One More Bit is Enough

Yong Xia, *Member, IEEE*, Lakshminarayanan Subramanian, Ion Stoica, and Shivkumar Kalyanaraman, *Member, IEEE*

***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, CUBIC, etc. 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.

***Index Terms*—AQM, ECN, congestion control, stability, TCP.**

I. INTRODUCTION

THE Additive-Increase-Multiplicative-Decrease (AIMD) [8] congestion control algorithm employed by TCP [1], [17], [34] 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.

Manuscript received November 11, 2007; revised January 23, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor F. Paganini. This work was supported in part by the National Science Foundation (NSF) under Grants ITR-00225660, ANI-0515259, NSF Career Award ANI-0133811, NSF-ITR 0313095, in part by the Defense Advanced Research Projects Agency (DARPA) under Contract F30602-00-2-0537, in part by the California MICRO Program, in part by the Air Force ESC Hanscom, in part by MIT Lincoln Laboratory Letter No. 14-S-06-0206, and in part by Grants from Intel and AT&T Laboratories Research. An earlier verion of this paper appeared in ACM SIGCOMM 2005.

Y. Xia is with the NEC Laboratories China, Beijing 100084, China (e-mail: xiayong@research.nec.com.cn; xy12180@gmail.com).

L. Subramanian is with the Computer Science Department, New York University, New York, NY 10003 USA (e-mail: lakshmi@cs.nyu.edu).

I. Stoica is with the Computer Science Division, University of California, Berkeley, CA 94720 USA (e-mail: istoica@cs.berkeley.edu).

S. Kalyanaraman is with the Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: shivkuma@ecse.rpi.edu).

Digital Object Identifier 10.1109/TNET.2007.912037

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 [13], FAST [22], BIC/CUBIC [41], [48], STCP [26], and HTCP [31], although being attractive short-term solutions (due to a lesser deployment barrier), may not be suitable for the long term. Indeed, for congestion control purpose, the end-to-end schemes artificially introduce packet loss or queuing delay at bottleneck routers, which should be avoided in the first place. Otherwise, it incurs undesirable effects such as periodical increase in end-to-end delay, which makes real-time applications like VoIP and video conferencing very hard to work well in the Internet.

To address the limitations of end-to-end congestion control schemes, many researchers have proposed the use of *explicit* network feedback. However, while traditional *congestion notification* schemes such as Active Queue Management (AQM) and Explicit Congestion Notification (ECN) proposals [2], [15], [29], [39] 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. This remains true even if we replace the standard TCP with the high-speed TCP variants mentioned above, as shown by the simulation results in Section IV. XCP [25] addresses this problem by having routers estimate the fair flow rate and send this rate back to the senders and achieves excellent performance. Congestion control schemes that use *explicit rate* feedback have also been proposed in the context of the ATM Available Bit Rate (ABR) service [19], [24]. 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.

The above-discussed schemes represent two ends of a spectrum of congestion control solutions that differ in the *amount of congestion information* feedback from the network to the end hosts. At one end of the spectrum, the TCP+AQM/ECN schemes require no more than one bit information per packet. At the other end, XCP and ATM ABR schemes encode in each packet multiple bits of congestion information. More congestion information results in better performance. It is therefore intriguing to ask how many explicit bits per packet is sufficient to achieve XCP-comparable performance.

From an architectural point of view, XCP represents a big departure from the traditional TCP+AQM/ECN approach. XCP's congestion and fairness control algorithms run inside the network routers, not in the end hosts. As a consequence, XCP has a different deployment path from TCP+AQM/ECN. While the latter can start its deployment from the end hosts, and then incrementally add the AQM and ECN support into the network along the way, the former has to start from inside the network. In

1063-6692/$25.00 © IEEE

practice the end-host approach's deployment path seems easier to bootstrap.

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 three congestion states while still keeping TCP+AQM/ECN's architecture. 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* [19], and uses this factor to classify the level of congestion into three regions: low-load, high-load and overload [20]. 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 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 [35] 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 loss rate, low persistent queue length and reasonable fairness. One limitation of VCP (as is the case for other end-host based AIMD 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 [39]. 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 cost.

The rest of the paper is organized as follows. In Section II, we describe the guidelines that motivate the design of VCP and in Section III, we provide a detailed description of the VCP protocol. In Section IV, we evaluate VCP's performance using extensive simulations and compare with XCP as well as many other recently-proposed TCP variants. In Section V, we develop a fluid model that approximates VCP's behavior and characterize its stability, fairness and convergence properties, with the detailed proofs presented in the technical report [47]. Section VI addresses concerns on the stability of VCP under heterogeneous delays and the influence of switching between MI, AI and MD on efficiency and fairness. It also discusses VCP's TCP-friendliness and incremental deployment issues. We review related work in Section VII and summarize our findings in Section VIII.

II. Foundations

Let us first review why XCP scales to high BDP networks better than the existing TCP+AQM/ECN schemes. Then, we present two guidelines that form the basis of the VCP design.

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 [17], [25]. 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 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.

A. Design Guidelines

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 and 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 [20] and determines the controller that

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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 appropriately 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 relative ratio of demand and capacity [19]. 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 ECN bit.

B. 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 III. 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 it 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.

Fig. 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.

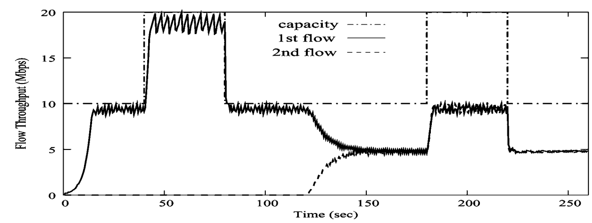


Fig. 1. The throughput dynamics of two flows of the same RTT (80 ms). They share one bottleneck with the capacity bouncing between 10 Mbps and 20 Mbps. This simple example unveils VCP's potential to quickly track changes in available bandwidth (with load-factor guided MIMD) and thereafter achieve a fair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

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.

