Active Queue Management for Fair Resource Allocation in Wireless Networks

Lachlan L.H. Andrew, *Senior Member*, *IEEE*, Stephen V. Hanly, *Member*, *IEEE*, and Rami G. Mukhtar, *Member*, *IEEE*

Abstract—This paper investigates the interaction between end-to-end flow control and medium access control (MAC)-layer scheduling on wireless links. We consider a wireless network with multiple users receiving information from a common access point; each user suffers fading and a scheduler allocates the channel based on channel quality but is subject to fairness and latency considerations. We show that the fairness property of the scheduler is compromised by the transport-layer flow control of Transmission Control Protocol (TCP) New Reno. We provide a receiver-side control algorithm, CLAMP, that remedies this situation. CLAMP works at a receiver to control a TCP sender by setting the TCP receiver's advertised window limit, and this allows the scheduler to allocate bandwidth fairly between the users.

Index Terms—Wireless communications, wireless networks, TCP, Transmission Control Protocol, active queue management, multiuser diversity, scheduling, flow control, access networks.

1 Introduction

Transmission Control Protocol (TCP) Reno (and its variants) is the dominant transport-layer (layer-4) protocol for data transfers in the Internet. In the last few years, increasing attention has been drawn to the performance of TCP across wireless networks. In this present paper, we focus on the interaction between TCP, which attempts to fill and overflow network buffers, and a lower layer wireless scheduler that tries to maximize throughput that is subject to fairness constraints. The model that we use is applicable to fading channels in which the scheduler can exploit multiuser diversity [1]. We undertake this study by comparing the TCP throughput with an alternative flow-control algorithm that clamps TCP's bandwidth probing mechanism.

1.1 Motivation

In this study, we investigate the potential impact of the TCP flow control on a lower layer scheduler in terms of the fairness and throughput that it can provide. Although it is already well known that TCP is unfair toward flows with long propagation delays, the studies that have concluded this fact and provided analyses to support it have focused on wireline networks. Wireless networks may be different for a number of reasons.

First, in wireless networks, fairness is typically handled at a lower layer than TCP. The recent trend has been toward a scheduled service at the wireless access points (APs), where the traditional first-come, first-served (FCFS) queue is replaced by a set of queues (perhaps per-receiver queues) with a scheduler allocating the capacity between the different streams. A motivation for this approach is multiuser diversity: It is well known that schedulers can take advantage of channel knowledge to schedule users that have good channel conditions and queue the packets of users who are in bad channel states until they can be rescheduled in more favorable channel states [1].

Second, channel rates cannot be treated as deterministic quantities and there is an interaction between TCP and the link and medium access control (MAC) layers where link rate adaptation and scheduling both respond to fluctuating channel conditions. TCP responds to buffer overflows, which may be caused by these fluctuations.

Third, queuing dynamics become important: Significant queuing is required to average out the lower layer fluctuations, and the amount of buffering required impacts TCP's performance. It is necessary to take account of these issues in models of TCP performance.

In this present paper, we study the performance of TCP over a wireless network by using a model that is rich enough to incorporate the above features. The terminals are mobile receivers downloading information from servers located anywhere in the Internet via a common AP. The AP must schedule the packets as they arrive, but it takes into consideration the current channel conditions of each link, which are fading due to the mobility.

We assume in this paper that the primary cause of interaction between TCP and the lower layers is TCP's fluctuating window size (window halving on packet loss). Since wireless channels are inherently random, we consider an alternative mechanism for window control that leads to fairly static window sizes (in equilibrium) with the window

- *L.L.H. Andrew is with the Department of Computer Science, California Institute of Technology, M/C 256-80, 1200 E. California Blvd., Pasadena, CA 91125. E-mail: l.andrew@ieee.org.*
- *S.V. Hanly is with the ARC Special Research Centre on Ultra-Broadband Information Networks, Department of Electrical and Electronic Engineering, University of Melbourne, Parkville, Australia, 3010. E-mail: s.hanly@ee.unimelb.edu.au.*
- *R.G. Mukhtar is with VaST Systems Technology, Ground Floor, 29 Christie St., St. Leonards, Australia, 2065. E-mail: ramim@ieee.org.*

Manuscript received 4 Mar. 2006; revised 31 Jan. 2007; accepted 30 May 2007; published online 25 June 2007.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0064-0306. Digital Object Identifier no. 10.1109/TMC.2007.70724.

sizes determined by *average quantities*, which are measured at the AP. This alternative control algorithm is itself a contribution of this present paper, and it is motivated by the large body of theoretical work on flow control for the Internet based on the concept of an explicit congestion price signal [2]. Another viewpoint of the results of this present paper is that it provides a study of the potential impact of flow control based on prices explicitly computed by the nodes in the context of wireless networks with the above properties.

We were also motivated by some recent works that use the receiver-side advertised window (*AWND*) as a control mechanism (see [3] and [4]) rather than attempting to change the way that the congestion window (*CWND*) is calculated at the sender. A benefit is that the modification can be implemented entirely within the wireless access network without the need for widespread changes to routers and servers throughout the Internet.

1.2 Approach

In this paper, we propose a novel flow-control mechanism that we use to study the benefit of clamping the TCP window. This is an algorithm, CLAMP, that controls the receiver *AWND* based on feedback from the AP. We consider its operation at the receiver side in conjunction with TCP New Reno at the sender side and compare its performance against that of TCP New Reno without CLAMP.

The first step is to describe the clamping algorithm. This algorithm runs at the mobile device (the receiver) and acknowledgments (ACKs) are sent back to the sender containing the new values of *AWND* as calculated by CLAMP. Since all versions of TCP interpret the *AWND* as an upper limit on the allowable window size, which is a mechanism to avoid an overflow of the receiver buffer, this provides an effective method of control provided that the *AWND* value is smaller than the *CWND* value calculated by the sender.

