Available online at www.sciencedirect.com

ScienceDirect

Performance Evaluation 65 (2008) 417–440

PERFORMANCE EVALUATION
An International Journal

www.elsevier.com/locate/peva

TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks

Shao Liu*, Tamer Başar, R. Srikant

Department of Electrical and Computer Engineering and Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1308 West Main Street, Urbana, IL 61801-2307, USA

Received 31 October 2007; accepted 7 December 2007
Available online 28 December 2007

Abstract

We introduce a new congestion control algorithm for high-speed networks, called TCP-Illinois. TCP-Illinois uses packet loss information to determine whether the window size should be increased or decreased, and uses queueing delay information to determine the amount of increment or decrement. TCP-Illinois achieves high throughput, allocates the network resource fairly, and is incentive compatible with standard TCP. We also build a new stochastic matrix model, capturing standard TCP and TCP-Illinois as special cases, and use this model to analyze their fairness properties for both synchronized and unsynchronized backoff behaviors. We finally perform simulations to demonstrate the performance of TCP-Illinois.
© 2008 Published by Elsevier B.V.

Keywords: Congestion control; TCP; Fairness; Stability; Synchronization

1. Introduction

TCP-Reno [13], TCP-NewReno [10], and SACK TCP [22] are the standard versions of TCP congestion control protocols currently deployed in the Internet, and they have achieved great success in performing congestion avoidance and control. The key feature of standard TCP is its congestion avoidance phase, which uses the additive increment multiplicative decrement (AIMD) algorithm [12]. Being a window-based algorithm, TCP controls the send rate by maintaining a window size variable W, which limits the number of unacknowledged packets in the network from a single user. This window size is adjusted by the AIMD algorithm in the following manner: W is increased by α/W ($\alpha = 1$ for standard setting) for each ACK, and thus is increased by a constant α/b per round trip time (RTT) if all the packets are acknowledged within an RTT, where b is the number of packets acknowledged by each ACK ($b = 1$ for original TCP, and $b = 2$ for delayed ACK [30]). On the other hand, W is decreased by a fixed proportion βW ($\beta = 1/2$ for standard setting) once some packets are detected to be lost in the last RTT.[1] Under this algorithm, senders gently probe the network for spare bandwidth by cautiously increasing their send rates, and sharply reduce their send

* Corresponding author. Tel.: +1 217 621 1632.
E-mail addresses: shaoliu@uiuc.edu (S. Liu), basar1@uiuc.edu (T. Başar), rsrikant@uiuc.edu (R. Srikant).

[1] Within one RTT, W may decrease multiple times in Reno and can decrease only once in NewReno and SACK.

0166-5316/$ - see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.peva.2007.12.007

rates when congestion is detected. Along with other features like slow start, fast recovery, and fast retransmission, TCP achieves congestion control successfully in the current low-speed networks.

However, the current TCP can perform poorly in networks with high-bandwidth-delay-product (BDP) paths, since the AIMD algorithm, being very conservative, is not designed for large window size flows. First, it takes too long time for a large window size user to recover after a backoff and the bandwidth is not effectively utilized [9]. Second, TCP's time average window size $\bar{W}$ is related with the loss event probability[2] p in the following manner [25]

$$\bar{W} \approx \sqrt{3/2bp} \quad \text{or} \quad p \approx \frac{3}{2b(\bar{W})^2}. \tag{1}$$

Since TCP interprets all packet losses as congestion signals, $\bar{W}$ is upper bounded by $\sqrt{3/2bp_t}$, where p_t is the transmission error rate [9]. p_t is around 10^{-7} in optical fiber networks, and much higher in other lossy networks, like wireless networks. So TCP, and its AIMD algorithm in particular, should be modified in high-bandwidth-delay-product networks.

Several alternatives to current versions of TCP have been proposed for implementation in high-speed networks. Some require the modification to router algorithms also, like XCP [15], and some modify the sender side only, like HS-TCP [9], Scalable TCP [16], TCP-Westwood [36], H-TCP [18], BIC-TCP [35], CUBIC-TCP [27], TCP-Vegas [8], FAST TCP [14], Adaptive Reno [28], and Compound-TCP [31]. Although each of these has shown advantages over standard TCP in some aspects, none of them have yet provided convincing evidence that they are overwhelmingly better than standard TCP and are very suitable for general deployment.

In this paper, we first list some desirable design specifications that a high-speed TCP variant should meet, and then introduce the TCP-Illinois algorithm, which uses packet loss information as the *primary* congestion signal to determine the *direction* of window size change (whether window size should be increased or decreased), and uses queueing delay information as the *secondary* congestion signal to adjust the *pace* of window size change (the amount of window size increment or decrement). We then show that TCP-Illinois satisfies all the design requirements we listed, and outperforms standard TCP and many other TCP variants.

To study the fairness, stability, and responsiveness properties of TCP-Illinois, we extend the stochastic matrix model [2–5,17,29,34] by allowing window size backoff probabilities to be functions of flow arrival rates at congestion events. Our contribution to this modeling technique includes the following: (i) we show that a large class of general AIMD algorithms, including standard TCP and TCP-Illinois, have similar fairness properties, and the fairness properties only depend on the backoff behaviors for these algorithms; (ii) the backoff behavior can be characterized by a function $f(\cdot)$, where $f(\cdot)$ is a user's backoff probability as a function of its send rate: if $f(\cdot) \equiv 1$, the backoffs of all the users are completely synchronized; if $f(\cdot)$ is linear, the backoffs are completely unsynchronized; and in general, the partially-unsynchronized backoff lies in the middle; (iii) $f(\cdot)$ is determined by the number of packets dropped in each congestion event: heavy congestion causes synchronization and light congestion leads to unsynchronized backoff; (iv) the heaviness of congestion (the number of packets dropped in one congestion event as compared to the number of flows) depends on the window size increment of these flows just before the congestion event, and thus a smaller (respectively, larger) increment before a congestion event causes the backoff to be more unsynchronized (respectively, synchronized).

The paper is structured as follows. In Section 2, we list the requirements for a new version of TCP, compare the existing protocols and point out their shortcomings, and describe our design objective. We next introduce the TCP-Illinois protocol in Section 3, and study its fairness and stability properties using a new stochastic matrix model for a class of general AIMD algorithms in Section 4. We further explore some other properties of TCP-Illinois in Section 5 and provide *ns-2* simulation results in Section 6 for a comparative study of TCP, HS-TCP and TCP-Illinois.

2. Background and motivation

As we have mentioned above, several new protocols have been introduced to replace standard TCP in high-speed networks. To compare these protocols and to provide insight into the development of an ideal protocol, we list below some requirements that a new protocol should satisfy. This list broadens the list of requirements in [18].

[2] All the packet losses within one RTT are regarded as one loss event. Loss event probability is the number of loss events divided by the number of packets sent.

Intra-protocol requirements: The requirements that the protocol should satisfy in a network consisting of a single protocol are the following:

- *Efficiency.* The average throughput for the new protocol should be larger than that of standard TCP in high-speed networks.
- *Intra-protocol fairness.* Network resources should be fairly allocated to all flows. Fairness here does not necessarily mean that all flows sharing the same link achieve the same throughput. Instead, this means that the new protocol should not be significantly more unfair than the current TCP. For example, under the current TCP, flows with different RTTs achieve similar average window sizes, and their average throughput are inversely proportional to their RTTs. The new protocol should not be significantly more biased against long RTT flows.
- *Responsiveness.* The congestion control algorithm should reach the fair operating point quickly, starting from any initial condition.
- *Heavy congestion*[3] *avoidance.* A simple idea to achieve a larger average window size for a given loss event probability is to choose a large value for α (window increment parameter) and a small value for β (window reduction parameter). However, rapid increase and small decrease in window size may cause large number of packets to be dropped during a congestion event, which we call heavy congestion, and thus may lead to some undesirable consequences. First, heavy congestion causes more timeouts and makes TCP enter the slow start phase more often, and causes under-utilization. For example, HS-TCP faces timeouts regularly if SACK is not used. Second, heavy congestion causes synchronization more often, which makes the resource allocation very unfair for large RTT users, as will be discussed later.
- *Router independence.* The new protocol should work well regardless of router characteristics, like the buffer size at the router, and the queue management algorithm of the router (Droptail or some Active Queue Management (AQM) schemes). With a more advanced router, like with a larger buffer or an AQM support, the new protocol might achieve better performance, but the performance with Droptail and small buffer should also be good.
- *Robustness.* The new protocol should be robust against the noise in congestion signal measurements, especially if this new protocol uses queueing delay as the congestion signal, since queueing delay measurements are typically noisy.

Inter-protocol requirements: The requirements on the protocol when it coexists in a network with standard TCP are the following:

- *Compatibility.* In low-speed networks, the new protocol should achieve a similar rate as that of standard TCP; and in high-speed networks, standard TCP should not suffer significant throughput loss when it coexists with the new protocol.
- *Incentive to switch.* By switching to the new protocol from standard TCP, the users should achieve a higher average throughput in a network that accommodates both protocols.

We now briefly discuss existing TCP variants to see whether they satisfy all these requirements. First, it is impractical to modify routers if the benefit is marginal or can be achieved by sender-side modifications, and thus algorithms which need router-side modifications, like XCP, are not ideal. Without modifying the router, a sender has only two congestion signals: packet loss and queueing delay. We can thus classify the prior sender-side protocols into one of two classes. Loss-based congestion control algorithms, like HS-TCP and Scalable TCP, use packet loss as primary congestion signal, increase window size for each ACK and decrease window size for packet loss. Loss-based algorithms can be regarded as generalizations of TCP's AIMD algorithm, and we call them general AIMD algorithms, since the only difference from AIMD is that they set different α and β values and allow them to be variables. On the other hand, delay-based congestion control algorithms, like TCP-Vegas and FAST TCP, are fundamentally different from AIMD, as they use queueing delay as the primary congestion signal, increase window size if delay is small and decrease window size if delay is large.

The advantage of delay-based algorithms is that they achieve better average throughput, since they can keep the system around full utilization. As a comparison, the loss-based algorithms purposely generate packet losses and

[3] In our context, heavy congestion means that many packets are dropped when congestion happens. It only concerns the time when congestion happens and it does not necessarily mean that the packet loss probability or the loss event probability is high. In some other papers, it is called heavy synchronization.

oscillate between full utilization and under-utilization. However, existing delay-based algorithms suffer from some inherent weaknesses. First, they are not compatible with standard TCP. TCP-Vegas gets a very small share of the link capacity if competing with TCP-Reno [1,24]; and FAST TCP yields a non-unique equilibrium point if competing with TCP-Reno: the allocation of the bandwidth between FAST and Reno users depend on which users enter the network first [32]. Second, they require the buffer size at the router to be larger than a specified value and this value increases with the number of users N. Both Vegas and FAST control the number of packets queued in the router for each flow, and this number cannot be too small. The requirement for the router buffer is thus N times this number. For a fixed buffer size, there is an upper bound on N for Vegas or FAST to work efficiently. Finally, the performance of these delay-based algorithms deteriorates if the delay measurements are noisy [7,21,26]. On the other hand, none of the existing loss-based algorithms satisfy all the requirements either. Scalable TCP sets α proportional to W, but it has been demonstrated to be unfair (see [18], Fig. 2). HS-TCP sets α to be a step-wise increasing function of W, and β a step-wise decreasing function of W, but its convergence speed is very slow (see [18], Fig. 1). H-TCP aims at a faster convergence and better utilization by setting α to be an increasing function of the time elapsed since last backoff and setting β to be such that the link is always around full utilization, even after the backoff. For all the above algorithms, the increase is initially slow, when the window size is small and the network is far from congestion, but becomes fast later, when the window size is large and the network is close to congestion. As a result, the window size curve between two consecutive loss events is *convex*.

This convex nature is not desirable. First, the slow increment in window size when the network is far from congestion is inefficient. For a given β, the convex window curve gets an even smaller average throughput than traditional linear increase, and thus these algorithms have to choose a smaller $\beta < 1/2$, which is not friendly to standard TCP. Second, the fast increment in window size when the network is close to congestion causes heavy congestion more easily. As we have mentioned before and will further discuss later, heavy congestion causes more frequent timeouts, more synchronized window backoffs, and is more unfair to large RTT users. In summary, the main problem with existing general AIMD algorithms is the convexity of the W curve. An ideal window curve should be *concave*, which is more efficient and avoids heavy congestion. It is proved in [23] that the optimal congestion control should have a concave window curve.

BIC-TCP and CUBIC-TCP were proposed recently to yield a non-convex window curve. They achieve a concave (linear to logarithmic) window curve at the initial stage by using a binary search method to lead the window size to a reference point, which is the old "maximum" window size (the window size right before the last congestion indication). However, after the reference point is reached, these two algorithms need to choose a max probing method to detect the new maximum, and this yields a convex (linear to exponential) window curve. So the window curves for BIC-TCP and CUBIC-TCP are first concave then convex. Although these two algorithms have better performance than convex curve algorithms, they are still not optimal [23]. An objective of our work is to design a general AIMD algorithm which results in a concave window curve throughout the whole congestion epoch (the time interval between two consecutive congestion events). We will compare our concave window algorithm with the non-concave curve algorithms (ex. HS-TCP and BIC-TCP) by *ns-2* simulations in Section 6.