III. 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.

A. 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 tradeoff 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, we require a reliable estimation algorithm of the load factor at the VCP routers. We discuss these issues in Section III-B.
- **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 III-C). 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 III-D.

We now discuss these three design issues in greater detail.

B. Load Factor Transition Point

Consider a simple scenario involving a fixed set of long-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 (4) in Section III-C). 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 at least 14 RTTs to halve the congestion window. At the other end, if we chose $\beta = 0.5$ (as in TCP [17]), 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 [40]. 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 Fig. 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 $\rho_l \in [100\%, \infty)$.

Then, 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 [39]. The encoded load factor is embedded in the two-bit ECN field in the IP header.

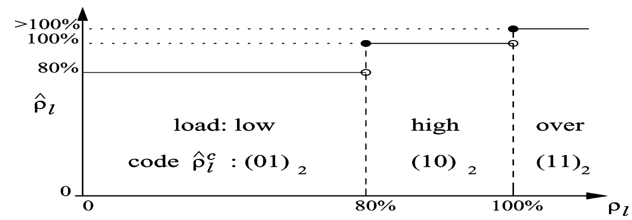


Fig. 2. The quantized load factor $\hat{\rho}_l$ at a link l is a non-decreasing function of the raw load factor ρ_l and can be represented by a two-bit code $\hat{\rho}_l^c$.

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 [38] 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 [2], [15], [19], [24], [29]

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

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 [2], [15] (we set $\kappa_q = 0.5$),[1] γ_l is the target utilization [29] (set to a value close to 1, *e.g.,* we use 0.98), 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).

C. 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 III-D.

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

$$\texttt{MI}: cwnd(t + rtt) = cwnd(t) \times (1 + \xi) \tag{2}$$
$$\texttt{AI}: cwnd(t + rtt) = cwnd(t) + \alpha \tag{3}$$
$$\texttt{MD}: cwnd(t + \delta t) = cwnd(t) \times \beta. \tag{4}$$

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

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

transition point and the MD parameter β, we chose $\beta = 0.875$ (see Section III-B). We use $\alpha = 1.0$ as is in TCP [17].

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 [25]. 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

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

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 V), 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 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.

D. 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 III-C, 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]

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

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

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, the influence of RTT heterogeneity on VCP flow's throughput is not as much as on TCP's [30], [36] (see Section IV-B).

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 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 (4), 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 (since there should be no congestion any more after the MD cut) 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 Mth 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

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

E. Protocol Description

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

1) 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 $\hat{\rho}_l^c$ in the IP header of each outgoing data packets if it is larger than the one, $\hat{\rho}_p^c$, carried by the packets 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:

$$\lambda_l \leftarrow \lambda_l + s_p \qquad // \, Count.$$

[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, (7) is obtained by solving $1 + \alpha = 1 + \frac{t_\rho}{rtt}\,\alpha_s$.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

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

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

where $a = 0.875$ and $t_q = 10$ ms. Like RED, VCP maintains a low-pass filtered queue length $\widetilde{q_l}$ using exponentially weighted moving average;

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

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

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 III-B;

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

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

2) 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., $(01)_2$. 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-load factor $\hat{\rho}_p^c = (01)_2$, per ACK:

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

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:

$$\begin{aligned} inc &= \alpha \cdot \min((\frac{srtt}{t_\rho})^2, \sigma_{ai}) && // \, Scaling \\ cwnd &\leftarrow cwnd + inc/cwnd && // \, AI \end{aligned}$$

where $\alpha = 1.0$, the AI scaling limiter $\sigma_{ai} = 10.0$ (to bound the bursty traffic introduced due to the AI scaling);

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:

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

where $\beta = 0.875$.

TABLE I
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

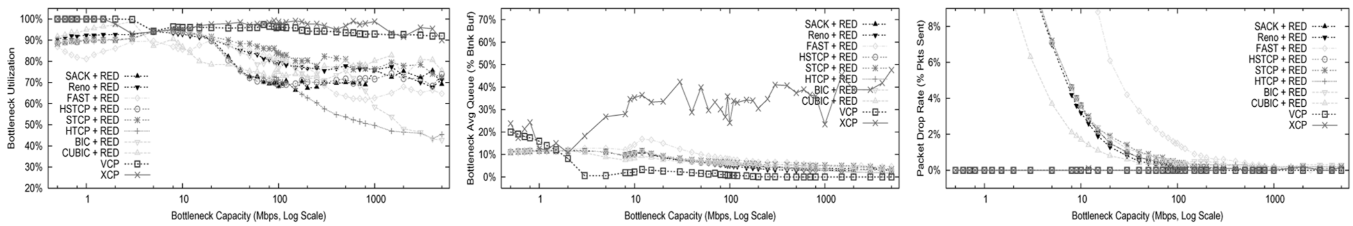
Table I summarizes the VCP router (upper part) and end-host (lower part) parameters and their typical values.

IV. PERFORMANCE EVALUATION

In this section, we use extensive ns2 simulations to evaluate the performance of VCP for a wide range of network scenarios including varying the link capacities in the range $[500\text{ kbps}, 5\text{ Gbps}]$, round trip times in the range $[1\text{ ms}, 1.5\text{ s}]$, numbers of long-lived, FTP-like flows in the range $[1, 1000]$, and arrival rates of short-lived, web-like flows in the range $[10\text{ s}^{-1}, 1000\text{ s}^{-1}]$. We always use two-way traffic with congestion resulted in the reverse path. The bottleneck buffer size is set to one bandwidth-delay product. The data packet size is 1000 bytes, while the ACK packet is 40 bytes. All simulations are run for at least 120 s 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 500 ms interval, while queue length and congestion window are sampled every 10 ms. Throughout all the simulations in this paper, we use the *same* set of parameter values listed in Table I. This suggests that VCP is robust in a large variety of environments.