To provide a simple context in which to introduce CLAMP, we begin in Section 3.2 with a fluid model of a deterministic link. In this model, a fixed-rate link is shared between a number of flows, and CLAMP is the agent for flow control. This model is simple enough to admit an analytical treatment and provides insight into the design of CLAMP, the meaning of the various parameters, and the effect that they have in the equilibrium achieved by the algorithm.

In Section 5, we describe the details of the wireless network scenario, which is the main focus of this present paper. A number of TCP flows are studied which are assumed to be bottlenecked at the AP. Packet loss can occur from the buffer overflow at the AP or from packet loss across the wireless link. The AP implements multiuser diversity by using per-receiver queues, and the scheduler selects which queue to service. We have built a comprehensive simulator that models the interactions among the first four protocol layers (physical, link, network, and transport). At layer 4, TCP New Reno is used at the sender side and the receivers have the option of using CLAMP to suppress the window oscillations of New Reno. We consider experiments in which either CLAMP is turned off in all mobiles or CLAMP is switched on in all mobiles.

The conclusions of this study are presented in Section 6. Our main conclusion is that clamping the TCP window oscillations in the way described in this paper allows the lower layer scheduler to achieve its aims in terms of fairness between users. It also provides fairness between flows that share the same queue, irrespective of their round-trip times (RTTs). These are flows that the scheduler cannot distinguish. Thus, the well-known unfairness of TCP toward flows that have long propagation delays is corrected by the clamping mechanism. CLAMPing the window in the way described in this paper also provides the AP some control over the queuing delays in its buffers.

2 Related Work

A useful survey of TCP enhancements for last-hop wireless networks is provided in [5]. Most work on TCP over wireless channels has focused on the issue of packet loss, its detrimental effect on TCP [6], [7], and the mechanisms at layers 1 and 2 to reduce packet loss rates [8], [9], [10], [11], [12], [13]. Retransmissions can also be handled at layer 4 [14]. The net effect of these mechanisms is to provide a low packet loss rate at the expense of variable packet transmission times [10], [15]. Some recent papers have focused instead on the rate and delay variation that results from a reliable link layer. Packet-level burstiness can lead to packet losses at the AP buffers and the mechanisms of delayed ACKs [16] and ACK regulation [17], [18], [19] attempt to overcome this problem. We also address this latter problem with CLAMP in the context of a cellular fading scenario.

Several recent papers have studied the interaction between TCP and the MAC layer of the wireless local area network (LAN; IEEE 802.11) [20], [21]. Fairness is considered, and a recent paper [22] has highlighted potential instabilities that may result from the interaction between layers. An experiment-based paper analyzing TCP over IEEE 802.11 ad hoc networks [23] shows the benefit of "clamping" the CWND to a value that depends on the number of hops in the network. Recent work to reduce the contention at the MAC layer by generalizing the concept of delayed ACKs is reported in [24].

The 802.11 family of standards does not include channel-based scheduling, as will occur in cellular networks. Papers [17] and [18] consider channel-state-based schedulers, as we do in this present paper. Other papers have considered the interactions between TCP and the MAC layer of cellular systems [25], [26], [27].

Another research thrust is to make TCP more resilient to wireless loss by distinguishing between wireless loss and congestion loss [28], [29]. A new sender-side protocol called "TPC Veno" tries to make this distinction and only halve the window on a congestion-related loss [30]. Recent proposals that modify the TCP sender-side also include TCP Westwood [31], [32] and TCP Jersey [33], which both involve bandwidth estimation at the sender to adjust the transmission rate after detecting congestion. TCP Jersey [33] also uses packet marking at the wireless routers to signify incipient congestion so as to distinguish wireless packet loss from network-congestion-related loss. Other works include

TCP Probing [34], which suspends data transfer and uses a probing technique to ride out a lossy period, TCP Santa Cruz [35], which adapts the *CWND* based on relative delays signaled in the ACKs from the receiver, and "wave and wait" [36], which sacrifices throughput to avoid unnecessary losses and, hence, wasted energy.

It is also possible to consider tighter cross-layer coupling, in which the various layers try to collectively solve a joint optimization problem. There have been many recent papers in this area (see [37] and the references therein). In particular, we mention the rate-based approaches in [38] and [39], which base the layer 4 congestion signal on the queue size, as we do in this paper.

There have been many recent papers concerning TCP over multihop wireless mesh networks [40], [41], [23], [19], [24]. This is an interesting direction for future work, but in this paper, we address the fairness and stability issues associated with TCP in the context of a single-hop wide-area wireless network with a reliable link layer and multiuser (variable rate) scheduling. In particular, we investigate the feasibility of a receiver-side enhancement to TCP to improve the throughput and fairness in this scenario.

A precursor to the CLAMP protocol was proposed in [42], and it was later implemented in a General Packet Radio Service (GPRS) performance-enhancing proxy with promising results, as reported in [43]. The performance of CLAMP over a single time-varying channel was studied in [44]. This paper extends these results to multiuser wireless scheduling with a more realistic model of wireless fading and link-rate adaptation. Other papers using the *AWND* to control the sender include Freeze-TCP [45] (for wireless), [3], and [4].

3 System Topology and the CLAMP Algorithm

In this section, we describe the system topology that we study in this paper, and we motivate and introduce a window flow-control algorithm for this scenario that, in contrast to TCP, does not react to loss and does not lead to wide fluctuations in the window.