3. The TCP-Illinois protocol

To achieve the concave window curve, we should set α large when far from congestion and set it small when close to congestion. To achieve a better throughput in networks with packet losses not due to congestion and to be fair with standard TCP, we should also set β small when far from congestion and set it large when close to congestion. The difficulty is in judging whether the congestion is imminent or not, since it requires an estimation of the current congestion level. Before congestion (packet loss) really happens, the only congestion indicating information is queueing delay. So our key idea is the following: when the average queueing delay d_a is small, the sender assumes that the congestion is not imminent and sets a large α and small β; when d_a is large, the sender assumes that the congestion is imminent and sets a small α and large β. As a result, $\alpha = f_1(d_a)$ and $\beta = f_2(d_a)$, where $f_1(\cdot)$ is decreasing and $f_2(\cdot)$ is increasing. Any combination of increasing $f_1(\cdot)$ and decreasing $f_2(\cdot)$ functions results in a concave window curve and therefore, we call such algorithms Concave-AIMD or C-AIMD algorithms.

Note that C-AIMD algorithms use loss to determine the *direction* and use delay to adjust the *pace* of window size change. So loss is the primary congestion signal and delay is the secondary congestion signal. This makes C-AIMD fundamentally different from another recently proposed algorithm, called Compound-TCP (C-TCP) [31], which uses

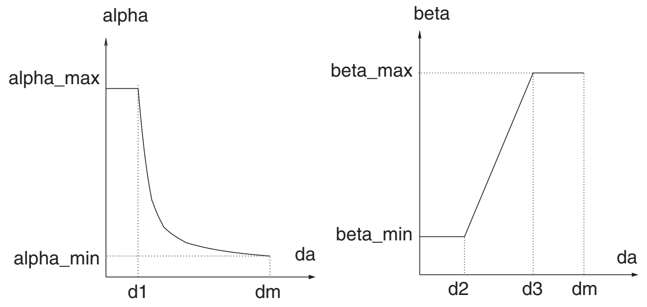

Fig. 1. α and β curves vs d_a.

loss and delay information as primary signals (determining *direction* of window size change) in different stages. We will also compare the performance of C-AIMD and C-TCP by *ns-2* simulations in Section 6. Aside from the difference in performance demonstrated in Section 6, C-TCP differs from C-AIMD in the following senses: C-TCP does not achieve the concave window curve (and thus not optimal), needs a sufficiently large buffer, and uses delay to determine the *direction* of window change. As we have mentioned, one problem in using delay to control congestion is that delay cannot be measured accurately since usually the RTT measurements are noisy. If delay determines the *direction* of window size change, noisy RTT measurements could degrade the performance significantly. Our C-AIMD algorithms which use delay only as a *secondary* signal, are much more robust to noise in RTT measurements, as discussed in Section 6.4. There are numerous choices for $f_1(\cdot)$ and $f_2(\cdot)$. TCP-Illinois is a special case of C-AIMD algorithms which use the following choices for $f_1(\cdot)$ and $f_2(\cdot)$:

$$\alpha = f_1(d_a) = \begin{cases} \alpha_{\max} & \text{if } d_a \le d_1 \\ \dfrac{\kappa_1}{\kappa_2 + d_a} & \text{otherwise.} \end{cases} \tag{2}$$

$$\beta = f_2(d_a) = \begin{cases} \beta_{\min} & \text{if } d_a \le d_2 \\ \kappa_3 + \kappa_4 d_a & \text{if } d_2 < d_a < d_3 \\ \beta_{\max} & \text{otherwise.} \end{cases} \tag{3}$$

We let $f_1(\cdot)$ and $f_2(\cdot)$ be continuous functions and thus $\frac{\kappa_1}{\kappa_2+d_1} = \alpha_{\max}$, $\beta_{\min} = \kappa_3 + \kappa_4 d_2$ and $\beta_{\max} = \kappa_3 + \kappa_4 d_3$. Suppose d_m is the maximum average queueing delay and let $\alpha_{\min} = f_1(d_m)$; then we also have $\frac{\kappa_1}{\kappa_2+d_m} = \alpha_{\min}$. From these conditions, we have

$$\begin{aligned} \kappa_1 &= \frac{(d_m - d_1)\alpha_{\min}\alpha_{\max}}{\alpha_{\max} - \alpha_{\min}} \quad \text{and} \quad \kappa_2 = \frac{(d_m - d_1)\alpha_{\min}}{\alpha_{\max} - \alpha_{\min}} - d_1, \\ \kappa_3 &= \frac{\beta_{\min} d_3 - \beta_{\max} d_2}{d_3 - d_2} \quad \text{and} \quad \kappa_4 = \frac{\beta_{\max} - \beta_{\min}}{d_3 - d_2}. \end{aligned} \tag{4}$$

This specific choice is shown in Fig. 1.

We now describe the TCP-Illinois protocol in more detail:

- All the features of TCP-NewReno except the AIMD algorithm are retained.
- In the congestion avoidance phase, the sender measures RTT T for each acknowledgement, and averages the RTT measurements over the last W acknowledgements (one RTT interval) to derive the average RTT T_a. The sender records the maximum and minimum (average) RTT[4] ever seen, as $T_{\max}$ and $T_{\min}$, respectively, and computes the maximum (average) queueing delay $d_m = T_{\max} - T_{\min}$ and the current average queueing delay $d_a = T_a - T_{\min}$.
- The sender picks the following parameters: $0 < \alpha_{\min} \le 1 \le \alpha_{\max}$, $0 < \beta_{\min} \le \beta_{\max} \le 1/2$, $W_{\text{thresh}} > 0$, $0 \le \eta_1 < 1$, $0 \le \eta_2 \le \eta_3 \le 1$. The sender sets $d_i = \eta_i d_m$ $(i = 1, 2, 3)$, computes κ_i $(i = 1, 2, 3, 4)$ from (4),

[4] The are two options here. In the default option, the maximum and minimum average RTTs are recorded. In an alternative option, the maximum and minimum instantaneous RTTs are recorded. These two options yield almost identical results, unless the delay signal is buried with noise and the noise is in a high level. Under this case, the default option is a better choice.

and computes α and β values from (2) and (3), respectively. The standard settings of these parameters are given in Section 6.
- $\alpha \leftarrow 1$ and $\beta \leftarrow 1/2$ if $W < W_{\text{thresh}}$.
- The κ_i ($i = 1, 2, 3, 4$) values are updated if $T_{\max}$ or $T_{\min}$ is updated. The α and β values are updated once per RTT.
- $W \leftarrow W + \alpha/W$ for each ACK.
- $W \leftarrow W - \beta W$, if in the last RTT there is packet loss detected through triple duplicate ACK.
- Once there is a timeout, the sender sets the slow start threshold to be $W/2$, enters slow start phase, and resets $\alpha = 1$ and $\beta = 1/2$, and α and β values are unchanged until one RTT after the slow start phase ends.

TCP-Illinois retains the fast recovery and fast retransmission features of NewReno in standard option. If the receivers support selective acknowledgement, TCP-Illinois can also back off its window size when packet loss is detected through selective ACK and adopt features from SACK TCP. However, the SACK support is not needed, since TCP-Illinois avoids heavy congestion effectively.

In addition to the above major features of the protocol, TCP-Illinois also contains another feature to improve the robustness against sudden fluctuations in delay measurements that can result from measurement noise, bursty packet arrival process, etc. To understand this feature, note that ideally once d_a becomes greater than d_1, it should stay above d_1 until some users reduce their window sizes. However, due to bursty packet arrival process or measurement noise, it is possible for d_a to drop rather suddenly below d_1 before some users reduce their window sizes. In this case, we should not set $\alpha = \alpha_{\max}$ unless we are really sure that the network is not in a congested state. Therefore, once $d_a > d_1$, we do not allow α to increase to $\alpha_{\max}$ unless d_a stays below d_1 for a certain amount of time. TCP-Illinois chooses another parameter θ, and lets θ times RTT be this amount of time. The standard setting for θ is again given in Section 6.

We note that the adaptation of α is the key feature of TCP-Illinois, whereas the adaptation of β as a function of average queueing delay is only relevant in networks where there are non-congestion-related losses, such as wireless networks or extremely high-speed networks. In wireless networks, some packet losses arise from channel fluctuations. In extremely high-speed networks, congestion loss probability is so small that it is at the same level as or even smaller than the probability of packet transmission error at the link, and as a consequence, a non-trivial proportion of packet losses are from transmission error. For these non-congestion-related packet losses, we wish to avoid a sharp window size reduction. Then, the β adaptation of TCP-Illinois shows its advantage: although it still reduces window size, the reduction percentage is very small, since the queueing delay is very small.

4. Fairness and stability

In this section, we study the fairness and stability of TCP-Illinois. This involves both the intra-protocol fairness between different TCP-Illinois users and also inter-protocol fairness with standard TCP, i.e., the resource allocation between TCP-Illinois users and standard TCP users. We first develop a new stochastic matrix model for a class of general AIMD algorithms, which include standard TCP and TCP-Illinois as special cases, and then study the fairness and stability properties of these algorithms using this new model.

4.1. Stochastic matrix model for general AIMD algorithms

There have been several recent papers on the stochastic matrix model of AIMD algorithms; see [2–5,17,29,34]. We first provide an overview of this model, and then extend this model by modifying one of the assumptions in the earlier work. Throughout, we consider networks with a single bottleneck link which uses Droptail, analyze the congestion avoidance phase only, and assume that all packet losses are caused by congestion.

Suppose a link with capacity C and queue limit B is shared by N users, indexed by i ($i = 1, 2, \ldots, N$). User i has a transmission rate (or throughput) x_i, a window size W_i, a window increment parameter α_i, a window backoff factor β_i, and RTT T_i. We define $\mathbf{W} := [W_1, \ldots, W_N]^{\mathrm{T}}$, and $\mathbf{x} := [x_1, \ldots, x_N]^{\mathrm{T}}$. When the link is congested and one or more packets are dropped, we call this a congestion event, and denote by t_k the time at the kth congestion event. At a congestion event, one or more flows see packet losses and backoff their window sizes, and we say that a loss event[5]

[5] In our terminology, a congestion event is for a link, while a loss event is for an individual user.

happens for these flows. For any variable v, we use $v(t)$ to denote its value at time t, use $v[k]$ (respectively, $v[k^+]$) to denote its value just before (respectively, after) the kth congestion event, use $E[v]$ to denote the expected value of $v[k]$, and use $\bar{v}$ to denote the average of all $v[k]$'s. Here, v could stand for W_i, x_i, α_i, β_i, T_i, $\mathbf{W}$, $\mathbf{x}$, as well as some other variables to be introduced later.

We now consider the congestion event k for Droptail queue. When congestion happens, the buffer is full, so every user experiences a maximum queueing delay $d_m = B/C$, and thus $T_i[k] \equiv \hat{T}_i := T_i^p + d_m, \forall k$, where T_i^p is the propagation delay of user i. At the congestion event, the instantaneous throughput for user i is $W_i[k]/\hat{T}_i$. Ignoring the burstiness of packet arrival process, we can assume that the outgoing packets from one particular user are evenly distributed along the path. For user i, altogether there are $W_i[k]$ packets, and thus the number of packets from user i queued in the link buffer should be $W_i[k]d_m/\hat{T}_i = x_i[k]d_m$. The sum of the queued packets from all users should be the link buffer limit B, and thus we have

$$B = \sum_{i=1}^{N} x_i[k]d_m = \sum_{i=1}^{N} x_i[k]\frac{B}{C}, \tag{5}$$

which leads to following equation:

$$\sum_{i=1}^{N} x_i[k] = C, \quad \forall k \in \{0, 1, 2, \ldots\}. \tag{6}$$

As mentioned earlier, in this analysis we have ignored the burstiness of packet arrival process; if we had considered this burstiness, then $\sum_{i=1}^{N} x_i[k]$ would not be a constant, and would be either greater than or less than C. We now define $\Sigma = \{\mathbf{z} = [z_1, \ldots, z_N]^{\mathrm{T}} \in \mathbb{R}^N : z_i \geq 0, \sum_{i=1}^{N} z_i = C\}$; then Σ is the set of all possible $\mathbf{x}[k]$'s, and we call Σ the feasible set of $\mathbf{x}[k]$.