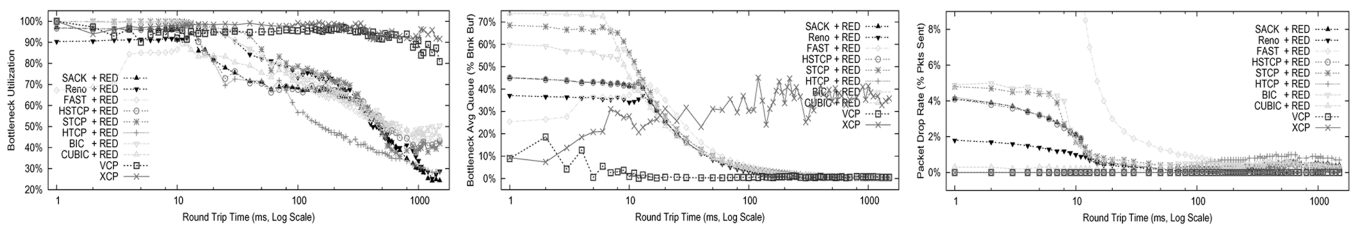
For comparison purpose, we also run simulations for other schemes including TCP Reno [1], SACK [34], HSTCP [13], HTCP [31], STCP [26], FAST [22], BIC [48], CUBIC [41], and XCP [25], under the same network and traffic settings. Except for XCP which has its own router algorithm, we run RED [15] with ECN enabled in the bottleneck routers for each of the above schemes. The protocol parameter settings of these TCP schemes (including RED) are those recommended by their respective authors. We choose to run RED/ECN because both VCP and XCP require router support; it is unfair to compare them with the TCP schemes without AQM at the routers. Without AQM, these TCP schemes will result in much larger bottleneck queue length than otherwise. As stated in Section I, since the Internet is increasingly used to carry real-time traffic like VoIP, keeping the bottleneck queues small is essential and should be a requirement for the future congestion control protocols.

The simulation results demonstrate that, for a wide range of scenarios, VCP and XCP significantly outperforms the other eight schemes. VCP achieves comparable performance to XCP, *i.e.,* exponential convergence to high utilization, negligible packet drop rate, low persistent queue and reasonable fairness,



(a) (b) (c)

Fig. 3. One bottleneck with the capacity varying from 500 kbps to 5 Gbps. Note the logarithmic scale of the x-axis. (a) Bottleneck Average Utilization. (b) Bottleneck Average Queue. (c) Packet Drop Rate.



(a) (b) (c)

Fig. 4. One bottleneck with the round-trip propagation delay ranging from 1 ms to 1500 ms. (a) Bottleneck average utilization. (b) Bottleneck average queue. (c) Packet drop rate.

except its significantly slower fairness convergence speed than XCP. In other words, the one extra bit of congestion information feedback of VCP does bring significant performance gain.

A. 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 and the arrival rate of flows on the performance of VCP. The basic setting is an 150 Mbps link with 80 ms 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. The simulation results are consistent; the deviation is generally within a few percent of the average.

1) Varying Bottleneck Capacity: As illustrated in Fig. 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 capacities varying from 500 kbps to 5 Gbps. (*Note the logarithmic scale of the x-axis of the figures in this and the next subsections.*) 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%–47% buffer size). This is because that the VCP router gives higher priority to ACK packets than data packets, as described in Section III-E. 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 200 Mbps, the bottleneck utilization mostly drops to around 70%–80% (even less than 60% for HTCP). When the capacity is below 5 Mbps, all these schemes result in higher than 4% packet loss rate.

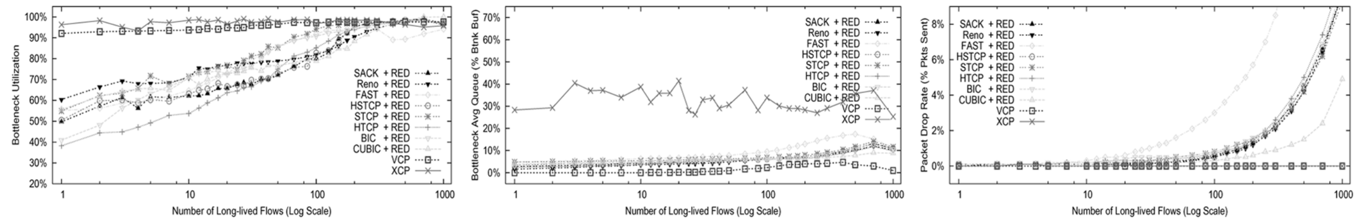
2) Varying Feedback Delay: We fix the bottleneck capacity at 150 Mbps and vary the round-trip propagation delay from 1 ms to 1.5 s. As shown in Fig. 4, we notice that, in most cases, VCP's bottleneck utilization is higher than 90%, and the average bottleneck queue is less than 2% buffer size. We also observe that VCP's RTT parameter scaling is more sensitive to very low values of RTT (e.g., < 5 ms), thereby causing the average queue length to grow to about 10%–20% buffer size. For the RTT values larger than 800 ms, VCP obtains lower utilization (80%–90%) since the link load factor measurement interval $t_\rho = 200$ ms is much less than the flow RTTs. 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;[3] the tradeoff is that the link load measurement will be less responsive. 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 lower queue length) and it performs significantly better than all the other TCP variants, which achieve lower than 75% bottleneck utilization when the RTT is higher than 200 ms.

3) 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 Fig. 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), VCP's 90-percentile queue is still less than 20% of the buffer size. Its average queue is consistently less than 5% buffer size across all the simulation cases.