3.1 Topology and Notation

The access network topology of interest is illustrated in Fig. 1. In the example scenarios studied in this paper, the AP maintains a separate queue for each user to allow channel-state-aware scheduling between users, but users may be involved in multiple TCP sessions, which then share the same queue, as in the case for user 1 in the figure. Alternatively, the AP may use only a single queue for all users (not depicted but is relevant for wireline applications and for many existing wireless networks such as 802.11a and b).

The service rate of a queue, that is, μ_c bytes/s, depends on the channel statistics of the users and the scheduling policy at the wireless AP. Referring to Fig. 1, each flow i has a source node S_i and an associated round-trip transmission delay, which includes all propagation and queuing delays except for the queuing delay at the access router.

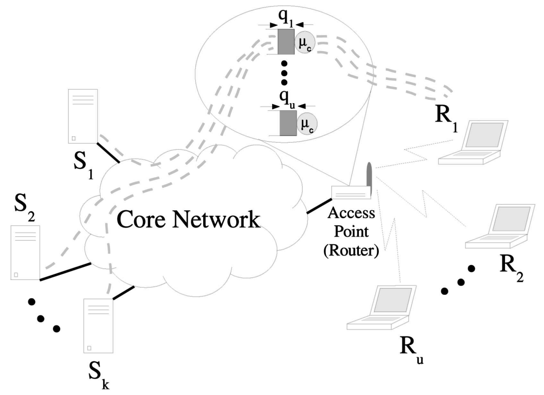


Fig. 1. System topology.

Each sending node implements a sliding-window flow control. Under the assumption that sources are greedy, the total number of packets and ACKs for flow i, which is in flight at any time t, is equal to the window size $w_i(t)$. The CLAMP algorithm selects $w_i(t)$ in a decentralized way such that each flow sharing the same queue obtains a proportional share of the service rate μ_c, and the equilibrium buffer occupancy of the queue $q(t)$ can be controlled, as described in the following.

3.2 Controlling the Window: A Fluid Model

Our aim is to introduce a model of the queuing at the AP, as well as a window control algorithm that responds to the queue size at the AP rather than buffer overflow events. The algorithm that we propose has a similar objective to TCP Vegas: to try and store a small number of bytes per flow at the bottleneck, for example, τ bytes per flow. The aim is to avoid interaction between the layer 4 flow control and the layer 2 wireless scheduler.

Let us focus for now on a single queue and assume that it receives a deterministic service rate of μ_c bytes/s. In a wireless scenario, there could be any number of queues (u queues depicted in Fig. 1; here, we are focusing on just one of them and assume that the scheduler offers it a rate μ_c bytes/s when it is backlogged). Assume that there is a single flow and it has a propagation delay of d seconds, so its bandwidth-delay product is $\mu_c d$ bytes, excluding the queuing delay at the AP. An alternative to TCP congestion avoidance would be the following control, which we express in the form of a differential equation:

$$\begin{aligned} \frac{dw}{dt} &= \frac{1}{d}(\tau - bq(t)), \\ q(t) &= \begin{cases} 0 & \text{if } w(t) < \mu_c d \\ w(t) - \mu_c d & \text{otherwise,} \end{cases} \end{aligned} \tag{1}$$

where b is a dimensionless constant and $q(t)$ is the amount of the flow (in bytes) stored at the AP (a continuous-valued quantity since this is a fluid model).

The algorithm (1) will increase $w(t)$ until $q(t)$ is nonzero (in Section 4, we will introduce a slow-start mechanism to speed up this initial increase). When $q(t) > 0$, the link output rate is a constant μ_c bytes/s. Since we are assuming a window flow control (which implies that self clocking occurs at the sender side), the input rate to the queue must also be μ_c bytes/s plus the rate of change of the window (positive or negative) that occurred d seconds earlier. When $q(t) > 0$, the only effect of the changing window size is the change in the amount of fluid $q(t)$ buffered in the queue: It will not affect the rate μ_c allocated by the scheduler. The point of the flow control is to keep the "pipe" full and maintain a target of τ bytes in the queue to avoid any interaction with the scheduler.

In this paper, we like to think of the control as occurring at the receiver, so $w(t)$ is the amount of flow outstanding ("in the pipe") at time t from the receiver's point of view. Thus, the input rate to the queue at time t is $\mu_c + \frac{dw}{dt}|_{t-d}$, where the time lag d reflects the time that it takes for the change in the window to be seen in the arrival rate at the queue. Finally, since the rate of change of the queue is the difference between the input and output rates, it follows that the queue size obeys the delayed differential equation

$$\frac{dq}{dt} = \frac{1}{d}(\tau - bq(t-d)) \tag{2}$$

during periods when the queue is nonzero.

The above model is highly simplified, but it contains the basic form of the flow control that we will define later. The assumption of a deterministic service rate μ_c is not realistic for wireless links, but we ignore this issue for now. The assumptions that the flow can be modeled by a fluid and that the window can vary in a continuous manner are clearly highly idealized, and we will relax all these assumptions in the precise definition of the CLAMP protocol in Section 4.