Between two consecutive congestion events, $W_i(t)$ is increased at rate $\alpha_i(t)/T_i(t)$, and thus

$$W_i[k+1] = W_i[k^+] + \int_{t_k}^{t_{k+1}} \frac{\alpha_i(t)}{T_i(t)}\mathrm{d}t. \tag{7}$$

If we define

$$\tilde{T}_i[k] := \frac{\int_{t_k}^{t_{k+1}} \alpha(t)\mathrm{d}t}{\int_{t_k}^{t_{k+1}} \frac{\alpha(t)}{T_i(t)}\mathrm{d}t}, \tag{8}$$

then, we have

$$W_i[k+1] = W_i[k^+] + \frac{1}{\tilde{T}_i[k]} \int_{t_k}^{t_{k+1}} \alpha_i(t)\mathrm{d}t. \tag{9}$$

For any user i and congestion event k, $\tilde{T}_i[k] \in [T_i^p, \hat{T}_i]$. If queueing delay is much smaller than propagation delay, then $\tilde{T}_i[k]$ varies in a very small range. For analytical convenience, we assume that $\tilde{T}_i[k] \equiv \tilde{T}_i, \forall k$.

At each congestion event k and for each flow i, we define the loss event random variable $D_i[k]$:

$$D_i[k] := \begin{cases} 1 & \text{if flow } i \text{ sees at least one packet loss,} \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

and define $\mathbf{D}[k] := [D_1[k], \ldots, D_N[k]]^{\mathrm{T}}$. Note that $D_i[k]$ and $D_j[k]$ are correlated, since $\sum_{i=1}^{N} D_i[k] \geq 1$.

With the loss event random variables defined, we have

$$W_i[k^+] = W_i[k](1 - \beta_i[k]D_i[k]). \tag{11}$$

Combining (9) and (11), and using the fact that $x_i[k] = W_i[k]/\hat{T}_i$, we have

$$x_i[k+1] = x_i[k](1 - \beta_i[k]D_i[k]) + \frac{1}{\hat{T}_i\tilde{T}_i} \int_{t_k}^{t_{k+1}} \alpha_i(t)\mathrm{d}t. \tag{12}$$

Eqs. (6) and (12) describe the discrete-time stochastic model of all general AIMD algorithms.

4.2. Markov chain for identical $\alpha_i(t)$ and constant $\beta_i[k]$

We consider the class of AIMD algorithms which have the following properties: (i) $\alpha_i(t) = \alpha(t), \forall i$, where $\alpha(t)$ is the common window increment for all users at time t; (ii) $\beta_i[k] \equiv \hat{\beta}, \forall i, k$, where $\hat{\beta}$ is a constant independent of i and k. This class includes standard TCP obviously, since it satisfies $\alpha_i(t) \equiv 1, \beta_i[k] \equiv \hat{\beta} = 1/2, \forall i, k$. This class also includes TCP-Illinois. First, $\alpha_i(t) = \alpha(t)$, since the queueing delay is the same for all users. Here, we ignore the differences in α among different flows due to feedback delays, since the queueing delays are averaged to compute α. Then, $\beta_i[k] \equiv \hat{\beta} = \beta_{\max}, \forall i, k$, since the average queueing delay d_a is larger than the threshold parameter d_3 when congestion happens, if the parameters are carefully chosen. Recall our modeling of congestion events in the previous subsection: we know that the maximum queueing delay d_m is reached at each congestion event. Even considering the averaging process, $d_a[k]$ is close to d_m and still larger than d_3.

For this class of AIMD algorithms, from (6), we have

$$\int_{t_k}^{t_{k+1}} \alpha(t)\mathrm{d}t \sum_{i=1}^{N} (\tilde{T}_i \hat{T}_i)^{-1} = \sum_{i=1}^{N} \hat{\beta} D_i[k] x_i[k], \tag{13}$$

and thus

$$\int_{t_k}^{t_{k+1}} \alpha(t)\mathrm{d}t = \frac{1}{\sum_{i=1}^{N} (\hat{T}_i \tilde{T}_i)^{-1}} \sum_{i=1}^{N} \hat{\beta} D_i[k] x_i[k]. \tag{14}$$

Define

$$\gamma_i := (\tilde{T}_i \hat{T}_i)^{-1} \Big/ \sum_{j=1}^{N} (\tilde{T}_i \hat{T}_i)^{-1}, \tag{15}$$

and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_N]^{\mathrm{T}}$. Then $\sum_{i=1}^{N} \gamma_i = 1$. We now have

$$x_i[k+1] = x_i[k](1 - \hat{\beta} D_i[k]) + \gamma_i \sum_{j=1}^{N} x_i[k] \hat{\beta} D_i[k]. \tag{16}$$

In vector form, we have

$$\mathbf{x}[k+1] = A[k]\mathbf{x}[k], \tag{17}$$

where

$$\begin{aligned} A[k] &= A(\mathbf{D}[k]) \\ &= \mathrm{diag}(1 - \hat{\beta} D_1[k], \ldots, 1 - \hat{\beta} D_N[k]) + \boldsymbol{\gamma} \hat{\beta} (D_1[k], \ldots, D_N[k]). \end{aligned} \tag{18}$$

We see that $\mathbf{x}[k]$ forms a discrete-time Markov Chain on the continuous state space Σ. For any k, $A[k]$ is a non-negative, random, column stochastic matrix [6,11]. The property of this Markov Chain is determined by the A matrix, and thus determined by $\mathbf{D}[k]$.

Note that, although $\alpha_i(t)$ determines the window curve, the recovery time after a congestion event, and the utilization of the bandwidth, once all users see the same $\alpha(t)$ at any time, $\alpha(t)$ does not influence the discrete-time Markov Chain at congestion events, and thus the exact form of $\alpha(t)$ is not important to understand the macroscopic fairness properties of this class of algorithms. So this stochastic matrix model applies to the entire class of such algorithms, and the special choice of $\alpha(t)$ in TCP-Illinois is not important when analyzing the fairness of TCP-Illinois. This special choice of $\alpha(t)$ indeed influences many other properties, such as efficiency and synchronization, as we will discuss later.

4.3. Stability and fairness: General case

In this subsection, we study the stability and fairness properties of the Markov Chain defined in (17) and (18). Let $\mathbb{S}$ be the set of all non-empty subsets of $\{1, 2, \dots, N\}$, and suppose $s[k] \in \mathbb{S}$ is the set of users that experience a loss event at congestion event k. Define $\rho_s[k] := \text{Prob}(s[k] = s)$, where $s \in \mathbb{S}$. Then, the distribution of $D[k]$ is determined by the values of $\rho_s[k], \forall s \in \mathbb{S}$. Let $q_i[k] := \text{Prob}(D_i[k] = 1)$, then $q_i[k] = \sum_{s:i\in s} \rho_s[k]$, and $q_i[k]$ denotes the window backoff probability of user i at congestion event k.

Most prior work assumed that $D_i[k]$ is independent of $\mathbf{x}[k]$, i.e., $\rho_s[k]$ is a constant independent of k, for each $s \in \mathbb{S}$, and $s[k]$ is identically independently distributed (i.i.d.). From this assumption, $q_i[k]$ is also a constant independent of k and $\mathbf{x}[k]$ for all users i. However, in reality, at different congestion events, a flow is more likely to see a loss event when it has a larger throughput. Therefore, we modify the stochastic matrix model by allowing $\mathbf{D}[k]$ to be dependent on $\mathbf{x}[k]$, and allowing $q_i[k]$ and $\rho_s[k]$ to be functions of $\mathbf{x}[k]$ as well:

$$\rho_s[k] = \rho_s(\mathbf{x}[k]), \quad \forall l, k, \quad \text{and} \quad q_i[k] = q_i(\mathbf{x}[k]), \quad \forall i, k. \tag{19}$$

We make the following assumption on ρ_s, q_i, and $\mathbf{D}$:

Assumption 1. (i) $\rho_s(\cdot)$ and $q_i(\cdot)$ are continuous functions in $\mathbf{x}[k]$. (ii) For any realization of the infinite length Markov Chain defined in (17) and (18) and for any user i, $D_i[k] = 1$ for infinitely many k's almost surely, i.e., for any $J > 0$,

$$\text{Prob}(D_i[k] = 0, \forall k \geq J) = 0, \quad \forall i \in \{1, 2, \dots, N\}.$$

We now state the following theorem.

Theorem 4.1. *Under Assumption 1, the Markov Chain defined in (17) and (18) has a unique invariant distribution, and starting from any initial state, the distribution of* $\mathbf{x}[k]$ *converges to this invariant distribution. Moreover, the Markov Chain is ergodic, i.e., for any continuous function* $h(\cdot) : \Sigma \to R$, $h(\overline{\mathbf{x}[k]})$, *the time average of* $h(\mathbf{x}[k])$, *equals* $E[h(\mathbf{x}[k])]$, *the expected value of* $h(\mathbf{x}[k])$ *under the invariant distribution.*

Proof. See [19], a longer version of this paper. □

With the existence and uniqueness of the invariance distribution established, we now study the fairness among different users, i.e., the resource allocation under the invariant distribution. We have the following theorems on fairness.

Theorem 4.2. *If* N *users sharing one link have homogeneous RTTs, under the unique invariant distribution of the Markov Chain defined in (17) and (18), all flows share the same expected throughput* $E[x_i[k]]$.

Proof. When all users have the same RTT, $\hat{T}_i$, $\tilde{T}_i$ and γ_i are the same for all users. Then, from (18), the A matrix does not depend on i or T_i. If we swap user i and user j, the Markov Chain is the same as when we do not swap user i and user j, but swap $x_i[0]$ and $x_j[0]$. From Theorem 4.1, we know that the invariant distribution is unique, independent of the initial condition. Therefore, the invariant distribution is unchanged if we swap user i and user j, and thus $E[x_i[k]] = E[x_j[k]], \forall i, j \in \{1, 2, \dots, N\}$. □

Theorem 4.3. *If* N *users sharing one link have heterogeneous RTTs, under the unique invariant distribution of the Markov Chain defined in (17) and (18), the following equation holds:*

$$\hat{T}_i \tilde{T}_i E[x_i[k] q_i(\mathbf{x}[k])] = E[W_i[k] \tilde{T}_i q_i(\mathbf{x}[k])] = C_1, \quad \forall i, \tag{20}$$

where C_1 *is a constant independent of* i.

Proof. Taking expectation of $x_i[k+1]$ given $\mathbf{x}[k]$ in (16), we have

$$E[x_i[k+1] | \mathbf{x}[k]] = x_i[k] - \hat{\beta} q_i(\mathbf{x}[k]) x_i[k] + \gamma_i \sum_{j=1}^{N} \hat{\beta} q_i(\mathbf{x}[k]) x_j[k]. \tag{21}$$

Under the invariant distribution,

$$E[x_i[k]] = E[x_i[k+1]] = E[E[x_i[k+1]|\mathbf{x}[k]]]$$
$$= E[x_i[k]] - \hat{\beta}E[q_i(\mathbf{x}[k])x_i[k]] + \gamma_i\hat{\beta}\sum_{j=1}^{N} E[q_j(\mathbf{x}[k])x_j[k]]. \tag{22}$$

So we have that $E[x_i[k]q_i(\mathbf{x}[k])]/\gamma_i = \hat{T}_i\tilde{T}_i E[x_i[k]q_i(\mathbf{x}[k])] = E[W_i[k]\tilde{T}_i q_i(\mathbf{x}[k])]$ is independent of i and we have proved (20). □

4.4. Models for $\rho_s(\cdot)$ and $q_i(\cdot)$

From Theorem 4.3, we see that the resource allocation depends on the form of $q_i(\cdot)$. If $q_i(\cdot)$ is constant, it is exactly the same as the prior results in [34]. In our model, $q_i[k]$ is allowed to be a function of $\mathbf{x}[k]$. We need to specify the $q_i(\cdot)$ function to further analyze the fairness property. Recall that the dependence of $q_i[k]$ on $\mathbf{x}[k]$ arises from the fact that a flow with a larger throughput is more likely to see a loss event than a flow with a smaller throughput. Accordingly, we make the following assumption:

Assumption 2. At each congestion event, the total number of packets dropped is a random variable that takes values in $\{1, 2, \dots, M_{\max}\}$, and its distribution is independent of k. Furthermore, for any packet dropped at congestion event k, the probability that it belongs to flow i is $x_i[k]/C$.

This assumption is justified by the following reasoning: since the total arrival rate is independent of k, so is the distribution for the total number of packets dropped; since at least one packet is dropped and only a finite number of packets are dropped, there are lower and upper bounds for the total number of packets dropped; since the probability of an arbitrary packet belonging to flow i is $x_i[k]/C$, so is the probability of a dropped packet belonging to flow i.

Lemma 4.1. *Assumption 1 holds given Assumption 2*

Proof. We first prove that Assumption 1 (i) holds. Let M be the random variable indicating the total number of packets dropped in one congestion event, let $P_M(m) = \text{Prob}(M = m)$ for all $m \in \{1, 2, \dots, \bar{M}\}$, and let $\hat{M} = E[M]$. Then, we have

$$q_i = 1 - \text{Prob(no dropped packets from flow } i)$$
$$= \sum_{m=1}^{M_{\max}} P_M(m)\left[1 - \left(1 - \frac{x_i[k]}{C}\right)^m\right] = f(x_i[k]), \tag{23}$$

where $f(x) := \sum_{m=1}^{M_{\max}} P_M(m) f_m(x)$, and $f_m(x) := 1 - (1 - \frac{x}{C})^m$. Both $f_m(x), \forall m$ and $f(x)$ are strictly increasing continuous functions in $x \in [0, C]$. Note that Assumption 1 allows $q_i[k]$ to be functions of all users' rates, while Assumption 2 further tells us that $q_i[k]$ is only a function of its own rate $x_i[k]$, and this relationship $f(\cdot)$ is common for all users.

