 - \frac{y^*}{y(t)}] + \frac{k_1}{2} [\, x^2(t) - x^2(t-T) \,] \\
&= \frac{\dot{y}(t) \, x(t)}{y(t)} + \frac{k_1}{2} [\, x^2(t) - x^2(t-T) \,] \\
&= -k_1 \, x(t) [k_2 \, x(t) + x(t-T)] + \frac{k_1}{2} [\, x^2(t) - x^2(t-T) \,] \\
&= -\frac{k_1}{2} \, [\, (2k_2 - 1) \, x^2(t) + x^2(t-T) + 2x(t)x(t-T) \,] \\
&= -\frac{k_1}{2} \, \{\, 2(k_2 - 1) \, x^2(t) + [x(t) + x(t-T)]^2 \,\} \\
&\leq -k_1 (k_2 - 1) \, x^2(t).
\end{aligned} \tag{29}$$

Integrating this inequality over [0, t], we have

$$V(t) + k_1 (k_2 - 1) \int_0^t x^2(\tau) \, d\tau \;\leq\; V(0) \tag{30}$$

where $V(0) = y(0) - y^* \log y(0) + \frac{k_1}{2}\int_{-T}^{0}[y(\tau) - y^*]^2\, d\tau$ is bounded since $y(t)$ is finite when $t \in [-T, 0]$.

Note we have proved in Lemma 1 that $x(t)$ is bounded for all $t > t_2$ for some $t_2 > 0$. So does $y(t)$ because $y(t) = x(t) + y^*$. Thus $V(t)$ is also bounded. If $k_2 > 1$ (*i.e.,* $\kappa < \frac{N\alpha}{\gamma CT}$), we must have $\lim_{t\to+\infty} x(t) = 0$ since otherwise we obtain $k_1(k_2 - 1)\int_0^t x^2(\tau)\, d\tau \to +\infty$ as $t \to +\infty$, resulting in a conflict with (30).

Now we are ready to prove the theorem. When $\kappa \le \frac{1}{2}$, we always have at least one of the two inequalities, $\kappa \le 1 - \frac{N\alpha}{\gamma CT}$ and $\kappa < \frac{N\alpha}{\gamma CT}$, holds. We thus complete the proof using the above two lemmas. The equilibrium of (14) is therefore $w^* = \frac{\gamma CT}{y^*} = \gamma CT + N\frac{\alpha}{\kappa}$.

The steady-state load factor, due to (12), is $\rho^* = \frac{w^*}{\gamma CT} = 1 + \frac{N\alpha}{\kappa\gamma CT}$. Let (10) be 0, we obtain, for all $i \in [1, N]$, flow i's steady-state congestion window $w_i^* = \frac{\gamma CT}{N} + \frac{\alpha}{\kappa}$. Its steady-state rate is thus $r_i^* = \frac{w_i^*}{T} = \frac{\gamma C}{N} + \frac{\alpha}{\kappa T}$. QED.

11.2 Proof of Theorem 2

Denote the aggregate rate at link l as $A_l = \sum_{i\in I_l} r_i$ where $r_i = \frac{w_i}{T}$ is the sending rate of flow i. The load factor, according to (12), is thus $\rho_l = \frac{A_l}{\gamma C_l}$ where C_l is link l's capacity.

Define function [19]

$$g(\rho_l) = \frac{\alpha}{\kappa T(1 - \frac{1}{\rho_l})} \tag{31}$$

which is strictly decreasing when $\rho_l > 1$. (Note $\rho_l > 1$ for all $l \in L$ according to Theorem 1.) Then, if we set $\dot{w}(t) = 0$ in (10) and consider (16), we have the equilibrium rate of flow i as

$$\begin{aligned} r_i &= \frac{\alpha}{\kappa T(1 - \frac{1}{\max_{l\in L_i}\rho_l})} \\ &= \min_{l\in L_i} g(\rho_l) \\ &\le g(\rho_l) \qquad\qquad \forall l \in L_i. \end{aligned} \tag{32}$$

Consider the subset of link(s) L_m^* that has the highest load factor $\rho_m^* = \max_{l\in L}\rho_l$ among all the links:

$$L_m^* = \{\, l \in L \,|\, \rho_l = \rho_m^* \,\}, \tag{33}$$

and the subset I_m^* of flow(s) that traverse at least one link in L_m^*, *i.e.,*

$$I_m^* = \{\, i \in I \,|\, L_i \cap L_m^* \neq \phi \,\}. \tag{34}$$

So all the flows in I_m^* have the same rate $r_m^* = g(\rho_m^*) = \min_{l\in L} g(\rho_l)$, which is the lowest rate allocation.