For now, let us motivate the above flow-control algorithm with the following observations: First, if $q(t)$ will converge, it will converge to the unique equilibrium

$$q(t) = \frac{\tau}{b}. \tag{3}$$

The question of stability is resolved in the following:

Theorem 1. *The necessary and sufficient condition for the total queue size $q(t)$ to converge under (1) is*

$$b < \frac{\pi}{2}. \tag{4}$$

Proof. Taking the unilateral Laplace transform of (2) gives

$$Q(s) = \frac{q(0^-)}{s + (b/d)e^{-ds}}. \tag{5}$$

The (infinite number of) poles of (2) determine the limiting behavior of (1). For $d = 0$, the system has a single real pole located at $s = -\infty$; hence, it is stable for all $b \geq 0$. For $b \geq 0$, as d is increased, an infinite number of poles will appear from infinity on the left half plane and ultimately cross the imaginary axis. Applying the method shown in [46, p. 26], we search for potential points where the poles cross the imaginary axis by solving

$$W(\omega^2) = \omega^2 - (b/d)^2 = 0$$

for ω^2. The solution $\omega^2 = (b/d)^2$ indicates that the poles cross the imaginary axis at $s = \pm j(b/d)$. Equating the characteristic equation of (5) to zero and substituting in $\omega = b/d$ reveals that the first time that two poles cross into the positive half of the imaginary axis occurs when $d = (\pi d)/(2b)$; that is, the system will be stable for $b < \pi/2$. Note that, if (2) is stable, then the asymptotic stability of (1) follows since $q(t)$ will not drop to zero if we start sufficiently close to equilibrium. □

Note from (1) that b is a "reactivity" parameter, which determines how quickly the control responds to the measured queue size. If it is too large, then the system is unstable, but if it is too small, then by (3), the queue size in equilibrium can be very large. We conclude that $b = 1$ is a safe choice. In this case, the system is stable and the equilibrium queue size is τ bytes.

One oversimplification of this model is that it assumes per-flow queuing, whereas, in fact, several flows may share the same queue, as in the case for queue 1 in Fig. 1. For example, in our simple wireless scenario, several flows may be destined for the same mobile device, but the scheduler may not be able to distinguish these flows. In more complex multihop network scenarios, per-receiver queuing may be more preferable than per flow queuing, so it is important to allow this possibility. In this case, our flow-control algorithm must provide fair allocation to the flows sharing a queue, as the scheduler is unable to distinguish them.

Let us modify the basic flow-control equation (1) to allow more than one flow to share the queue. In the following, we write the window evolution equation for one flow i that shares the queue:

$$\frac{dw_i}{dt} = \frac{1}{d_i}(\tau - p(q(t))\mu_i(t)), \tag{6}$$

where we have replaced bq/μ_c with the function

$$p(q) = \frac{bq - a}{\mu_c} \tag{7}$$

and μ_c with the rate $\mu_i(t)$ achieved by flow i at time t. In (7), the parameter a is a constant, in units of bytes, that allows more flexibility in controlling the equilibrium queue size, as will be seen. Note that τ can be interpreted as an additive increase term, and $p(q(t))\mu_i(t)$ can be interpreted as a multiplicative decrease term given that the decrease is proportional to the rate achieved by the flow.

This window update rule incorporates two important principles of fair, stable flow control. It is well known [47] that a simple "additive increase, multiplicative decrease" (AIMD) window update rule produces fair bandwidth allocation. This suggests that the rate at which the window is updated should be the sum of a positive constant and a negative term proportional to the flow's rate. It has also been recently shown [48] that the update rate should be inversely proportional to the flow's RTT in order for the flow-control system to be stable. The algorithm above incorporates both of these principles, and fairness and stability are indeed achieved, as we show in this paper.

Using k to denote the number of flows sharing the queue and applying the same reasoning as for the single-flow system, we obtain the delayed differential equation for the queue size:

$$\frac{dq}{dt} = \sum_{i=1}^{k} \frac{dw_i}{dt}\bigg|_{t-d_i} \tag{8}$$

$$= \sum_{i=1}^{k} \frac{1}{d_i}\left(\tau - (bq(t-d_i) - a)\frac{\mu_i}{\mu_c}\right). \tag{9}$$

To get insight into the behavior of this system of equations, note first that, since we are assuming that the queue does not empty (this is a stability issue that we address later), the following flow conservation law must hold:

$$\sum_{i=1}^{k} \mu_i(t) = \mu_c. \tag{10}$$

If the system is stable, that is, $q(t)$ converges to a constant q^*, and each window size $w_i(t)$ converges to a corresponding constant w_i^*, then it follows from (6) that, at equilibrium, $\mu_i(t) = \tau/p(q^*)$. Since this is a constant, which is independent of i, it follows from (10) that $\mu_i(t)$ must converge to μ_c/k. Thus, the system is fair in the sense that it allocates an equal fraction of the total service rate to each flow. Assuming that the system is at equilibrium, we obtain from (6) and (7) that

$$q^* = \frac{k\tau + a}{b}. \tag{11}$$

The question of stability is resolved in the case of equal delays (that is, $d_i = d$ for all i) by noticing that, in this case, (9) reduces to the simpler form

$$\frac{dq}{dt} = \frac{1}{d}(k\tau + a - bq(t-d)) \tag{12}$$

(using (10)). This becomes identical to (2) if we replace the constant $k\tau + a$ with τ. Since the value of this constant did not enter the stability analysis, Theorem 1 continues to apply, and we obtain the stability condition that $b < \frac{\pi}{2}$.

We do not have a complete analysis of stability in the case of unequal delays, but the numerical evidence that we have obtained is that the system is stable when $b < \frac{\pi}{2}$. Preliminary analysis in the case of unequal delays can be found in [49]. The results that we present in this paper, which are for the CLAMP protocol (as defined in Section 4), also support the hypothesis that the underlying equations are stable.

Stability justifies our assumption that the output rate of the queue is fixed at μ_c bytes/s. Provided that we start the system sufficiently close to equilibrium, the scheduler will always see a backlogged queue and be able to allocate it a constant rate of μ_c bytes/s. Although the output rate is constant, its *composition* in terms of the individual flow rates is time varying and converges to an equal share of the output rate.

One can interpret the value $p(q)$ in (7) as a *congestion price signal* that is used by the flow-control equations (6) to adjust the window size. Note that it will be the AP that signals this price to the mobiles, and in this context, the AP only needs to know $q(t)$ and the total rate μ_c (both of which can be measured after a suitable averaging to smooth out fluctuations), as well the constants a and b.