We then study $\rho_s(\cdot)$. For a specific $s \in \mathbb{S}$, suppose $s = \{i_1, i_2, \dots, i_H\}$, where $1 \le i_1 < i_2 < \cdots < i_H \le N$. Then, $\text{Prob}(s[k] = s|M = m) = 0$ if $m < H$. If $m \ge H$, we have

$$\rho_{s,m}(\mathbf{x}[k]) := \text{Prob}(s[k] = s|M = m)$$
$$= \sum_{m_1,\dots,m_H} \left(\frac{x_{i_1}[k]}{C}\right)^{m_1} \cdots \left(\frac{x_{i_H}[k]}{C}\right)^{m_H} \binom{m}{m_1, m_2, \dots, m_H},$$

where the summation is over all $m_h \ge 1, \forall h \in \{1, 2, \dots, H\}$, and $\sum_{h=1}^{H} m_h = m$. And we have

$$\rho_s(\mathbf{x}[k]) = \text{Prob}(s[k] = s) = \sum_{m=H}^{\infty} P_M(m)\rho_{s,m}(\mathbf{x}[k]). \tag{24}$$

So both $q_i(\mathbf{x}[k])$ and $\rho_s(\mathbf{x}[k])$ are continuous functions of $\mathbf{x}[k]$.

Next we prove Assumption 1(ii). At each congestion event, at least one user will decrease its window size by at least 1, and thus $\sum_{i=1}^{N} x_i[k]\hat{\beta} D_i[k] \geq 1/T_m$, where $T_m = \max_i \hat{T}_i$. Hence, if user i does not back off at congestion event k, $x_i[k+1] \geq \gamma_i/T_m$. Since at least one packet is lost at congestion event $k+1$ and the probability of a lost packet belonging to flow i is $x_i[k+1]/C$, we know that $q_i[k+1] \geq x_i[k+1]/C$. Thus $q_i[k+1] \geq \epsilon := \gamma_i/(CT_m) > 0$. So the probability that user i backs off at least once in any two consecutive congestion events is lower bounded by $\epsilon > 0$. As a consequence, for any user $i \in \{1, 2, \ldots, N\}$, $D_i[k] = 1$ for infinitely many k's almost surely. □

Since Assumption 2 implies Assumption 1, we know that Theorems 4.1–4.3 hold under Assumption 2 also. In the next subsection, we analyze the fairness property for the specific $f(x_i)$ function given by (23).

4.5. Synchronization and fairness

From Theorem 4.3 and Eq. (23), the $f(\cdot)$ function uniquely determines the backoff behavior and the fairness property. Different $f(\cdot)$ functions lead to different backoff behaviors: for example, if $f(\cdot) \equiv 1$, the backoffs are completely synchronized; otherwise, they are not. The exact form of $f(\cdot)$ depends on the distribution of M, and is thus unknown if $P_M(\cdot)$ is unknown. However, we can bound $f(x)$ in general and approximate $f(x)$ for some special cases. Since

$$1 - \frac{x}{C} \geq \left(1 - \frac{x}{C}\right)^m \geq 1 - \frac{mx}{C}, \quad \forall\, 0 \geq x \geq C,$$

we have

$$\frac{x}{C} \leq f(x) \leq \sum_{m=1}^{M_{\max}} P_M(m) \left[1 - \left(1 - \frac{mx_i[k]}{C}\right)\right] = \frac{\hat{M}x}{C}. \tag{25}$$

Note that the loss event for a flow is the union of the events that each dropped packet belongs to this flow, therefore the bound on $f(x)$ in (25) is just the union bound. Since $x \geq 0$ and $f(x) \geq 0$ always, we have

$$\frac{x^2}{C} \leq E[xf(x)] \leq \frac{\hat{M}x^2}{C}.$$

From Theorem 4.3, we have

$$\frac{\tilde{T}_j}{\hat{M}\hat{T}_j} \leq \frac{\bar{W_i^2}}{\bar{W_j^2}} \leq \frac{\hat{M}\hat{T}_i}{\tilde{T}_i}, \quad \forall i \neq j. \tag{26}$$

The bounds are tight and $\bar{W_i^2} \approx \bar{W_j^2}$ if $\hat{M}$ is close to 1 (very light congestion). If the variance of $W_i[k]$ is much smaller than $(E[W_i[k]])^2$, then $\bar{W_i^2} \approx (\bar{W}_i)^2$, and thus the average window sizes of all flows are almost the same under the very light congestion case. From simulations which will be presented later, we observe that $\mathrm{Var}(W_i[k]) \ll (E[W_i[k]])^2$ indeed. If $\hat{M}$ is very large (heavy congestion), the bounds are meaningless and $\bar{W_i^2}$ and $\bar{W_j^2}$ can be significantly different.

We then consider the approximation of $f(x)$ under some special cases. When x is small, such that $mx/C \ll 1$, the probability that more than one dropped packet belong to one flow (with rate x) is very small, and thus the union upper bound is nearly reached: $f_m(x) \approx mx/C$. If $M_{\max}x/C \ll 1$, then $f(x) \approx \hat{M}x/C$, and $f(x) \propto x$. When x is large, such that $mx/c \gg 1$, then $f_m(x) \approx 1$. If $M \gg c/x$ with very high probability, then $f(x) \approx 1$. In general, $f(x)$ is a concave curve, and $f(x) \propto x^\lambda$, where $0 \leq \lambda \leq 1$, and $\lambda \to 1$ as $M_{\max} \to 1$, and $\lambda \to 0$ as $\hat{M} \to \infty$.

As we have mentioned, simulations show that the standard deviation of $x_i[k]$ is very small compared with $E[x_i[k]]$, so with high probability, $x_i[k]$ lies not far away from $E[x_i[k]]$. If the RTTs of different users do not differ significantly, $E[x_i[k]]$ is not significantly different from C/N, and the probability of $x_i[k] \ll C/N$ or $x_i[k] \gg C/N$ is very small. So if $M_{\max} \ll N$, which we call "light congestion", almost always $Mx_i[k]/c \ll 1$, and thus $f(x_i[k]) \propto x_i[k]$ ($\lambda \approx 1$), and the window backoffs of different users are completely unsynchronized. If $\hat{M} \gg N$, which we call "heavy congestion", almost always $Mx/c \gg 1$, and thus $f(x) \propto 1$ ($\lambda \approx 0$), and the window backoffs are completely synchronized. In the middle of these two extreme cases, $0 < \lambda < 1$, and the window backoffs are partially unsynchronized.

Plugging $f(x) \propto x^{\lambda}$ into (20), we get that $\hat{T}_i \tilde{T}_i E[(x_i[k])^{\lambda}]$ is the same for all users. If the variance of $x_i[k]$ is small, and if the difference between $\tilde{T}_i$ and $\hat{T}_i$ is small also, we get the following fairness property:

$$\bar{x_i} \propto \frac{1}{\hat{T}_i^{1+\mu}}, \quad \text{and} \quad \bar{W_i} \propto \frac{1}{\hat{T}_i^{\mu}}, \tag{27}$$

where $\mu = (1-\lambda)/(1+\lambda)$. We know that $0 \le \mu \le 1$ in general; $\mu \approx 0$ for light congestion and completely unsynchronized window backoff; and $\mu \approx 1$ for heavy congestion and completely synchronized window backoff. So when the variance of $x_i[k]$ is small and when the RTTs are not significantly different, the fairness depends on the synchronization, which further depends on the heaviness of congestion. Light congestion leads to unsynchronized window backoff and equality of window sizes ($\mu = 0$); heavy congestion leads to synchronized window backoff and an inverse proportional relationship between window size and RTT ($\mu = 1$); and the general case ($0 < \mu < 1$) lies in the middle of these two extreme cases. We then explore the factors that influence the distribution of M. Consider the homogeneous RTT case and suppose the system is slotted with each slot being one RTT. Since the pipe can hold at most $CT + B$ packets and W_i increases by α_i within each slot, $\sum_{i=1}^{N} W_i \in [CT + B + 1 - \sum_{i=1}^{N} \alpha_i, CT + B]$ in the slot just before congestion, and $\sum_{i=1}^{N} W_i \in [CT + B + 1, CT + B + \sum_{i=1}^{N} \alpha_i]$ in the slot of congestion. As a result, anywhere from 1 to $\sum_{i=1}^{N} \alpha_i$ packets could be dropped at one congestion event, and we know that the congestion is heavier if the increment before congestion is larger. Approximately, we can assume that M takes values from 1 to $\sum_{i=1}^{N} \alpha_i$ with equal probability, and thus $M_{\max} = \max(1, \sum_{i=1}^{N} \alpha_i)$, and $\hat{M} = \max(1, (1 + \sum_{i=1}^{N} \alpha_i)/2)$. Since TCP-Illinois chooses very small $\alpha \ll 1$ just before congestion, $M_{\max} \ll N$ and light congestion condition is satisfied. From this analysis, one advantage of TCP-Illinois is that it avoids heavy congestion and synchronized backoff, and it reaches a fair resource allocation between different users. On the contrary, convex curve algorithms, like HS-TCP, yield heavy congestion regularly, and this further causes synchronization and unfairness, as shown in Section 6.

We finally perform Matlab simulations to support our assumption of small $x_i[k]$ variance and validate our analysis on the relationship between heaviness of congestion and fairness. We have performed a large number of simulations on the evolution of the Markov Chain defined in (17) and (18). We vary N (the number of users) from 4 to 10. For each N, we select three probability distributions of M: (i) light congestion, M is uniformly distributed in $[1, N/2]$; (ii) medium level congestion, M is uniformly distributed in $[1, N]$; (iii) heavy congestion, M is uniformly distributed in $[1, 2N]$. For each scenario, we perform 50 simulations. For each simulation, $\mathbf{x}[0]$ and $T_i, \forall i$ are randomly generated initially, $\gamma_i, \forall i$ are computed by (15), and $M[k]$ and $s[k]$ are randomly generated according to Assumption 2 at each congestion event k, and thus each $A[k]$ and the sample path of the Markov Chain are derived. For each sample path, we average 1000 congestion events after the distribution converges, to compute average throughput $\bar{x}_i$, standard deviation of throughput $\text{Std}(x_i) := \sqrt{\text{Var}(x_i)}$, for each user i. We also compute x_i^* for all i by assuming that (27) holds for $\mu = 0$, i.e., all users share the same window size. We plot $\bar{x}_i/x_i^*$ and $\text{Std}(x_i)/\bar{x}_i$ for all users and all simulations performed, and plot $\log(x_i)$ vs $\log(T_i)$ for all users in each simulation. The results are shown from Figs. 2–7.[6] From the figures, we have the following observations: (i) the $\bar{x}_i/x_i^*$ ratio is very close to 1 for light congestion, which indicates that all users share almost the same window size under light congestion, and as the congestion becomes heavy, the range of this ratio becomes wider and thus the difference between W_i's becomes larger; (ii) the $\text{Std}(x_i)/\bar{x}_i$ ratio is always much smaller than 1 for any N and any heaviness of congestion, which supports our assumption of small variance; (iii) $\log(x_i)$ is linear with $\log(T_i)$, which validates the fairness property in (27): $x_i \propto 1/T_i^{1+\mu}$, where $\mu \in [0, 1]$, and μ increases as the congestion becomes heavy.

Remark 1. In Eq. (27), the average of W and x is over their values at the congestion events, and not over all time. Since a general AIMD algorithm can yield any window size curve, it is a challenging problem to compute the average W and x over all time, and it is an open problem whether the above conclusion on fairness holds for time averages of W and x. However, for general AIMD algorithms, since the all time average of W_i lies between $E[W_i[k]]$ and $E[W_i[k^+]] = E[(1 - \hat{\beta} q_i[k])W_i[k]] \ge (1 - \beta_{\max})E[W_i[k]]$, we know that time average of W for one user is similar to the average at congestion events. For TCP-Illinois, in particular, since x increases to near full utilization very quickly and stays around full utilization for a long time, the time average of x is very close to $E[x[k]]$, and thus the

[6] Due to space limitation, we only provide the case for $N = 4$ and 10. For other values of N, the results turn out to be similar.