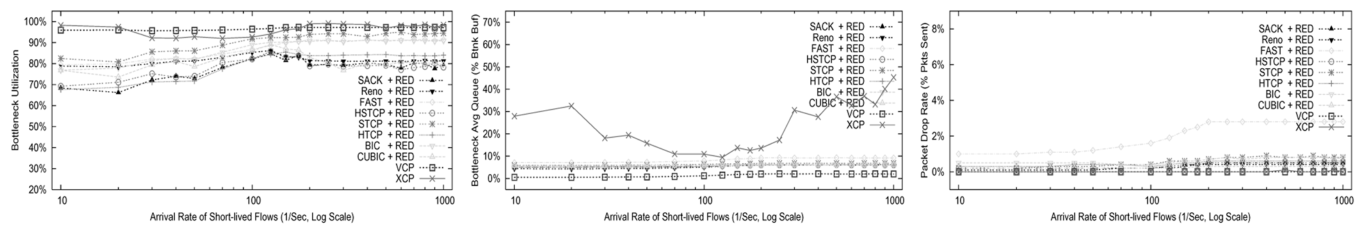
For high per-flow BDP scenarios, where there is no more than 10 flows on the forward path (*i.e.,* the per-flow BDP is 150–1500

[3]For all the cases where RTT is larger than 800 ms, by simply increasing the link load factor measurement interval t_ρ from 200 ms to 500 ms, VCP's average bottleneck utilization boosts to higher than 90% with the average queue lower than 2% bottleneck buffer size.



(a) (b) (c)

Fig. 5. One bottleneck with the number of long-lived, FTP-like flows increasing from 1 to 1000. (a) Bottleneck Average Utilization, (b) Bottleneck Average Queue, (c) Packet Drop Rate.



(a) (b) (c)

Fig. 6. One bottleneck with short-lived, web-like flows arriving/departing at a rate from 10/s to 1000/s. (a) Bottleneck Average Utilization, (b) Bottleneck Average Queue, (c) Packet Drop Rate.

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.

4) 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 10/s to 1000/s. Their transfer size obeys the Pareto distribution with an average of 30 packets. As shown in Fig. 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.

B. RTT Fairness

TCP flows with different RTTs achieve bandwidth allocations that are proportional to $1/rtt^z$ where $1 \leq z \leq 2$ [30]. 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 150 Mbps bottleneck, with 30 FTP flows on the reverse path. Each forward flow i's RTT value $rtt_i = 40 + i * rtt_{delta}$ ms for $i = 0, 1, \ldots, 29$. We perform eleven sets of simulations with rtt_{delta} increasing from $0, 1, 2, \ldots,$ to 10 ms. When $rtt_{delta} = 0$ ms, all the flows' RTTs equal to 40 ms; As rtt_{delta} increases to 4 ms, the RTTs are in the range of $[40 \text{ ms}, 156 \text{ ms}]$ with the RTT ratio of about 4; When $rtt_{delta} = 10$ ms, the RTTs are in the range of $[40 \text{ ms}, 330 \text{ ms}]$ with the RTT ratio of more than 8.

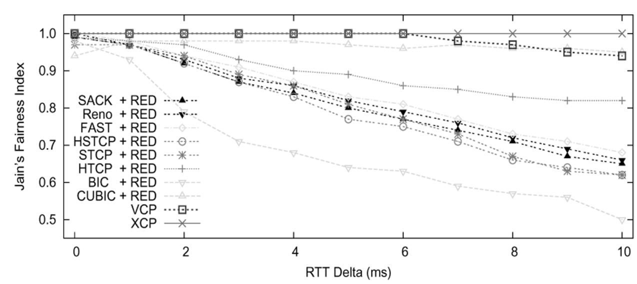


Fig. 7. Jain's fairness index under scenarios of one bottleneck link shared by 30 flows, whose RTTs are in the ranges varying from $[40 \text{ ms}, 40 \text{ ms}]$ to $[40 \text{ ms}, 330 \text{ ms}]$.

Fig. 7 shows that, in terms of RTT fairness, XCP performs the best by achieving Jain's fairness index [21] [4] of 1.0 across the whole set of simulations, closely followed by VCP (0.94–1.0) and CUBIC (0.94–0.98). All the other seven schemes fall short of distributing bandwidth fairly among flows with heterogenous RTTs.[5] 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 (with RED) achieves 88% average bottleneck utilization and 10%-bottleneck-buffer-size 90-percentile queue.

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 III-D. To prevent sudden traffic bursts from such VCP flows which can cause

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

[5] It is debatable if we must 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 III-D) to achieve bandwidth allocation proportional to RTT^{-1}.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

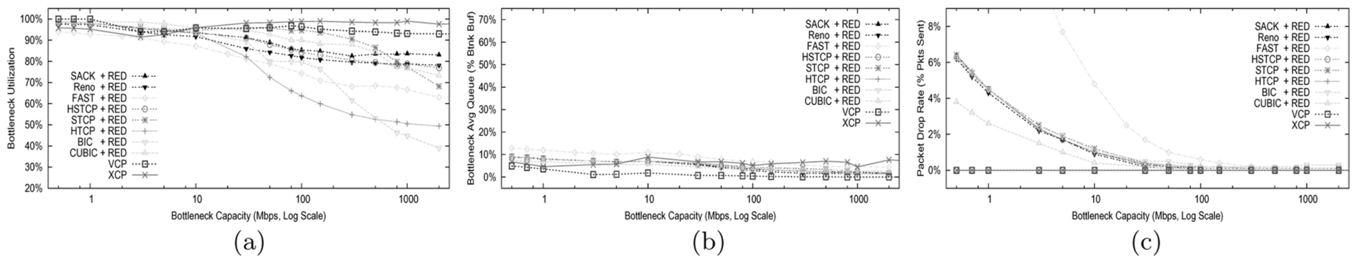


Fig. 8. Seven bottlenecks with the capacity varying from 500 kbps to 2 Gbps. (a) Bottleneck Average Utilization, (b) Bottleneck Average Queue, (c) Packet Drop Rate.