In the protocol implementation of CLAMP in Section 4, τ will be a parameter that is used in the mobile receiver to calculate the value of *AWND*, whereas a is a parameter that is used in the AP to compute a congestion price signal. We will use a fixed value of $\tau = 500$ bytes in all scenarios in this present paper. The parameter a provides a tunable parameter to the AP to help target a suitable equilibrium queue size. We will illustrate the effectiveness of this control knob in Section 5.5, as well as why it is particularly useful at a wireless AP. The viewpoint here is that a is free to be chosen by the AP, but it is a fixed parameter for the system once that choice has been made (it is possible that it can be varied on a slow timescale in response to changing network conditions, but we have not yet investigated this approach). Finally, as noted earlier, $b = 1$ is a safe choice, and this is one that we make for the remainder of this paper.

A disadvantage of the proposed method of control is that the equilibrium queue size q^* depends on the number of flows k. Indeed, there is a linear growth with respect to k, as given in (11), and the constant τ provides the rate of growth with k. It is an open research problem to devise a control signal that allows the AP to have complete control over the equilibrium queue size without suffering this growth with the number of flows that share the same queue. In the context of this present paper, that is, a single hop wireless network from AP to mobiles, this is not a major problem. There will not usually be many flows in the same receiver queue, and in any case, the AP can use the parameter a to offset the effect of $k\tau$. For example, if a single flow requires significant queuing (to average out the channel fluctuations), then the value of a may dominate the value of $k\tau$ for small values of k.

We propose to implement (6) in the mobile receivers, and this requires them to measure the rate in bytes per second (μ_i for flow i) that they are receiving and also that they receive the congestion price signal $p(q(t))$ from the AP. In a wireless scenario, it will be necessary to perform some averaging on the measured received rate to remove statistical fluctuations that are not accounted for above. It also appears that the mobile needs a precise measurement of each flow's propagation delay, which might not be known at the receiver. However, we remark that the normalization by d_i in (6) is only a gain parameter, which is needed to ensure stability, and its precise value does not affect the equilibrium achieved. In practice, one can replace d_i by the estimate $\frac{w_i(t)}{\mu_i(t)}$ (in seconds), and this is what we use in the definition of the CLAMP protocol in Section 4.

All bets are off in the more realistic case that the scheduler offers a random and time-varying rate to each queue. In this case, one cannot assume that the output rates are constant nor that the queues will not empty. It is then hard to avoid an interaction between the layer 4 flow control and the layer 2 scheduling. Unfortunately, realistic scenarios are hard to treat analytically, and for this reason, we resort to simulation experiments in Section 5.

4 The CLAMP Protocol

In Section 5, we will implement the window clamping algorithm as a receiver-side modification to TCP, operating over a simulated wireless network. To do so, we need to recast the algorithm in terms of discrete packets, which arrive at possibly random time instants and are controlled at the sender by TCP New Reno. In order to implement CLAMP, the access network should be modified by inserting a software agent in both the AP and the mobile client, as described in the following.

4.1 Access Router Agent

The software agent in the access router samples each user's queue length and computes an exponential moving average q (a separate value for each queue). Precisely, when the nth packet arrives in the queue, the moving average $q(n)$ is computed via

$$q(n) = \exp(-2I)q(n-1) + (1 - \exp(-2I))Q(n),$$

where I is the time between packet arrivals (in seconds) at the AP buffer and $Q(n)$ is the current queue size at packet arrival n.

Given q, it evaluates the following price:

$$p_s(q) = \max((q - a)/\mu_c, 0), \tag{13}$$

which is the same as that described in (7), aside from an explicit nonnegativity constraint to avoid the possibility of a "multiplicative increase." This nonnegativity constraint has no effect on the equilibrium calculated previously: The parameter a (in bytes) still controls the equilibrium mean queue size q^* exactly, as it did in Section 3.2. The value μ_c is the measured output rate from the queue. Each packet that is sent out carries a field containing $p(q)$, where q is the current value of the moving average when the packet is sent.

4.2 Client-Side Agent: Window Update Algorithm

The client agent's function is to receive the router's congestion measure p and set the receiver's TCP *AWND* value according to the algorithm described below, hence clamping the sender's TCP transmission window.

For simplicity, the algorithm will be described for a single flow. Let t_k denote the time instant when the kth packet, with a size of s_k bytes, is received by the receiving client. CLAMP calculates the change in window size (in bytes [50]) as

$$\Delta w(t_k) = \left[\frac{\tau - p(q(t_k))\hat{\mu}(t_k)}{\hat{d}(t_k)}\right](t_k - t_{k-1}), \tag{14}$$

where $\hat{\mu}$ is an estimate of the average received rate of the flow in bytes per second, $\hat{d}(t_k)$ is given in the following, and $\tau > 0$ (in bytes) is a constant for all flows. Note that this is a discrete approximation to the fluid equation (6).

The factor in brackets is the rate of update of the window. The factor $\hat{d}(t_k)$ is a coarse approximation of the RTT, which ensures that the control loop has a gain with the appropriate scaling behavior. In Section 3.2, we showed that the fluid equations were stable with this scaling of the gain. In [49], we showed that, without this scaling, the equations can be unstable in some circumstances.