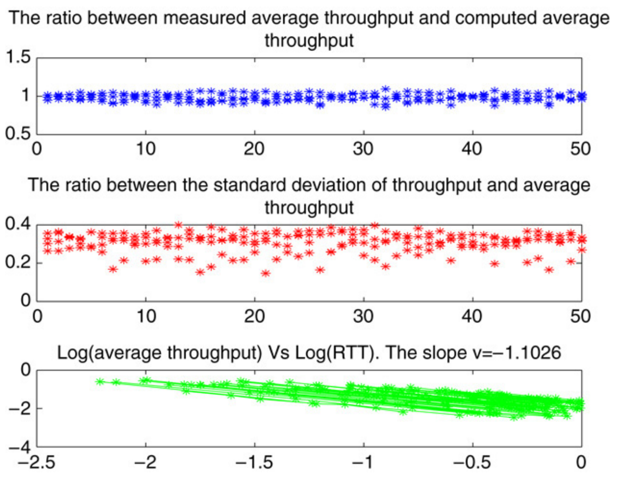

Fig. 2. $N = 4$, light congestion, $M \in [1, N/2]$, uniformly distributed.

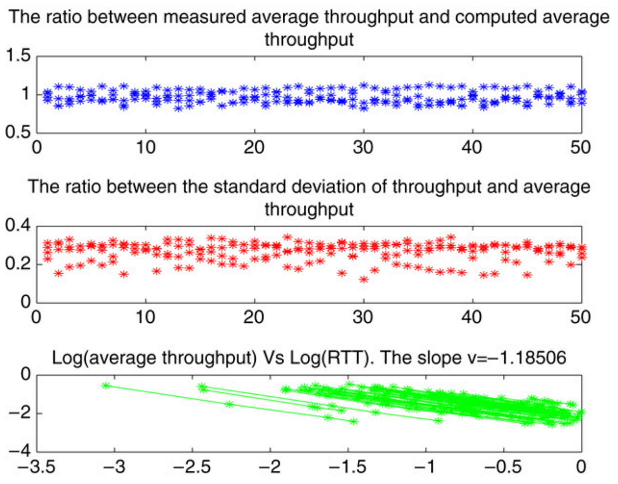

Fig. 3. $N = 4$, medium level congestion, $M \in [1, N]$, uniformly distributed.

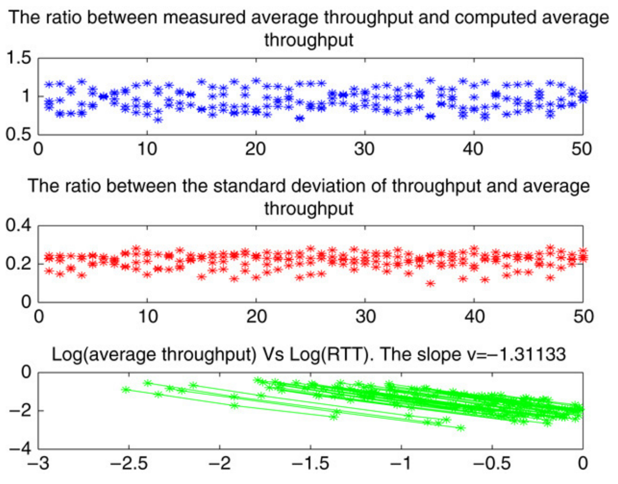

Fig. 4. $N = 4$, heavy congestion, $M \in [1, 2N]$, uniformly distributed.

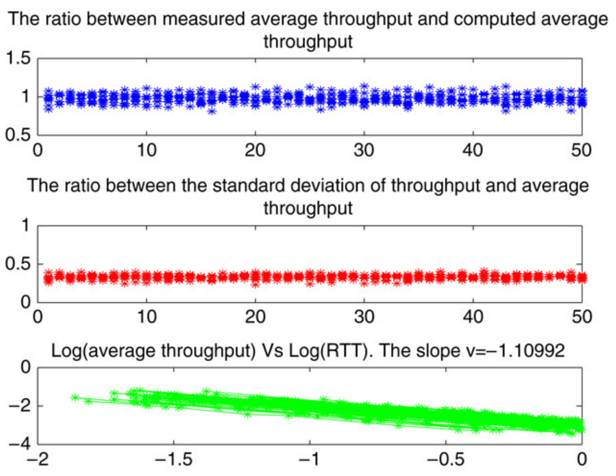

Fig. 5. $N = 10$, light congestion, $M \in [1, N/2]$, uniformly distributed.

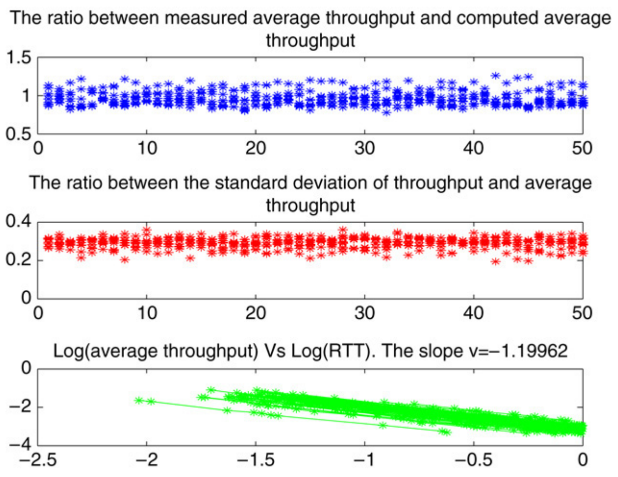

Fig. 6. $N = 10$, medium level congestion, $M \in [1, N]$, uniformly distributed.

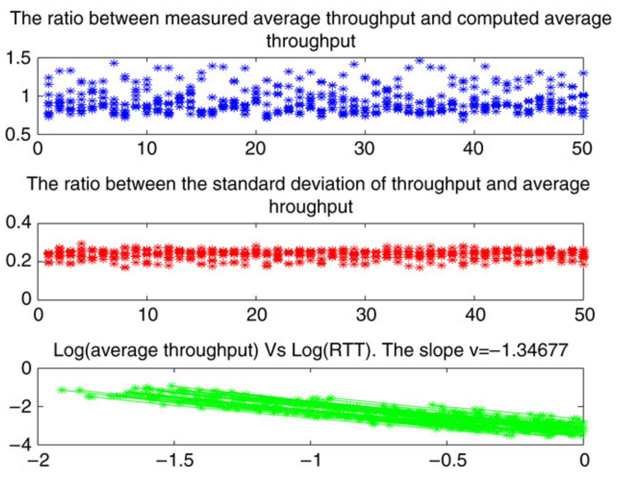

Fig. 7. $N = 10$, heavy congestion, $M \in [1, 2N]$, uniformly distributed.

fairness property should hold approximately for time average also. From our *ns-2* simulations in Section 6, we observe that the time averages of W for different users are also approximately the same.

4.6. Compatibility with the standard TCP

We now consider the equilibrium allocation when TCP-Illinois coexists with TCP-Reno (NewReno and SACK are the same) and all flows share the same RTT.[7] If we use $\hat{T}$ to denote the common maximum RTT for all flows, then $W_i[k] = x_i[k]\hat{T}$. The users are divided into two classes, Illinois user set $\mathbb{I}$ and Reno user set $\mathbb{R}$, and $\alpha_i(t) = \alpha_{IL}(t), \beta_i[k] = \beta_{IL}[k], \forall t, \forall k, \forall i \in \mathbb{I}, \alpha_j(t) \equiv 1, \beta_j[k] \equiv 1/2, \forall t, \forall k, \forall j \in \mathbb{R}$. Define

$$\bar{\alpha}_i[k] = \frac{\int_{t_k}^{t_{k+1}} \frac{\alpha_i(t)}{T(t)} \mathrm{d}t}{\int_{t_k}^{t_{k+1}} \frac{1}{T(t)} \mathrm{d}t}. \tag{28}$$

Then, we know that $\bar{\alpha}_i[k] \equiv 1, \forall k, \forall i \in \mathbb{R}$ and $\alpha_{\min} \le \bar{\alpha}_i[k] = \alpha_j[k] \le \alpha_{\max}, \forall k, \forall i, j \in \mathbb{I}$, and we have

$$W_i[k+1] = (1 - \beta_i[k]D_i[k])W_i[k] + \bar{\alpha}_i[k] \int_{t_k}^{t_{k+1}} \frac{1}{T(t)} \mathrm{d}t$$

From similar steps in (13)–(16), we have

$$W_i[k+1] = (1 - \beta_i[k]D_i[k])W_i[k] + \frac{\bar{\alpha}_i[k]}{\sum_{i=1}^{N} \bar{\alpha}_i[k]} \sum_{i=1}^{N} \beta_i[k]D_i[k]W_i[k]. \tag{29}$$

Suppose $\bar{\alpha}_i[k]$ is independent of $W_i[k]$, and define $\hat{\alpha}_i = E[\bar{\alpha}_i[k]]$, and $\hat{\beta}_i = E[\beta_i[k]]$. Then, $\hat{\alpha}_i = 1, \hat{\beta}_i = 1/2, \forall i \in \mathbb{R}$, and $\hat{\alpha}_j = \hat{\alpha}_k, \hat{\beta}_j = \hat{\beta}_k, \forall j, k \in \mathbb{I}$. We use α_{IL}^* and β_{IL}^* to denote the common $\hat{\alpha}_j$ and $\hat{\beta}_j$ values for TCP-Illinois users. If all packets dropped are due to congestion, and Droptail is used, $\beta_{IL}^* \approx \beta_{\max}$. Taking conditional expectation of $W_i[k+1]$ given $W_i[k]$, we get

$$\left(\sum_{j=1}^{N} \hat{\alpha}_j\right) E[W_i[k+1]|\mathbf{W}[k]] = \left(\sum_{j=1}^{N} \hat{\alpha}_j\right)\left(W_i[k] - \hat{\beta}_i f\left(\frac{W_i[k]}{\hat{T}}\right) W_i[k]\right) + \hat{\alpha}_i \sum_{j=1}^{N} \hat{\beta}_j f\left(\frac{W_j[k]}{\hat{T}}\right) W_j[k].$$

Equating $E[W_i[k+1]]$ and $E[W_i[k]]$, we have

$$E\left[W_i[k] f\left(\frac{W_i[k]}{\hat{T}}\right)\right] \frac{\hat{\beta}_i}{\hat{\alpha}_i} = C_2, \tag{30}$$

where C_2 is a constant independent of user i. If the congestion is light and the backoff is unsynchronized ($\hat{M} \approx 1$ or $\hat{M} \ll C/x$), $f(x)$ is approximately proportional to x, and thus we have that $\frac{\hat{\beta}_i}{\hat{\alpha}_i} E[(W_i[k])^2]$ is the same for all users i. And as $\mathrm{Var}(W_i[k]] \ll (E[W_i[k]])^2$, approximately we have that $\frac{\hat{\beta}_i}{\hat{\alpha}_i} (E[W_i[k]])^2$ is the same for all users i.

Since $\hat{\alpha}_i$ and $\hat{\beta}_i$ are the same within each protocol, we know that at equilibrium, all Reno users share the same average window size $\bar{W}_R$ and all Illinois users share the same average window size $\bar{W}_{IL}$, and

$$\frac{\bar{W}_{IL}}{\bar{W}_R} \approx \sqrt{\frac{\alpha_{IL}^*}{2\beta_{IL}^*}} \approx \sqrt{\frac{\alpha_{IL}^*}{2\beta_{\max}}}.$$

In simulations presented later, for typical scenarios, $\sqrt{\alpha_{IL}^*}$ is slightly larger than 1 and $\beta_{\max}$ is usually picked to be 1/2 for friendliness with TCP-Reno. Thus, $\bar{W}_{IL}$ is usually slightly larger than $\bar{W}_R$. This means that in a network with both TCP-Illinois and TCP-Reno users, the TCP-Reno users will not suffer a significant degradation in performance. Furthermore, unlike TCP-Vegas which performs poorly when used with TCP-Reno, TCP-Illinois actually performs better than TCP-Reno, thus providing the right incentive for users to switch to TCP-Illinois.

[7] Due to lack of space, here we considered only the homogeneous RTT case. For the heterogeneous RTT case, a similar method can be used to derive the equilibrium allocation.

5. TCP-Illinois properties

In Section 2, we listed some requirements for the new TCP variant to satisfy, and in Section 4, we showed that TCP-Illinois maintains the intra-protocol fairness the same way as standard TCP, satisfies the stability and scalability requirement, avoids heavy congestion, and is compatible with the current TCP. In this section, we consider the remaining requirements.

Since q_a increases with increasing W, α decreases with increasing W, and thus the W curve is concave. We can show that the curve is actually at first linear, and then close to a parabola, and finally linear again. The proof is omitted due to space limitations and is available in [19]. In [19], we also show that TCP-Illinois achieves a better average throughput than standard TCP for any router buffer size B, and its average throughput increases as B increases, since compared with standard TCP, TCP-Illinois increases its rate to full utilization faster and stays around full utilization longer (the length of time it stays around full utilization increases with increasing B). Thus the requirements of efficiency, router buffer independence, and incentive to switch are all met.

From Section 4, we see that convergence speed in k for TCP-Illinois is the same as that for standard TCP. So the response time is only determined by the time interval between two consecutive congestion events. We can show that this time interval of TCP-Illinois is similar to or smaller than that of standard TCP for a wide range of $\alpha_{\min}$ values (see [19] for a proof), and thus the responsiveness requirement is also satisfied.