Suppose r_m^* is not maximized, then there exists $\tilde{r}_{i_1} > r_m^*$ for $i_1 \in I_m^*$. If we pick one link l_1 from $L_{i_1} \cap L_m^* \neq \phi$, we have $\tilde{\rho}_{l_1} = \sum_{i\in I_{l_1}} \tilde{r}_i / C_l > \sum_{i\in I_{l_1}} r_m^*/C_l = \rho_m^*$. However, because $g(\rho_l)$ is strictly decreasing, this leads to $\tilde{r}_{i_1} \le g(\tilde{\rho}_{l_1}) < g(\rho_m^*) = r_m^*$, resulting in a conflict.

If there is no such $\tilde{r}_{i_1} > r_m^*$ for $i_1 \in I_m^*$, we remove the sets L_m^* and I_m^* from the network and reduce the capacity of the remaining links (if any, otherwise we are done) by the sum of the passing flows' sending rates, then repeat the above process until there is no link/flow left. The flow rates defined by (32) thus solve a hierarchy of optimization problems within each of them the minimal allocation is maximized—this is the definition of max-min fairness. QED.

11.3 Proof of Theorem 3

We first consider the MI part of VCP in Section 3.3 when there is bandwidth to claim. Due to the MI/AI parameter scaling in Section 3.4 that handles RTT difference, we can assume that flows have the same RTT. Suppose the flows start from the unit aggregate rate $r(0) = 1$. At the end of m rounds of MI, the aggregate rate

$$r(m) = r(0) \cdot (1 + \xi)^m$$

where $\xi = 0.0625$ as set in Section 3.3. Given any capacity $C > 1$, to reach a major part (*i.e.,* the load factor transition point, $\hat{\rho}_l = 80\%$) of it with MI, let $r(m) = \rho_0 C$, and then we obtain

$$m = \frac{\log \hat{\rho}_l C}{\log(1 + \xi)} = O(\log C). \tag{35}$$

The same result also holds when the flows have to release bandwidth with the MD algorithm of VCP. Now suppose the initial aggregate rate $r(0) = C > 1$, to reach $r(m) = 1$ where $r(m) = \beta \cdot r(m-1) = ... = \beta^m \cdot r(0)$, we get

$$m = \frac{\log C}{\log(1/\beta)} = O(\log C) \tag{36}$$

as well. QED.

11.4 Proof of Theorem 4

For fairness convergence speed we focus on the AIMD part of VCP in Section 3.3. Consider any bottleneck of capacity C shared by N flows of the same RTT value T. After an MD that cuts the aggregate congestion window w from CT to βCT, it takes $\frac{(1-\beta)CT}{N\alpha}$ rounds of AI (which we call an epoch) to increase w from βCT to CT. Note during AI all flows grow the same amount of congestion window; only MD reduces the congestion window difference. For any two flows i and j, at the m-th epoch, we have

$$\Delta w_{ij}(m) = \beta \cdot \Delta w_{ij}(m-1) = ... = \beta^m \cdot \Delta w_{ij}(0),$$

where $\Delta w_{ij}(m) = w_i(m) - w_j(m)$. To reach a small-enough congestion window difference, *e.g.,* $\Delta w_{ij}(m) = 1$, then we have

$$m = \frac{\log \Delta w_{ij}(0)}{\log(1/\beta)}.$$

The total number of RTTs spent is

$$m \cdot \frac{(1-\beta)CT}{N\alpha} = \frac{1-\beta}{\alpha \log(1/\beta)} \cdot P \log \Delta w_{ij}(0) \tag{37}$$

where $P = \frac{CT}{N}$ is the per-flow BDP.

The fairness convergence time of a link l is obviously the time needed for the two passing flows with the largest initial congestion window difference to reach the fair allocation, which is $O(P \log \Delta P)$, where $\Delta P = \max_{i,j\in I_l} \Delta w_{ij}(0)$.

In contrast, XCP's fairness convergence time is $O(\log \Delta P)$, since it shuffles bandwidth in each round of AIMD control. QED.