The current received rate $\hat{\mu}$ is estimated by using an exponential moving average:

$$\hat{\mu}(t_k) = \frac{\sum_{i=k-\alpha}^{k} s_i}{t_k - t_{k-\alpha}} \tag{15}$$

and

$$\hat{d}(t_k) = (1 - \beta)\hat{d}(t_{k-1}) + \beta\frac{\mathrm{AWND}(t_k)}{\hat{\mu}(t_k)}, \tag{16}$$

where the integer α and $\beta \in (0, 1)$ are smoothing factors, and $\mathrm{AWND}(t_k)$ is the actual AWND (see below). $\hat{d}$ is initialized by the time between the SYN-ACK being sent and the first data packet being received. In the experiments in this paper, we use the following value:

$$\alpha = \min\{\text{number of packets received so far}, 1{,}000\}.$$

Thus, initially, $\hat{\mu}(t_1) = 0$, $\hat{\mu}(t_2) = (s_1 + s_2)/(t_2 - t_1)\ldots$, and so forth. Finally, we use the value $\beta = 0.001$ in the update of $\hat{d}$.

In equilibrium, $\hat{d}$ will be the RTT of the flow in seconds, that is, propagation plus queuing delay. However, computing $\hat{d}(t_k)$ clearly does not require an explicit measurement of such a delay, which is an important consideration, given that the algorithm must operate during the transient period. Before updating the window size w, Δw is clipped to a maximum value of 10 Kbytes to avoid sudden increases after idle periods. It is also clipped below: The window cannot decrease by more than the number of bytes in the received packet to ensure that the right edge of the window is monotonic nondecreasing [50, p. 42]. Note that, for notational ease, we have omitted the index of the flow in the above notation. However, each flow will maintain its own w, $\hat{\mu}$, and $\hat{d}$.

The *AWND* value advertised to the sender is then

$$\mathrm{AWND}(t_k) \equiv \min(w(t_k), \mathrm{AWND}), \tag{17}$$

where the *AWND* on the right-hand side of the assignment is the value received by the client-side agent from the receiver's operating system. The assigned value on the left-hand side is the one used in (16).

4.3 Receiver-Side Slow-Start

It is important for CLAMP to allow TCP slow-start to take place at the start of a connection to allow TCP to fill the network pipe. We propose that CLAMP should mimic TCP slow-start at the beginning of a connection so as to find a reasonable estimate of the bandwidth-delay product. TCP starts with a window size of 1, and during its "slow-start" phase, it increases its window by one segment for each ACK received, doubling the window every RTT. Slow-start terminates when the buffer overflow causes a packet loss, and the window size is then halved.

To track this, a new CLAMP connection starts with $w(0) = MSS$, where MSS is the connection's maximum segment size, and for each packet received, it sets

$$w(t_k) \leftarrow w(t_k) + MSS$$

instead of using (14). CLAMP stops tracking slow-start when $\Delta w(t_k) < 0$ in (14), or three duplicate ACKs are generated, indicating packet loss. The receiver then sets $w(t_{k+1}) \leftarrow w(t_k)/2 \approx w(t_k - \mathrm{RTT})$. The reason for this is that the observations at time t_k, which indicate congestion, reflect the *AWND* size one RTT earlier. This terminates CLAMP's slow-start in a way very similar to the sender's TCP slow-start. The only difference is that the condition

$\Delta w(t_k) < 0$ can terminate slow-start in addition to duplicate ACKs. This allows the buffer to be much larger at the AP without filling it up during slow-start.

CLAMP does not use the AP buffer to control the flow rates, so the CLAMP buffer can be much larger than with TCP New Reno. In fact, in the results in this paper, we use a large buffer size of 6 Mbytes for CLAMP to avoid any chance of buffer overflow for the CLAMP protocol. Thus, we require the CLAMP agent to terminate slow-start at the natural point when the pipe is full (which would occur naturally with a buffer overflow if the buffer was a smaller value).

It is necessary to place an upper limit on the growth of the receiver window during slow-start to counter the possibility that the sender-side slow-start terminates prior to the pipe filling up (due to a small value of the sender-side threshold $ssthresh$). Therefore, in practice, a receiver-side threshold would also need to be used. In the present paper, we assume that the sender and receiver thresholds are larger than the bandwidth-delay product of our relatively low-rate flows, and hence, we do not consider the issue in our simulation.

The CLAMP slow-start is only invoked at the start of a TCP connection to get an initial estimate of the bandwidth-delay product. If a time-out occurs, TCP will revert to slow-start, but CLAMP will just continue with CLAMP "congestion avoidance," that is, (14), with the view that it has a reasonable estimate of the bandwidth-delay product. In other words, handling time-outs is left to TCP.

5 Performance Evaluation of CLAMP through Simulation

In the following sections, we investigate the performance of TCP New Reno over the Internet for a scenario in which all flows terminate in devices in the same network, all receiving information from a common AP. In these experiments, the last-hop network is assumed to be the bandwidth bottleneck. This assumption is reasonable for many cellular networks, where the base station is connected to the high-speed network core. In some scenarios, a flow may be bottlenecked elsewhere in the Internet: Congestion control is then provided by TCP New Reno [51].

The CLAMP protocol and TCP New Reno are both too complex to treat analytically, so we have developed an event-driven simulator to model the physical, link, and transport layers. Our implementation of TCP follows TCP New Reno [52], including the so-called "bugfix." We study various network scenarios and compare the performance of TCP New Reno with and without the CLAMP protocol at the receiver end of the connections.

With regard to CLAMP, it is no longer clear that it will be able to provide the fairness promised in the fluid analysis of Section 3.2. Now, the protocol has to cope with random and time-varying service rates, wireless packet loss, and the fact that it is limited to only the receiver-side control. TCP New Reno is also operating at the sender-side and can potentially take control. A major focus is to see how much of the fluid analysis carries over to this more realistic scenario: Can CLAMP reduce the interactions between the layer 4 flow control and the layer 2 scheduling?