In lossy networks such as wireless networks, many packets are dropped not due to congestion. These packet drops greatly reduce the throughput for standard TCP, but for TCP-Illinois, the degradation is not as severe, since when a packet is dropped before congestion, the average queueing delay is always almost zero, and thus $\beta \approx \beta_{\min}$ and $\alpha \approx \alpha_{\max}$ always, and TCP-Illinois is essentially an AIMD algorithm with a larger $\alpha = \alpha_{\max}$ and smaller $\beta_{\min}$. Since $W \propto \sqrt{\alpha/\beta p}$, the ratio of the average window size of TCP-Illinois to that of standard TCP can be up to

$$\sqrt{\alpha_{\max}/(2\beta_{\min})}. \tag{31}$$

This improvement is significant. For example, if $\alpha_{\max} = 9$, $\beta_{\min} = 1/8$, then W_{Illinois} can be up to $6W_{\text{Reno}}$.

6. Simulation results

In this section, we provide *ns-2* simulation results to validate the properties of TCP-Illinois and compare its performance with TCP-Reno, HS-TCP, BIC-TCP, C-TCP and TCP-Vegas. HS-TCP and TCP-Vegas are in the standard *ns-2* simulation package, and we use the Linux TCP implementation for *ns-2* [33] to simulate BIC-TCP and C-TCP. For HS-TCP, BIC-TCP and C-TCP, we use the default parameter settings. For TCP-Vegas, we let $\alpha = \beta = \gamma$ and vary the γ values. For TCP-Illinois, without explicit explanation, we choose the following standard parameter setting: $\alpha_{\max} = 10, \alpha_{\min} = 0.1, \beta_{\max} = 1/2, \beta_{\min} = 1/8, W_{\text{thresh}} = 10, \eta_1 = 0.01, \eta_2 = 0.1, \eta_3 = 0.8$, and $\theta = 5$. Throughout, we consider a dumb-bell network topology with single bottleneck link shared by one or multiple users. For each simulation scenario, we run the simulation for a sufficiently long time so that tens to hundreds of cycles (congestion epochs) are covered in one simulation run.

6.1. Single user: Efficiency property

We first perform simulations for a single user (choosing one of the above algorithm) scenario, with $C = 100$ Mbps, $B = 100$ packets,[8] and $T_p = 100$ ms. The window sizes are plotted in Fig. 8. The simulations clearly demonstrate the concave nature of the curve of TCP-Illinois and show that TCP-Illinois achieves a larger average window size than TCP-Reno. For HS-TCP, we have chosen the Reno base, NewReno base and SACK base, and we have found that HS-TCP generates timeouts frequently for Reno and NewReno bases, and only works well if SACK is used.[9] This supports our claim that HS-TCP causes heavy congestion. From the figure, we also see that BIC-TCP has a concave–convex curve, and the window curve for C-TCP is not concave either. We then compare the average goodputs for different protocols, and plot them in the left subplot of Fig. 9. From the figure, it can be seen that TCP-Illinois has the best network bandwidth utilization among these algorithms.

[8] The packet size is 1000 bytes throughout.

[9] Henceforth, we mean SACK-based HS-TCP when we mention HS-TCP without specifying its base. BIC-TCP and C-TCP are also SACK based.

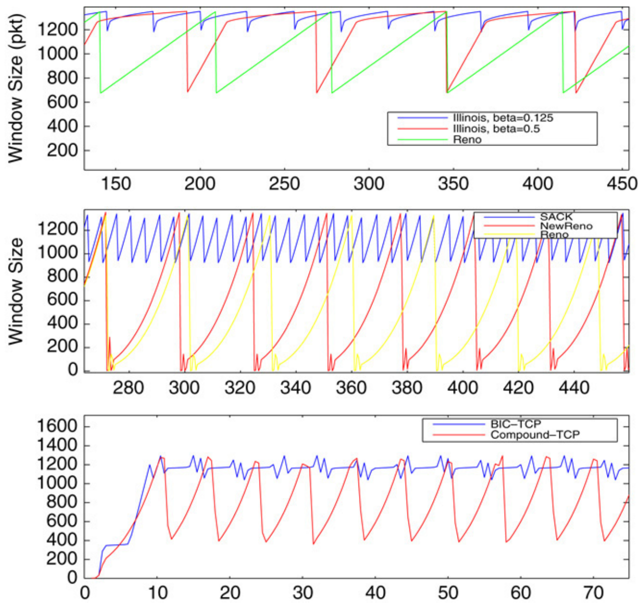

Fig. 8. Single user, TCP-Reno, HS-TCP, TCP-Illinois, BIC-TCP, and C-TCP. Top plot: Reno and Illinois ($\beta = 0.125$ and $\beta = 0.5$). Middle plot: HS-TCP, with Reno, NewReno, and SACK bases. Bottom plot: BIC-TCP and C-TCP.

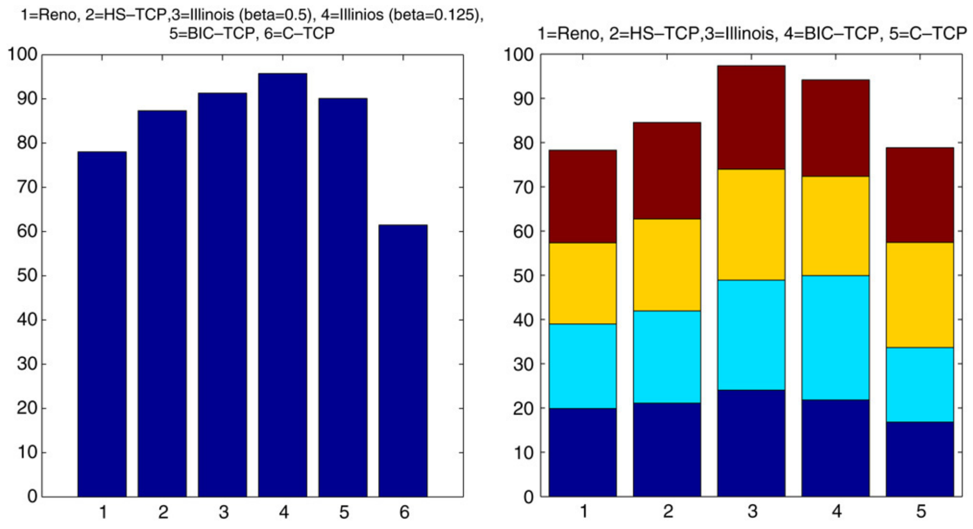

Fig. 9. Left: Average goodput of 1 users. From left to right: Reno, HS-TCP, Illinois with $\beta = 0.5$, Illinois with $\beta = 0.125$, BIC-TCP, C-TCP. Right: Average goodputs of 4 homogeneous RTT (100 ms) users. From left to right: Reno, HS-TCP, Illinois, BIC-TCP, C-TCP.

We then study the effect of the buffer limit on the window size curve. We fix $C = 10$ Mbps, $T_p = 60$ ms, and vary B from 10 to 50 packets, and the window curve is plotted in Fig. 10. It is clear that as B increases, there is more time around full utilization and the average window size increases.

6.2. Multiple users: Fairness property

We now perform simulations for multiple ($N = 4$) users. The network parameter is still $C = 100$ Mbps and $B = 100$ packets. For homogeneous RTT scenarios, $T_p = 100$ ms for all users; for heterogeneous RTT scenarios, the RTTs for the four flows are 60, 80, 100, and 120 ms, respectively.

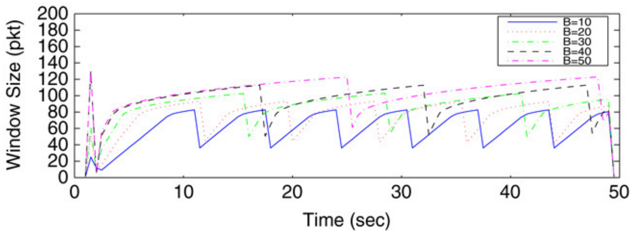

Fig. 10. The window size curve for different buffer sizes.

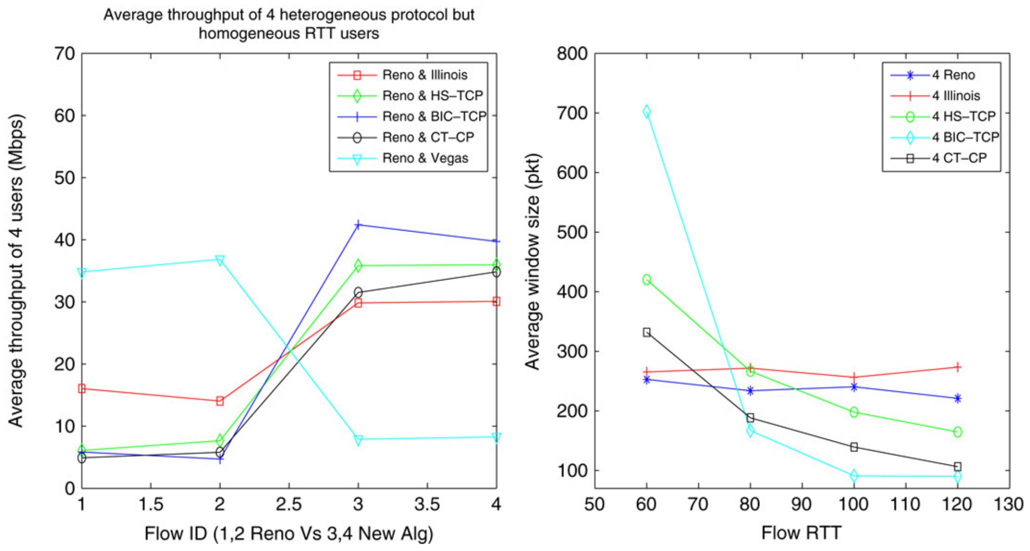

Fig. 11. Left: Average throughput of four homogeneous RTT users, two use Reno, two use a new algorithm. Right: Average window of four heterogeneous RTT users, all use the same algorithm.

We first simulate on the same protocol homogeneous RTT scenario, and plot average goodputs in the right subplot of Fig. 9. We see that in the multiple user case, TCP-Illinois is still the most efficient algorithm.

We then demonstrate the inter-protocol fairness (the bandwidth allocation when Reno and a new algorithm coexist) in a homogeneous RTT scenario; and demonstrate the intra-protocol fairness (RTT fairness) of these algorithms in a heterogeneous RTT scenario. The average throughput in the homogeneous RTT inter-protocol scenario and average window size in the heterogeneous RTT intra-protocol scenario are plotted in Fig. 11. From this figure, we see clearly that TCP-Illinois is more fair to the competing Reno user and large RTT users than HS-TCP, BIC-TCP and C-TCP. For inter-protocol fairness, we also simulate TCP-Vegas, and we see that TCP-Vegas users get very small rates when competing with TCP-Reno. Therefore, TCP-Vegas is not compatible with TCP-Reno and provides no incentives for Reno users to switch.

To further demonstrate the performance of these protocols, we also plot the window curves of these simulations in Figs. 12 and 13. From all these figures, it can be seen that TCP-Illinois outperforms HS-TCP, BIC-TCP and C-TCP in terms of efficiency, fairness, and compatibility, and is the only algorithm which yields a concave window curve throughout a congestion epoch.