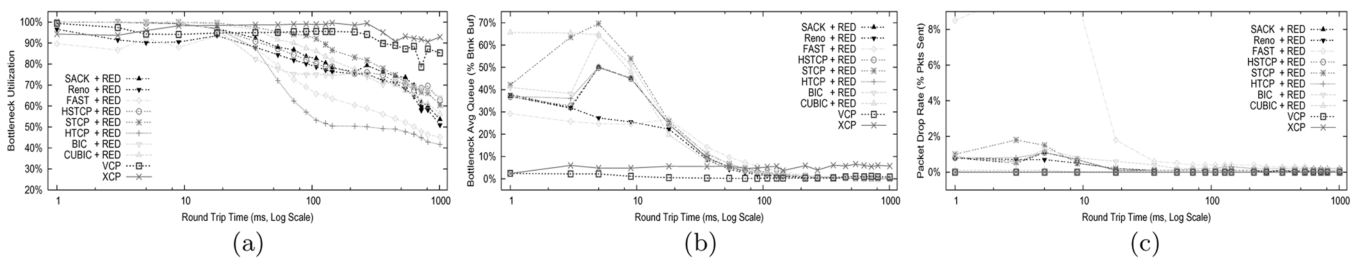


Fig. 9. Seven bottlenecks with the longest path's round-trip propagation delay ranging from 1 ms to 1000 ms. (a) Bottleneck Average Utilization, (b) Bottleneck Average Queue, (c) Packet Drop Rate.

the bottleneck instantaneous queue to increase substantially, we place bounds on the actual values of these parameters (see Section III-E). These bounds restrict the throughput of flows with very high RTTs.

C. Multiple Bottlenecks

Now, 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 orders of magnitude.

1) Varying Bottleneck Capacity: First, we set all the bottleneck links' one-way propagation delay to 5 ms. The longest path's round-trip propagation delay is 80 ms. We vary all the bottleneck links' bandwidth from 500 kbps to 2 Gbps. Fig. 8 shows that, for all the cases, VCP performs as good as in the single-bottleneck scenarios. It 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.

In comparison to XCP, one key difference is that *VCP penalizes flows that traverse more bottlenecks.* For example, when the bottlenecks' capacity is 150 Mbps, VCP allocates 2.7 Mbps to each long flow that traverses all the seven bottlenecks, and 10.0 Mbps to each cross flow that passes one bottleneck; while under XCP, each long flow gets 3.8 Mbps and each cross flow 4.4 Mbps. So comparing to XCP, VCP's bandwidth allocation is more sensitive to the number of bottlenecks that a flow traverses. We discuss the reason behind this in Section V. All the other TCP variants exhibit similar behavior as VCP.

2) Varying Path RTT: Second, we fix all the bottlenecks' bandwidth to 150 Mbps and vary the longest path's round-trip propagation delay from 1 ms to 1 s. Again, VCP and XCP outperform all the other schemes, as shown in Fig. 9. 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 outperform the other TCP variants.

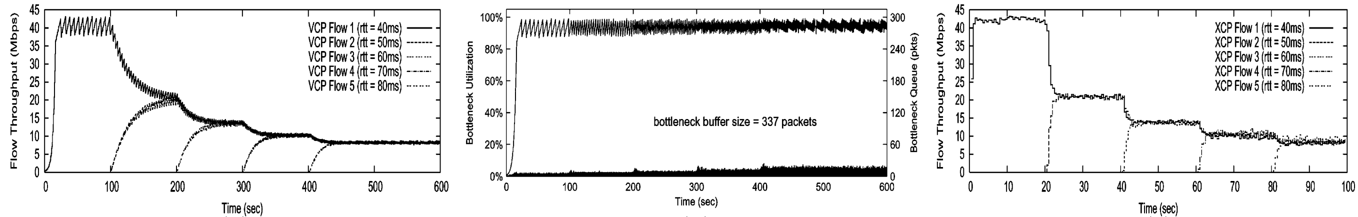
D. Dynamics

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.

1) Convergence Behavior: To study the convergence behavior of VCP, we revert to the single bottleneck link with a bandwidth of 45 Mbps where we introduce 5 flows into the system, one after another, with starting times separated by 100 s. We also set the RTT values of the five flows to different values. The reverse path has 5 flows that are always active. Fig. 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 V.

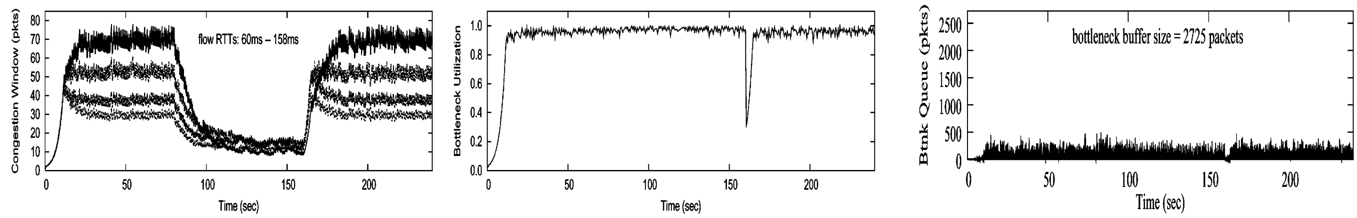
2) Sudden Demand Change: We illustrate how VCP reacts to sudden changes in traffic demand using a simple simulation. Consider an initial setting of 50 forward FTP flows with varying RTTs (uniformly chosen in the range $[60\text{ ms}, 158\text{ ms}]$) sharing a 200 Mbps bottleneck link. There are 50 FTP flows on the reverse path. At $t = 80$ s, 150 new forward FTP flows become active; then they leave at 160 s. Fig. 11 clearly shows

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



(a) (b) (c)

Fig. 10. VCP converges onto good fairness, high utilization and small queue. However, VCP's sub-linear fairness convergence time is significantly longer than XCP's logarithmic time. (a) VCP Flow Throughput, (b) VCP Bottleneck Utilization/Queue, (c) XCP Flow Throughput.