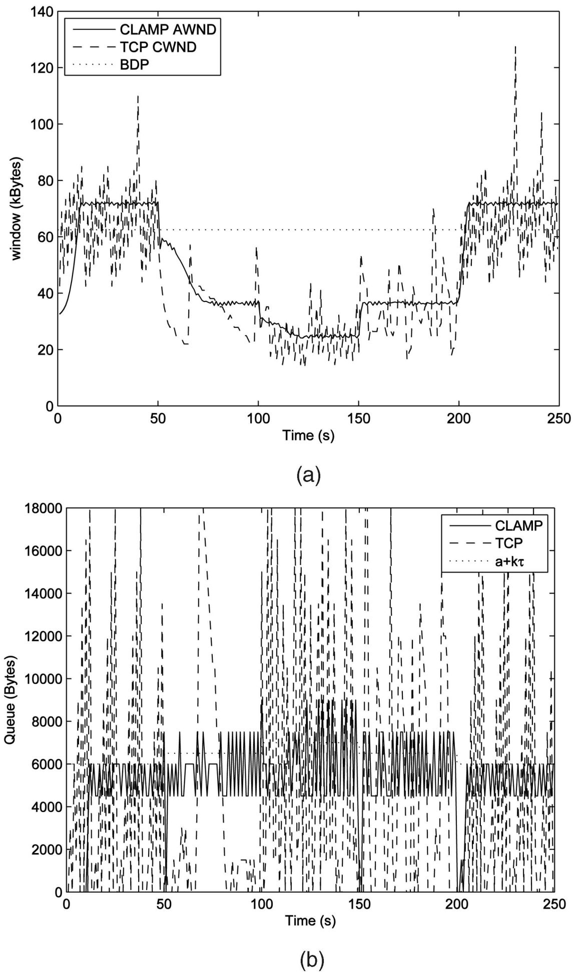


Fig. 2. Deterministic link cross traffic from 50 to 200 seconds and from 100 to 150 seconds. (a) Window dynamics. (c) Queue dynamics.

5.1 Window Dynamics for a Deterministic Link

Before considering random channels, let us consider a static channel with dynamic traffic. Section 3.2 provides a mathematical fluid model to describe the equilibrium behavior of CLAMP, but in this section, we illustrate the behavior in a more dynamic environment, in which flows come and go and the flow-control algorithm must be able to adapt to the fluctuations in the system load.

In the first experiments, we consider a deterministic link of 10 megabits per second (Mbps) that is shared by a number of receivers. A deterministic link could model a wireline network with a number of users connected to a central hub. Alternatively, it could be a wireless network with sufficient diversity (frequency, time, or space) that can be treated as essentially error-free and with a constant rate. By presenting results for a constant bit rate link, we are able to see more clearly the dynamics of the CLAMP window flow-control algorithm and contrast it with that of TCP New Reno (without CLAMP at the receiver).

In Fig. 2a, we plot the dynamics of flow 1 starting at time 0 and terminating at time 250 seconds. This flow initially gets all the bandwidth, which is 10 Mbps, but at time 50 seconds, it is joined by flow 2, and at time 100 seconds, by flow 3. Flow 3 leaves at time 150 seconds, and flow 2 leaves at time 200 seconds. All flows have a propagation delay of 50 ms and share the same queue.

There are two experiments for this scenario, both with TCP New Reno at the senders: One with CLAMP switched on and the other with CLAMP switched off. Since there is no wireless loss, and CLAMP prevents congestion loss, the *CWND* will exceed CLAMP's *AWND* and need not be plotted. The plotted TCP (*CWND*) is the *CWND* from the experiment in which CLAMP is not used. The buffer at the AP has a size of 19,500 bytes, a is 5,500 bytes, and τ is 500 bytes. With these values, the average queuing delays in the two experiments are approximately equal.

The window and queue dynamics from both experiments are depicted in Figs. 2a and 2b, respectively. It is clear from the window evolution that CLAMP tracks the dynamics of flow arrivals and departures in a way similar to TCP New Reno but with much fewer fast timescale fluctuations. Note that the queue for CLAMP tracks the queue size predicted by the fluid model quite closely in terms of the given parameters a, τ, and k, the number of flows.

The window evolution depicted in Fig. 2a can be interpreted as a rate-versus-time plot as well due to the fact that the queuing delay is small in this experiment. The figure shows how CLAMP reacts to the changes of load, decreasing the rate in proportion to the number of flows with the queue size increasing (slightly) with the number of flows, as predicted in Section 3.2. TCP's queuing fluctuations are much greater, but it also achieves a fair rate allocation between the flows, which is an example of the fairness of TCP when the propagation delays are the same.

Fig. 2a contrasts the window dynamics of CLAMP with that of TCP. The TCP dynamics are the well-known additive increase and multiplicative decrease in the window size. The additive increase occurs once per RTT, and the multiplicative decrease occurs on the first packet loss within an RTT. With CLAMP, there is an additive increase term τ, but the multiplicative decrease term is multiplicative in the rate, not in the window size, and both terms operate simultaneously and on every packet, thus getting averaged over an RTT. The net effect is a window size that varies more smoothly, as illustrated in this example, and the aim is to avoid buffers from emptying unnecessarily.

5.2 Wireless Channel Model and Link Layer

We now consider performance when the last-hop network is time varying. The main scenario that we have in mind is a cellular network in which the receivers are highly mobile and the base station plays the role of the AP. Our event-driven simulator requires a different layer 1 and 2 model for the time-varying scenarios. The model used in these simulations aims to be as simple as possible while describing the variable rate transmission characteristic of optimal wireless links and providing a reasonable model for packet loss.