6.3. *Performance in lossy/wireless networks*

We then perform simulations for lossy/wireless links. It is a single link single user scenario, with the user choosing either TCP-Reno or TCP-Illinois, and the link randomly dropping packets with dropping probability p_d much larger than the congestion loss probability (since p_d is large, the link is under-utilized and there is no congestion loss at all in many cases). The capacity and buffer length of the link are 40 Mbps and 200 packets, respectively, and the propagation delay for the single user is 100 ms. Instead of choosing the default setting, the TCP-Illinois user sets

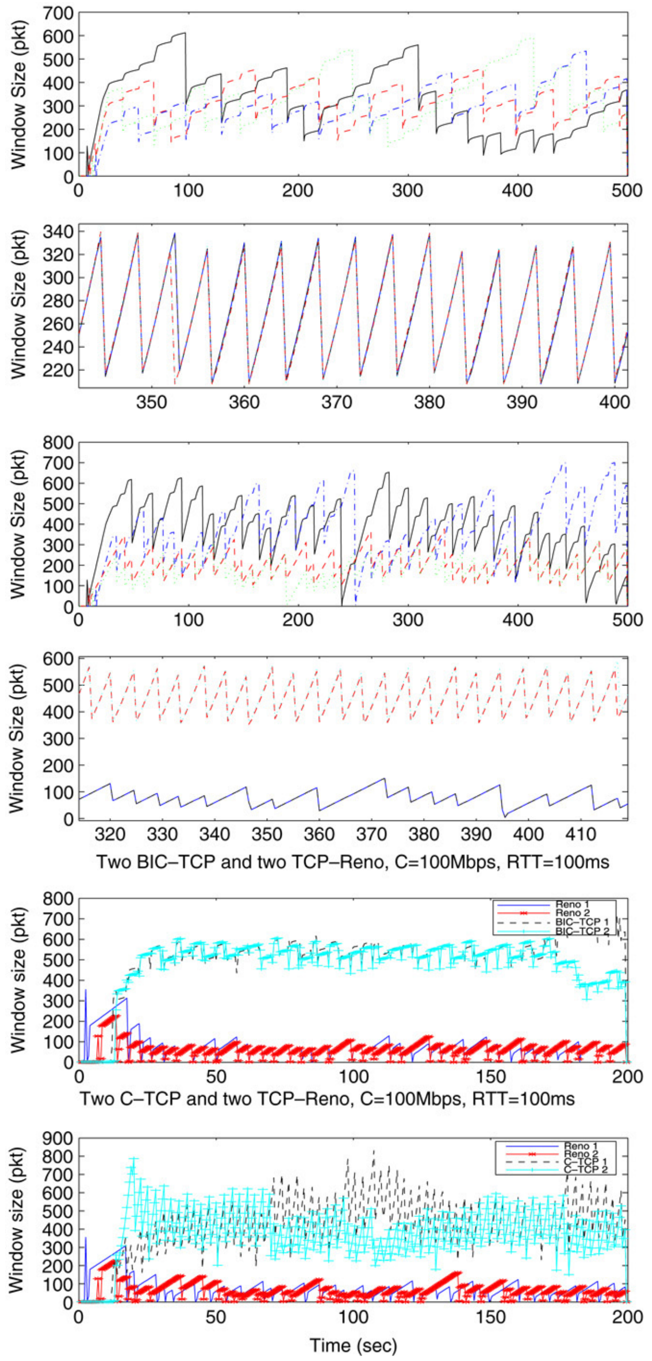

Fig. 12. Homogeneous RTT Users. First: 4 TCP-Illinois. Second: 4 Hs-TCP. Third: 2 TCP-Reno and 2 TCP-Illinois. Fourth: 2 TCP-Reno and 2 HS-TCP. Fifth: 2 TCP-Reno and 2 BIC-TCP. Sixth: 2 TCP-Reno and 2 C-TCP. TCP-Illinois is demonstrated to avoid synchronization effectively and to be compatible with TCP-Reno. Algorithms like HS-TCP are demonstrated to generate synchronization frequently and to be more unfair to TCP-Reno.

$\eta 1 = 0.2$. We vary p_d values from 0.0005 to 0.05, and plot the average window size[10] for TCP-Illinois and Reno and the ratio of these two multiplied by 20, as in the left plot of Fig. 14. From the plot, we see that $W_{\text{Illinois}} \approx 4W_{\text{Reno}}$

[10] We had only one run for each simulation scenario, but this run lasts for a long time. We choose different time intervals to compute the average window size, and to compute the confidence interval from the averages over different time intervals.

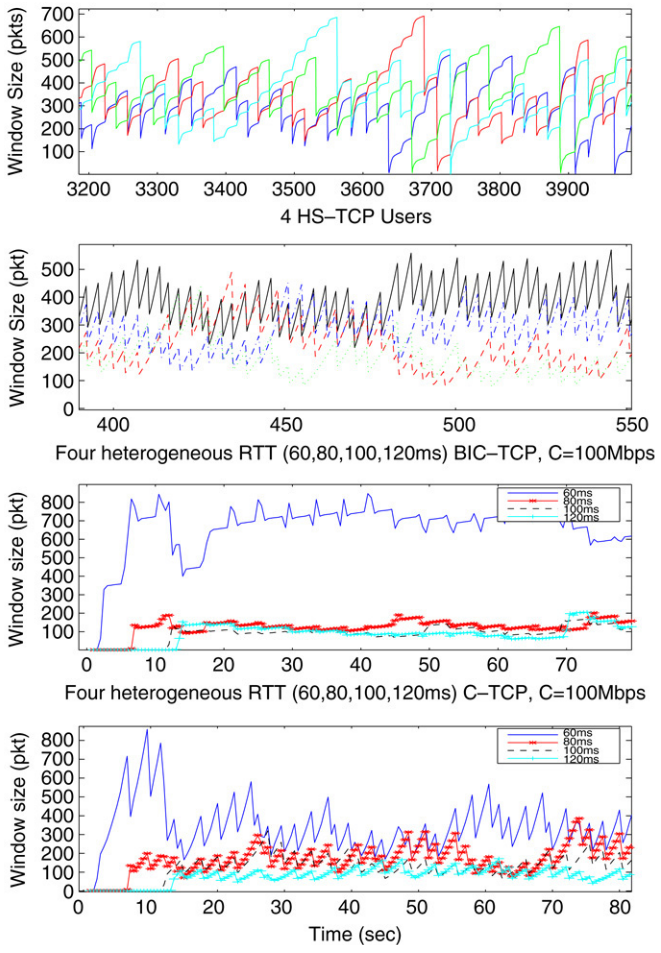

Fig. 13. Heterogeneous RTT Users. First: 4 TCP-Illinois. Second: 4 HS-TCP. Third: 4 BIC-TCP. Fourth: 4 C-TCP. TCP-Illinois and Reno are fair to large RTT users, while HS-TCP, BIC-TCP, C-TCP are not.

in most cases. From (31), the ratio should be $\sqrt{\alpha_{\max}/(2\beta_{\min})} = \sqrt{40} \approx 6.32$. The difference of the ratio between simulation and analysis can be explained if we observe the window curve plot of TCP-Illinois and Reno, as in the right plot of Fig. 14. From the window curve plot, we see that timeout happens frequently for the TCP-Illinois user, since the increment amount α is very large before a packet loss happens. Eq. (31) only considers the congestion avoidance phase, and timeout is the reason that $W_{\text{Illinois}}/W_{\text{Reno}}$ is around 4 instead 6. However, TCP-Illinois achieves a much better throughput than Reno in wireless networks.

6.4. *Performance with noisy RTT measurement*

We finally perform simulations to compare the performance of TCP-Illinois and TCP-Vegas when the delay measurement is inaccurate. We consider a single link and two user scenario. For the link, $C = 10$ Mbps and $B = 50$ packets (correspondingly, the maximum queueing delay $d_m = 40$ ms). For the users, either both choose TCP-Illinois or both choose TCP-Vegas, and the propagation delay is $T_p = 60$ ms for each user. We now suppose that there is an extra white noise term in the RTT measurement, denoted by n, and let n be uniformly distributed between $[0, 2\sigma]$ (the noise term is an extra delay due to reasons other than propagation and queueing, so it is non-negative). Then, $\text{RTT} = T_p + d + n$, where d is the queueing delay. We vary the value of σ to vary the noise level and study the performance of TCP-Vegas and TCP-Illinois under noisy RTT measurement. For each protocol, we have two groups of simulations. In group one, both users face the noise term in the RTT measurement; and in group two, only one user

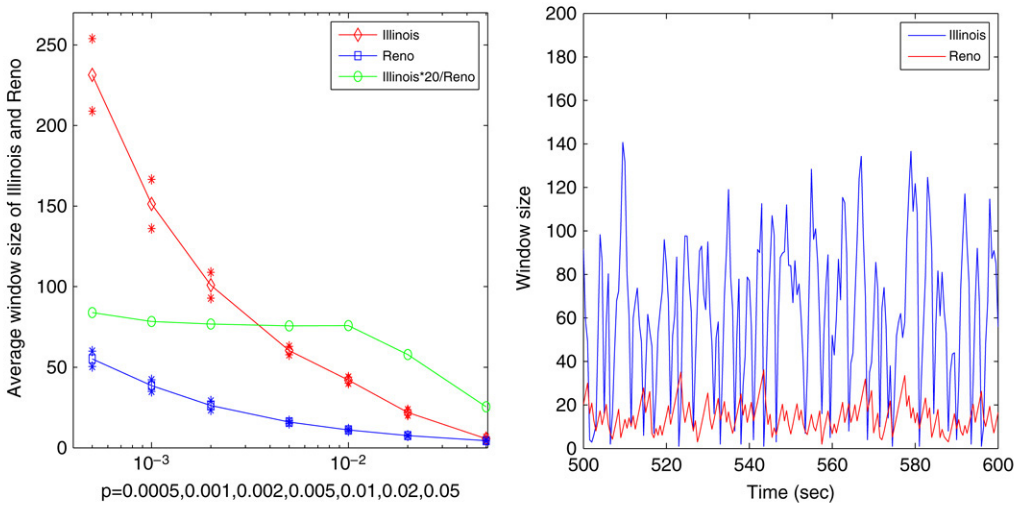

Fig. 14. Window sizes of TCP-Illinois vs Reno in Lossy Networks. Left: Average window vs p. The squares and diamonds connected by the solid lines are the average values. The stars above and below the average values are the average values plus/minus standard deviations. So the distance between the stars gives the confidence interval with the confidence level being 0.95. Right: Window curve over t for $p = 0.005$.

faces the noise term and the other user measures RTT accurately. The average throughput of the users under different noise levels are plotted in Fig. 15. From Fig. 15, we see that as the noise level increases, TCP-Illinois is very robust to noise, while TCP-Vegas is not: there is a threshold on σ, which depends on the γ parameter. If the noise level exceeds this threshold, the performance is degraded significantly: if both users have noise terms, both get a much smaller throughput than the noise free case and the link is under-utilized; if one user has the noise term and the other one does not, then the resource allocation is very unfair to the user with inaccurate RTT information. It is easy to explain the degradation when σ is larger than the threshold. At equilibrium, the Vegas user satisfies

$$\text{diff} = \left(\frac{W}{T_p} - \frac{W}{T_p + d_a}\right) T_p = W \frac{d_a}{T_p + d_a} = \gamma \Rightarrow W^* = \gamma \frac{d_a^* + T_p}{d_a^*}, \tag{32}$$

where d_a^* and W^* are the equilibrium values of d_a and W. So W^* is an decreasing function of d_a^*, and d_a^* is a positive value such that the sum of $W^*/(T_p + d_a^*)$ over all users equals the capacity C. If RTT $= T_p + d_a + n$, then since n may hit zero, still we have BaseRTT $= T_p$, and (32) becomes the following equation:

$$\text{diff} = \left(\frac{W}{T_p} - \frac{W}{T_p + d_a + n}\right) T_p = W \frac{d_a + n}{T_p + d_a + n} = \gamma$$

$$\Rightarrow W = \gamma \frac{d_a + T_p + n}{d_a + n} \quad \text{and} \quad \bar{W} \approx \gamma \frac{\bar{d}_a + T_p + \bar{n}}{\bar{d}_a + \bar{n}}, \tag{33}$$

where $\bar{n}$, $\bar{d}_a$ and $\bar{W}$ are the time average values of n, d_a and W. We see that if $\bar{n} = \sigma \le d_a^*$, we can pick $\bar{d}_a = d_a^* - \sigma$ so that $\bar{W} = W^*$; and if $\bar{n} = \sigma > d_a^*$, then $\bar{W}$ is definitely smaller than W^*. And when $\sigma > d_a^*$, as σ increases, $\bar{W}$ decreases. So there exists a threshold of $\bar{n} = \sigma$ such that if the noise level is smaller than this threshold, the performance can be similar to the noise free case; and if the noise level is larger than this threshold, the performance is degraded and the degradation becomes more significant as σ increases. Since this threshold approximately equals d_a^* and d_a^* is proportional to γ, we know that this threshold is also proportional to γ, as shown in Fig. 15.