(a) (b) (c)

Fig. 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 Congestion Window, (b) Bottleneck Utilization, (c) Bottleneck Queue.

that VCP can adapt sudden fluctuations in the traffic demand. (Fig. 11(a) 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 $t = 160$ s, 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 decrease/increase 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 V).

Besides what has been discussed so far, it is also straight-forward to use VCP to provide differentiated bandwidth to competing flows by augmenting (8) with a weight factor. We refer the reader to [47] for more details.

V. 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 approximation model. We consider a single bottleneck with infinite buffer traversed by N synchronous flows that have the same, constant 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)–(4) in Section III-C:

$$\dot{w}_i(t) = \frac{1}{T} \cdot [\, w_i(t) \cdot \xi(\rho(t)) + \alpha \,] \tag{9}$$

with the MI parameter

$$\xi(\rho(t)) = \kappa \cdot \frac{1 - \rho(t)}{\rho(t)} \tag{10}$$

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 $\xi(\rho(t))$.

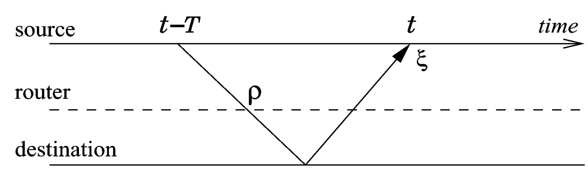


Fig. 12. A simplified VCP model. The source sending rate at time $t - T$ is used by the router to calculate a load factor ρ, which is echoed back from the destination to the source at time t. Then the source adjusts its MI parameter $\xi(\rho(t))$ based on the load factor $\rho(t)$.

As shown in Fig. 12, the load factor $\rho(t)$ received by the *source* at a time t is computed based on the sender's rate at time $t - T$, i.e.,

$$\rho(t) = \frac{\sum_{i=1}^{N} w_i(t-T)}{\gamma C T} \tag{11}$$

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.[6]

Using (9) to sum over all N flows yields

$$\dot{w}(t) = \frac{1}{T} \cdot [\, w(t) \cdot \xi(\rho(t)) + N\alpha \,] \tag{12}$$

[6]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.

where $w(t) = \sum_{i=1}^{N} w_i(t)$ is the sum of all the congestion windows. This result, together with (10) and (11), leads to

$$\dot{w}(t) = \frac{1}{T} \cdot \{ \kappa \cdot w(t) \cdot [\, \frac{\gamma CT}{w(t-T)} - 1\,] + N\alpha \} \tag{13}$$

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 [47], we prove the following global stability result.

Theorem 1: Under the model (9), (10), and (11) 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 (13) 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 (11) 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.,

$$\gamma(t + T_\gamma) = \gamma(t) + \mathtt{sgn}\,(\gamma^* - \widetilde{\gamma}^*(t)) \cdot \delta\gamma \tag{14}$$

where $T_\gamma \gg T$, $sgn(\cdot)$ is the sign function, $\widetilde{\gamma}^*(t)$ is a low-pass filtered link utilization which is very easy to measure at the router, and $0 < \delta\gamma \ll \gamma^*$ is a constant stepsize (*e.g.,* 0.01). This adjustment process will stop if and only if the target utilization γ^* has been achieved.

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

$$\xi(\rho^i(t)) = \kappa \cdot [\, \frac{1}{\rho^i(t)} - 1\,] \tag{15}$$

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 [47].

Theorem 2: In a multiple-bottleneck topology where all flows have the same round-trip time, if there exists a unique equilibrium, then the algorithm defined by (9) and (15) 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 $\rho_l^* = \frac{\sum_{i \in I_l} w_i^*}{\gamma CT}$.

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 throughput.

Having established the stability and fairness properties of the MIAIMD model, we now turn our attention to the convergence of the VCP protocol. The following two theorems, proved in [47], 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: T*he 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 ΔP is the largest congestion window difference between flows sharing that link ($\Delta P > 1$).*

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, VCP tends to allocate more bandwidth to flows that traverse fewer bottleneck links (see Section IV-C). This is because, given only two ECN bits to represent the load factor, some amount of information is lost in the load factor quantization process (Section III-B).

VI. Discussions

Since VCP switches between MI, AI, and MD modes 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 now discuss these concerns as well as VCP's TCP-friendliness and incremental deployment issues.

A. Stability Under Heterogeneous Delays

Although the MIAIMD model presented in Section V 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 III-D.

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 IV suggest that VCP is indeed stable over a large variety of network scenarios including per-flow bandwidths from 16.7 kbps to 167 Mbps and RTTs from 1 ms to 1.5 s.

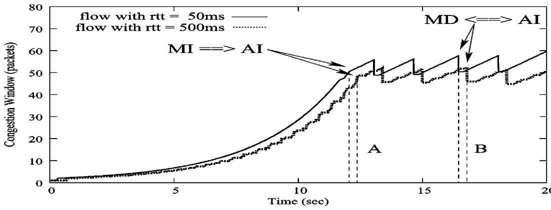


Fig. 13. The congestion window dynamics of two flows with dramatically different RTTs (50 ms versus 500 ms). Due to its longer delay, the larger-RTT flow always slides its mode later than the one with smaller RTT (see the regions labeled as A and B). However, the effect of this asynchronous switching is accommodated by VCP and does not prevent it from maintaining stability and achieving efficiency and fairness.

B. Influences of Mode Sliding

From an efficiency perspective, VCP's goal is to bring and maintain the system in the high utilization region. While MI enables VCP to quickly reach 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 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 III-E).

As discussed in Section III-D, 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) and (8) in Section III-D]. To address the second issue, VCP bounds the MI gain, as discussed in Section III-E. To illustrate the effectiveness of limiting the MI gain, Fig. 13 shows the congestion window evolution for two flows with RTTs of 50 ms and 500 ms, respectively, traversing a single 10 Mbps link. At time 12.06 s, the 50 ms-RTT flow switches from MI to AI. In contrast, due to its larger RTT, the 500 ms-RTT flow keeps performing MI until time 12.37 s. However, because VCP limits the MI gain of the 500 ms-RTT flow, the additional bandwidth acquired by this flow during the 0.31 s interval is only marginal when compared to the bandwidth acquired by the 50 ms-RTT flow.

C. 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 [36]. 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 $\alpha = \frac{3(1-\beta)}{1+\beta} = 0.2$ for $\beta = 0.875$ according to the TCP-friendly general AIMD formula [49]. 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 [15]. 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.

D. Incremental Deployment

If VCP is to be gradually deployed on the Internet, the deployment could follow the similar path as CSFQ [42] 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 one 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.

VII. Related Work

This paper builds upon a great body of related work, particularly XCP [25], TCP [1], [14], [17], [34], AIMD [8], [21], AQM [2], [15], [29] and ECN [39], [40]. Congestion control is pioneered by TCP and AIMD. The research on AQM starts from RED [15], [32], followed by Blue [12], REM [2], PI controller [16], AVQ [29], and CHOKe [37], 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 [19] and [24]) previously proposes explicit rate control. VCP does learn from these schemes. Nevertheless, 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 [20]. The link load factor is suggested as a congestion signal in [19], based on which VCP quantizes and encodes it for a more compact representation for the degree of congestion. MaxNet [46] uses the maximal congestion information among all the bottlenecks to achieve max-min fairness. Quick-Start [18] occasionally uses several bits per packet to quickly ramp up source sending rates. VCP is complementary to Quick-Start as it constantly uses two bits per packet.

Congestion Notification Based Schemes: For high BDP networks, according to [25], the performance gap between XCP and $\mathrm{TCP} + \mathrm{RED/REM/AVQ/CSFQ}$ [42] with one-bit ECN support seems large. VCP's performance is comparable to

XCP's, as shown in Section IV. 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 (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 [10], which is very different from the way VCP uses. GAIMD [49] 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 [13] extends the standard TCP by adaptively setting the increase/decrease parameters according to the congestion window size. HTCP [31] employs an adaptive AIMD with its parameters set as functions of the elapsed time since the last congestion event. Adaptive TCP [28] also applies dynamic AIMD parameters with respect to the changing network conditions. STCP [26] changes to a fixed MIMD algorithm. FAST [22] uses queuing 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/CUBIC [41], [48] 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 [7] 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 IV show that, even with AQM/ECN support from network, these schemes still cannot achieve similar performance as VCP does in high BDP networks.

Analytical Results: The nonlinear optimization framework [27] 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 [23], [44] and then extended to the case of heterogeneous delays by [33]. The local stability of a modified algorithm for the case of heterogeneous delays is proved by [51], which establishes a model that is similar to what we show in Section V. 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 [9], [45], and [50].

Variable-structure control with sliding modes has a long history in control theory [11], [43]. It is useful when a set of features are desired in a system but no single algorithm can provide all of them. Our work can be viewed as an application of this idea to network congestion control.

VIII. 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, negligible packet loss rate, low persistent bottleneck queue, and reasonable fairness. 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 synchronous, 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. 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.

Acknowledgment

The authors would like to thank the Editor and the anonymous reviewers of this journal, the SIGCOMM reviewers, X. Fan, S.Floyd, D. Harrison, J. Hu, D. Joseph, J. Kannan, D. Katabi, Y. Kuang, K. K. Ramakrishnan, S. Shenker, L. Shi, G. Wang, and J. Wen for their comments that greatly help improve the quality of this paper.

References

[1] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IETF RFC 2581, Apr. 1999.

[2] S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, May 2001.

[3] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms," *Proc. INFOCOM'01*, Apr. 2001.

[4] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. New York: Simon & Schuster, 1991.

[5] S. Bhandarkar, S. Jain, and A. Reddy, "Improving TCP performance in high bandwidth high RTT links using layered congestion control," *Proc. PFLDNet'05*, Feb. 2005.

[6] L. Brakmo and L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE J. Sel. Areas Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[7] C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi, and R. Wang, "TCP Westwood: End-to-end congestion control for wired/wireless networks," *Wireless Netw. J.*, vol. 8, no. 5, pp. 467–479, Sep. 2002.

[8] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *J. Comput. Netw. ISDN*, vol. 17, no. 1, pp. 1–14, Jun. 1989.

[9] S. Deb and R. Srikant, "Global stability of congestion controllers for the Internet," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 1055–1060, Jun. 2003.

[10] A. Durresi, M. Sridharan, C. Liu, M. Goyal, and R. Jain, "Multilevel explicit congestion notification," *Proc. SCI'01*, Jul. 2001.

[11] C. Edwards and S. Spurgeon, *Sliding Mode Control: Theory and Applications*. New York: Taylor and Francis, Aug. 1998.

[12] W. Feng, K. Shin, D. Kandlur, and D. Saha, "The BLUE active queue management algorithms," *IEEE/ACM Trans. Netw.*, vol. 10, no. 8, pp. 513–528, Aug. 2002.

[13] S. Floyd, "Highspeed TCP for large congestion windows," IETF RFC 3649, Dec. 2003.

[14] S. Floyd and T. Henderson, "The newreno modification to TCP's fast recovery algorithm," IETF RFC 2582, Apr. 1999.