7. Conclusion

In this paper, we have considered some natural requirements for a new TCP protocol for high-speed networks and have introduced a class of C-AIMD algorithms, which use loss to determine the *direction* and use delay to adjust the *pace* of window size change. This idea is rooted in the following two assumptions or understanding of the entire

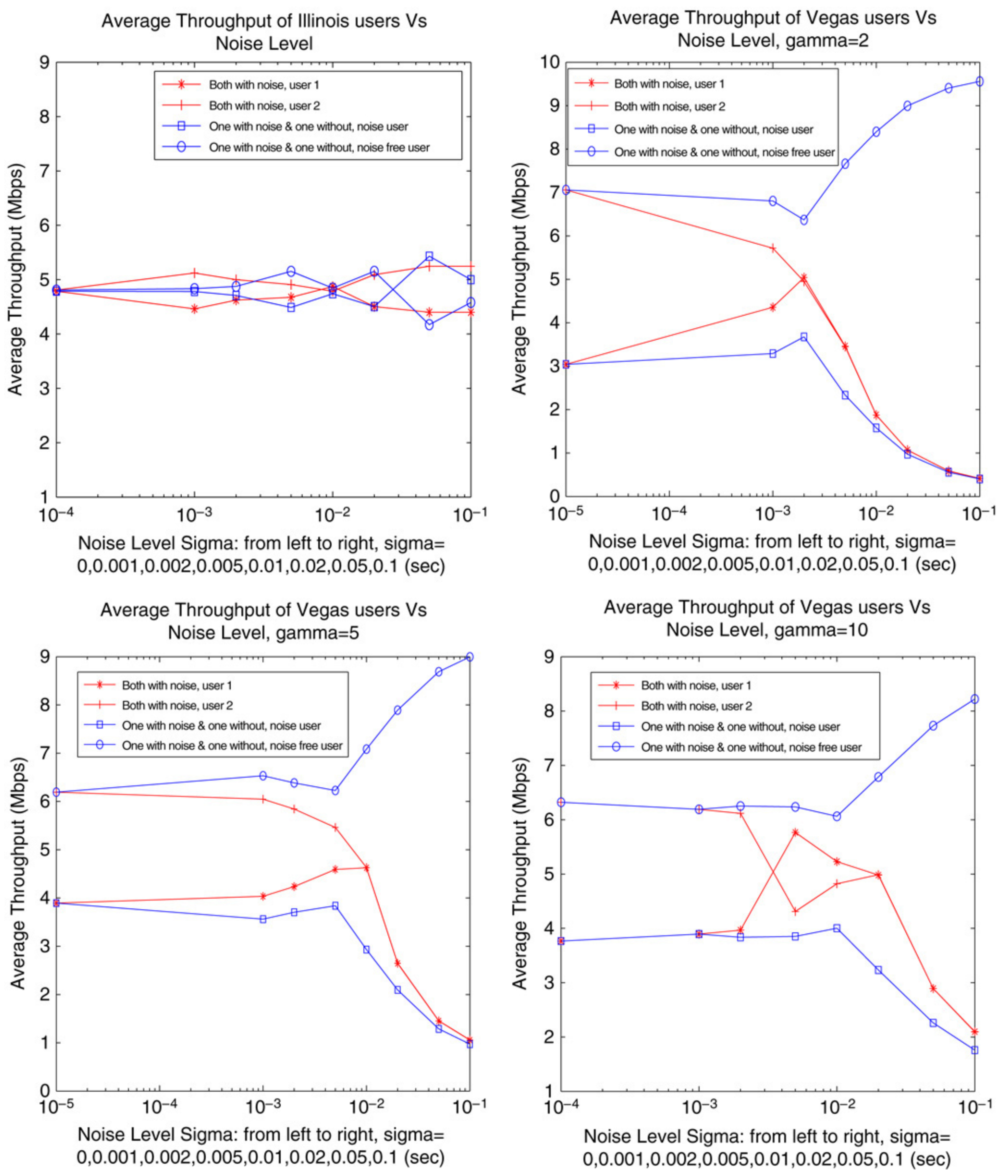

Fig. 15. Average Throughput vs Noise Level. Top left: TCP-Illinois. Top right: Vegas, $\gamma = 2$. Bottom left: Vegas, $\gamma = 5$. Bottom right: Vegas, $\gamma = 10$.

congestion control system: (i) delay is indeed a useful signal, i.e., congestion or packet loss is indeed correlated to delay information; (ii) delay is not an accurate signal, i.e., the correlation between loss and delay is weak. Combining these two, we should use loss as the primary signal and delay as the secondary signal. Using this idea, we have designed a specific protocol called TCP-Illinois, which achieves a concave window size curve and a better throughput than standard TCP, and maintains the fairness of standard TCP. Various properties of TCP-Illinois are studied, and TCP-Illinois is shown to satisfy all the requirements for an ideal high-speed TCP variant.

To analyze the fairness property of TCP-Illinois, a new stochastic matrix model of general AIMD algorithms is introduced. Using this model, we have shown that TCP-Illinois leads to unsynchronized backoff and yields similar window sizes for different RTT users. There are still some open problems regarding the new model, however, which include: (i) the exact relationship between $E[W_i]$ and $E[W_j]$, $\forall i \neq j$ for the $N > 2$ heterogeneous user scenario; (ii), the relationship between $E[\bar{W}_i]$ and $E[\bar{W}_j]$, $\forall i \neq j$, where the expectation is over all time.

Acknowledgements

We thank Professor F. Baccelli and Professor S. Meyn for helpful discussions. We also thank Dr. D.X. Wei for helpful instructions on simulating BIC-TCP and Compound-TCP.

The research was supported by the NSF ITR Grant CCR 00-85917. This is an extended version of [20], a paper with the same title, which was presented at the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS), October 2006.

References

[1] J. Ahn, P. Danzig, Z. Liu, L. Yan, Experience with TCP vegas: Emulation and experiment, in: Proceedings of ACM SIGCOMM, 1995.

[2] E. Altman, D. Barman, B. Tuffin, M. Vojnovic, Parallel tcp sockets: Simple model, throughput and validation, in: Proc. IEEE INFOCOM, Barcelona, Spain, 2006.

[3] F. Baccelli, D. Hong, Interaction of TCP flows as billiards. Technical Report, INRIA Rocquencourt, April 2002.

[4] F. Baccelli, D. Hong, The AIMD model for TCP sessions sharing a common router, in: Proceedings of 39th Annual Allerton Conf. on Communication, Control and Computing, October 2001.

[5] F. Baccelli, D. Hong, AIMD, fairness and fractal scaling of TCP traffic, in: Proceedings of IEEE Infocom, June 2002.

[6] A. Berman, R. Plemmons, Nonnegative Matrices in the Mathematical Sciences. SIAM, 1979.

[7] S. Biaz, N. Vaidya, Is the round-trip time correlated with the number of packets in flight, in: Proceedings Internet Measurement Conference, IMC, October 2003.

[8] L. Brakmo, S. O'Malley, L. Peterson, TCP vegas: New techniques for congestion detection and avoidance, in: Proceedings of ACM SIGCOMM Symposium, August 1994. p 24–35.

[9] S. Floyd, Highspeed TCP for large congestion windows. Internet draft, draft-floyd-tcp-highspeed-01.txt, Available at: http://www.icir.org/floyd/hstcp.html, December 2003.

[10] S. Floyd, T. Henderson, The new reno modification to TCPs fast recovery algorithm, RFC 2582, 1999.

[11] A. Graham, Nonnegative Matrices and Applicable Topics in Linear Algebra, Ellis Horwood Limited, Chichester, England, 1987.

[12] V. Jacobson, Congestion avoidance and control, ACM Computer Communication Review 18 (August) (1988) 314–329.

[13] V. Jacobson, Berkeley TCP evolution from 4.3-tahoe to 4.3-reno, in: Proceedings of the Eighteenth Internet Engineering Task Force, July 1990.

[14] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, S. Singh, FAST TCP: From theory to experiments, April 2003.

[15] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, in: Proceedings on ACM Sigcomm, 2002.

[16] T. Kelly, On engineering a stable and scalable TCP variant, Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR.435, June 2002.

[17] D. Leith, R. Shorten, Analysis and design of synchronised communication networks, Automatica 41 (2005) 725–730.

[18] D. Leith, R. Shorten, Y. Li, H-TCP: A framework for congestion control in high-speed and long-distance networks, HI technical report. Available at: http://www.hamilton.ie/net/htcp/, August 2005.

[19] S. Liu, T. Başar, R. Srikant, TCP-illinois: A delay and loss-based congestion control algorithm for high-speed networks, technical report, UIUC. Available at: http://www.ews.uiuc.edu/˜shaoliu/tcpillinois_full.pdf, 2006.

[20] S. Liu, T. Başar, R. Srikant, TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks, in: Proc. First International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS, Pisa, Italy, October 2006.

[21] J. Martin, A. A. Nilsson, I. Rhee, Delay-based congestion avoidance for TCP, IEEE/ACM Transactions on Networking (June) (2003) 356–369.

[22] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP selective acknowledgement options. RFC 2018. Available at: http://www.icir.org/floyd/sacks.html, April 1996.

[23] B. Miller, K. Avrachenkov, K. Stepanyan, G. Miller, Flow control as stochastic optimal control problem with incomplete information, in: Proc. IEEE INFOCOM, Miami, FL, 2005.

[24] J. Mo, R. J. La, V. Anantharam, J. C. Walrand, Analysis and comparison of TCP reno and vegas, in: Proceedings of IEEE INFOCOM, 1999.

[25] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: A simple model and its empirical validation, in: Proceedings of ACM SIGCOMM, 1998.

[26] R. S. Prasad, M. Jain, C. Dovrolis, On the effectiveness of delay-based congestion avoidance, in: Proceedings of Second International Workshop on Protocols for Fast Long-Distance Networks, 2004.

[27] I. Rhee, L. Xu, CUBIC-TCP: A new TCP-friendly high-speed TCP variant, White paper. Available at: http://www4.ncsu.edu/˜rhee/export/bitcp/cubic-paper.pdf.

[28] H. Shimonishi, T. Hama, T. Murase, TCP-Adaptive Reno: Improving Efficiency-Friendliness Tradeoffs of TCP Congestion Control Algorithm, PFLDnet, February 2006.

[29] R. Shorten, F. Wirth, D. Leith, A positive systems model of TCP-like congestion control: Asymptotic results, IEEE/ACM Transactions on Networking 14 (3) (2006) 616–629.

[30] W. Stevens, TCP/IP Illustrated, in: The Protocols, vol. 1, Addison-Wesley, 1994.

[31] K. Tan, J. Song, Q. Zhang, M. Sridharan, A compound TCP approach for high-speed and long distance networks, in: Proceedings of IEEE Infocom, April 2006.

[32] A. Tang, J. Wang, S. Low, M. Chiang, Equilibrium of heterogeneous congestion control protocols, in: Proceedings of IEEE Infocom, Miami, FL, March 2005.
[33] D. X. Wei, P. Cao, Ns-2 tcp-linux: An ns-2 tcp implementation with congestion control algorithms from linux, in: Proceeding from the 2006 Workshop on ns-2: The IP Network Simulator, Pisa, Italy, 2006.
[34] F. Wirth, R. Stanojevic, R. Shorten, D. Leith, Stochastic equilibria of AIMD communication networks, SIAM Journal on Matrix Analysis and Applications 28 (3) (2006) 703–723. Available at http://www.hamilton.ie/chris/SIMAX_july28.pdf.
[35] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control for fast long-distance networks, in: Proceedings of IEEE INFOCOM, 2004.
[36] A. Zanella, G. Procissi, M. Gerla, M. Y. Sanadidi, TCP westwood: Analytic model and performance evaluation, in: Proceedings of IEEE Globecom, 2001.

Shao Liu (S'05) received B.S. degree from Peking University, Beijing, and M.S. and Ph.D. degrees in electrical engineering from University of Illinois at Urbana-Champaign. He is currently with Princeton University, where he is a Postdoctoral Researcher in the Department of Electrical Engineering. His research interests include congestion control for communication networks, service differentiation and quality of service, peer-to-peer streaming and IPTV. His email address is: shaoliu@princeton.edu.

Tamer Başar (S'71-M'73-SM'79-F'83) received B.S.E.E. from Robert College, Istanbul, and M.S., M.Phil, and Ph.D. from Yale University. After stints at Harvard University and Marmara Research Institute (Gebze, Turkey), he joined the University of Illinois at Urbana-Champaign in 1981, where he is currently the Swanlund Endowed Chair in the Department of Electrical and Computer Engineering, Center for Advanced Study Professor, and Research Professor in the Coordinated Science Laboratory and the Information Trust Institute. He has published extensively on systems, control, communications, and dynamic games, and has current research interests in modeling and control of communication networks, control over heterogeneous networks, resource management and pricing in networks, and security and trust in computer systems.

Dr. Başar is the Editor-in-Chief of Automatica, Editor of the Birkhäuser Series on Systems & Control, Managing Editor of the Annals of the International Society of Dynamic Games (ISDG), and a member of editorial and advisory boards of several international journals. He has received several awards and recognitions over the years, among which are the Medal of Science of Turkey (1993), Distinguished Member Award (1993), Axelby Outstanding Paper Award (1995) and Bode Lecture Prize (2004) of the IEEE Control Systems Society (CSS), Quazza Medal (2005) of IFAC, and Bellman Control Heritage Award (2006) of the American Automatic Control Council. He is a member of the National Academy of Engineering, a member of the European Academy of Sciences, a Fellow of IEEE, a Fellow of IFAC, a past president of CSS, and a past (founding) president of ISDG.

R. Srikant (S'90-M'91-SM'01-F'06) received his B.Tech. from the Indian Institute of Technology, Madras in 1985, his M.S. and Ph.D. from the University of Illinois in 1988 and 1991, respectively, all in Electrical Engineering. He was a Member of the Technical Staff at AT&T Bell Laboratories from 1991 to 1995. He is currently with the University of Illinois at Urbana-Champaign, where he is a Professor in the Department of Electrical and Computer Engineering, and a Research Professor in the Coordinated Science Lab.

He was an associate editor of Automatica and the IEEE Transactions on Automatic Control, and is currently an associate editor of the IEEE/ACM Transactions on Networking. He has also served on the editorial boards of special issues of the IEEE Journal on Selected Areas in Communications and IEEE Transactions on Information Theory. He was the chair of the 2002 IEEE Computer Communications Workshop in Santa Fe, NM and a program co-chair of IEEE INFOCOM, 2007. His research interests include communication networks, stochastic processes, queueing theory, information theory, and game theory.