[15] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 8, pp. 397–413, Aug. 1993.
[16] C. Hollot, V. Misra, D. Towlsey, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," *Proc. INFOCOM'01*, Apr. 2001.
[17] V. Jacobson, "Congestion avoidance and control," *Proc. SIGCOMM'88*, Aug. 1988.
[18] A. Jain and S. Floyd, "Quick-start for TCP and IP," *IETF Internet Draft Draft-Amit-Quick-Start-02.txt*, Oct. 2002.
[19] R. Jain, S. Kalyanaraman, and R. Viswanathan, "The OSU scheme for congestion avoidance in ATM networks: Lessons learnt and extensions," *Perf. Eval.*, vol. 31, no. 1, pp. 67–88, Nov. 1997.
[20] R. Jain and K. K. Ramakrishnan, "Congestion avoidance in computer networks with a connectionless network layer: Concepts, goals, and methodology," in *Proc. IEEE Computer Netw. Symp.*, Apr. 1988.
[21] R. Jain, K. K. Ramakrishnan, and D. Chiu, "Congestion avoidance in computer networks with a connectionless network layer," *DEC-TR-506*, Aug. 1987.
[22] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proc. INFOCOM*, Mar. 2004.
[23] R. Johari and D. Tan, "End-to-end congestion control for the Internet: Delays and stability," *IEEE/ACM Trans. Netw.*, vol. 9, no. 12, pp. 818–832, Dec. 2001.
[24] 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. Netw.*, vol. 8, pp. 87–98, Feb. 2000.
[25] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," *Proc. SIGCOMM'02*, Aug. 2002.
[26] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM Computer Commun. Rev.*, vol. 32, no. 2, Apr. 2003.
[27] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, pp. 237–252, 1998.
[28] A. Kesselman and Y. Mansour, "Adaptive TCP flow control," *PODC'03*, Jul. 2003.
[29] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," *Proc. SIGCOMM'01*, Aug. 2001.
[30] T. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 336–350, Jun. 1997.
[31] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," *Proc. PFLDnet'04*, Feb. 2004.
[32] D. Lin and R. Morris, "Dynamics of random early detection," *Proc. SIGCOMM'97*, Aug. 1997.
[33] L. Massoule, "Stability of distributed congestion control with heterogeneous feedback delays," *IEEE Trans. Autom. Control*, vol. 47, no. 6, pp. 895–902, Jun. 2002.
[34] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," IETF RFC 2018, Oct. 1996.
[35] Network Simulator ns-2 [Online]. Available: Http://www.isi.edu/nsnam/ns/
[36] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *Proc. SIGCOMM'98*, Sep. 1998.
[37] R. Pan, K. Psounis, and B. Prabhakar, "CHOKe, a stateless active queue management scheme for approximating fair bandwidth allocation," *Proc. INFOCOM'00*, Mar. 2000.
[38] V. Paxson, "End-to-end Internet packet dynamics," *Proc. SIGCOMM'97*, Sep. 1997.
[39] K. K. Ramakrishnan and S. Floyd, "The addition of explicit congestion notification (ECN) to IP," IETF RFC 3168, Sep. 2001.
[40] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," *Proc. SIGCOMM'88*, Aug. 1988.
[41] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *Proc. PFLDNet'05*, Feb. 2005.
[42] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," *Proc. SIGCOMM'98*, Sep. 1998.
[43] V. Utkin, "Variable structure systems with sliding modes," *IEEE Trans. Autom. Control*, vol. 22, no. 2, pp. 212–222, Apr. 1977.
[44] G. Vinnicombe, "On the stability of end-to-end congestion control for the Internet," Univ. Cambridge, Cambridge, U.K., Tech. Rep. CUED/F-INFENG/TR.398, 2000.
[45] J. Wen and M. Arcak, "A unifying passivity framework for network flow control," *IEEE Trans. Autom. Control*, vol. 49, no. 2, pp. 162–174, Feb. 2004.
[46] B. Wydrowski and M. Zukerman, "MaxNet: A congestion control architecture for maxmin fairness," *IEEE Commun. Lett.*, vol. 6, no. 11, pp. 512–514, Nov. 2002.
[47] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," Jun. 2005, U.C. Berkeley, Tech Report, [Online]. Available: http://networks.ecse.rpi.edu/~xiay/pub/vcp_tr.pdf.
[48] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," *Proc. INFOCOM'04*, Mar. 2004.
[49] Y. Yang and S. Lam, "General AIMD congestion control," *Proc. ICNP'00*, Nov. 2000.
[50] L. Ying, G. Dullerud, and R. Srikant, "Global stability of Internet congestion controllers with heterogeneous delays," in *Proc. Amer. Control Conf.*, Jun. 2004.
[51] Y. Zhang, S. Kang, and D. Loguinov, "Delayed stability and performance of distributed congestion control," *Proc. SIGCOMM'04*, Sep. 2004.

Yong Xia (S'03–M'04) received the B.E. degree from Huazhong University of Science and Technology, Wuhan, China, the M.E. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1994, 1998, and 2004, respectively.

He is a Research Manager at NEC Laboratories, China, where he works on mobile networking. Previously, he was with Microsoft and China Telecom. He was a Visiting Research Student at the University of California, Berkeley, in the summer 2003. He is a member of the ACM.

Lakshminarayanan Subramanian received the B.Tech. degree from IIT-Madras, India, and the M.S. and Ph.D. degrees from the University of California at Berkeley.

He is an Assistant Professor in the Courant Institute of Mathematical Sciences, New York University, New York. His research focuses primarily in the areas of network security, routing protocols, congestion control, Internet architecture and technologies for developing countries.

Ion Stoica received the Ph.D. degree from the Carnegie Mellon University, Pittsburgh, PA, in 2000.

He is an Associate Professor in the Electrican Engineering and Computer Science Department, University of California, Berkeley, where he does research on peer-to-peer network technologies in the Internet, resource management, and network architectures.

Dr. Stoica is the recipient of a Sloan Foundation Fellowship (2003), a Presidential Early Career Award for Scientists and Engineers (PECASE) (2002), and the ACM Doctoral Dissertation Award (2001). He is a member of the ACM.

Shivkumar Kalyanaraman (S'93–M'97) received the B.Tech. degree in computer science from IIT, Madras, India, and the M.S. and Ph.D. degrees from Ohio State University, Columbus.

He is a Professor at the Department of Electrical, Computer and Systems Engineering at Rensselaer Polytechnic Institute, Troy, NY. His research interests include various traffic management topics and protocols for emerging tetherless networks. He is a member of the ACM.