The multipath fading process between the base station and each receiver is a stationary stochastic process according to Jakes' model, with $G = 20$ generators and Doppler frequencies of either 0.6 or 60 Hz (a 1.8 GHz carrier and mobile speeds of either 0.1 or 10 m/s, respectively). That gives a signal-to-noise ratio (SNR) at time t of

$$N(t) = SNR_{ave}\left|\sqrt{\frac{2}{G}}\sum_{n=1}^{G}\cos(\omega_n t)\exp\left[\frac{j\pi n}{G}\right]\right|^2,$$

where we take $SNR_{ave} = 4(= 6\text{ dB})$ as the mean SNR, $\omega_n = 2\pi f_d \cos\left(\frac{\pi}{2G}(n+0.5)\right)$ is the angular frequency of the nth path, and f_d is the Doppler frequency. We will use the above model for all mobiles in our experiments, using the same average SNR. Thus, large-scale propagation effects are not explicitly modeled.

Frame transmissions over the physical channel occur in slots of $\tau_{\text{slot}} = 3$ ms. A slot is the time period of one transmission attempt. A single 3 ms slot can contain data from multiple packets, and packets can be split between slots. The coding rate for the slot is calculated at the start of each slot, and $B(s)$ information bits are transmitted, where

$$B(s) = \tau_{\text{slot}} \min(W_0 \log(1 + (1-m)N(s\tau_{\text{slot}})), C_{\max}), \qquad (18)$$

where $C_{\max} = 10$ Mbps is the maximum possible rate that can be transmitted over the channel, $W_0 = 4.3$ MHz models the bandwidth, and m is a "backoff margin," which is adaptively tuned such that the packet error rate after retransmissions is approximately a target nominal packet error rate, as specified in the simulation.

In order to capture the bursty nature of errors, we use a simple error model: If the SNR at the end of the slot, $N((s+1)\tau_{\text{slot}})$, is such that

$$B(s) > \tau_{\text{slot}} W_0 \log(1 + N((s+1)\tau_{\text{slot}})),$$

then an error is declared. This will occur if the channel deteriorates sufficiently during the slot so that it can no longer support the selected rate. If the SNR is sufficiently high at both the start and end of a slot, then it is assumed to be sufficiently high throughout the slot. The backoff margin m is included to allow some degradation to occur without causing an error.

If n consecutive attempts to a given destination fail, then all packets that would have been completed by that transmission are dropped. We use different values of n for fast and slow fading, as discussed in the experiments. Packets that are started in the initial 3 ms slot but would not be completed are not dropped but buffered to reattempt in the next slot.

We will see later that this approach to link-layer coding results in higher throughput over slowly varying channels, as compared to fast fading channels. This is due to the fact that we are exploiting channel-state information at the transmitter and that information is less reliable in a fast fading environment, requiring a larger backoff margin.

5.3 The Scheduler

The AP maintains separate queues for each receiver (mobile device) and acts as an Internet Protocol (IP) router, routing packets that are destined for the mobiles into the appropriate queue based on the address of the mobile. Thus, all TCP flows destined for the same mobile are put into the same output queue.

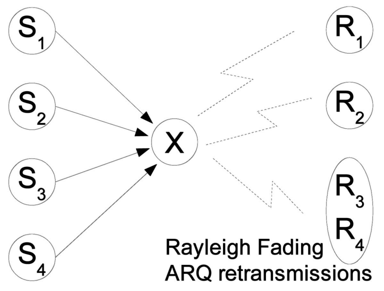


Fig. 3. Simulation network topology.

The scheduler is based on the principle of multiuser diversity. At each slot, a new scheduling decision is made, and the general preference is for scheduling a mobile that is in a good channel state. However, fairness must also be considered: Some users may be in bad locations, thus getting much lower rates, if only channel quality is considered.

In this study, we implement a scheduler along the lines of the proportionally fair scheduler [53] used in Qualcomm's high data rate (HDR). This scheduler gives proportional fairness when users have arbitrary average SNRs. Since we assume that the average SNRs are the same, it gives equal rates to each mobile. At each slot, the scheduler picks the queue i with the largest utility that has data to send in this slot. The utility U_i is calculated as

$$U_i := \frac{\mu_i}{r_i}, \tag{19}$$

where μ_i is the current rate for the mobile i at the beginning of the slot and r_i is the exponential moving average of the rate obtained by mobile i with an averaging time constant of 100 ms. Precisely, r_i is obtained from

$$r_i(n) = 0.97 r_i(n-1) + 0.03\mu_i(n),$$

where n indexes time, which is in slots. In the fast fading scenarios (below), the coherence time of an individual user's channel is 16.67 ms, and we want the smoothing time constant for the rate achieved by the mobile to be significantly larger than this to allow the scheduler to allocate slots to good channels rather than in a round-robin fashion. In the slow fading scenario, the scheduler will act more like a weighted round robin with better channels getting higher weights. We chose a time constant of 100 ms to give the scheduler the chance to schedule based on the channel quality in the fast fading scenario yet be able to adapt to the changing dynamics of flow arrivals and departures.

5.4 Wireless Network Topology

Fig. 3 depicts the topology of the time-varying wireless simulation experiments of this section. Due to space limitations, we have focused on a specific scenario with a small number of long-lived TCP flows and a range of different propagation delays.

Sources S_1, S_2, S_3, and S_4 represent servers that are TCP rate controlled and the links from S_i to X model the entire paths through the Internet. Node X is the wireless access router that contains the CLAMP router software agent. The wireless access router transmits to the TCP/CLAMP clients $R_1 - R_4$ over a common broadcast radio channel. Clients R_3 and R_4 are both running in the same physical device and so have a common queue at router X and common radio propagation conditions, as described below. Clients R_1 and R_2 have separate queues, so in this experiment, there are three channels to be scheduled. All links are bidirectional, with the characteristics shown in Table 1, and the wireless bandwidth is $W_0 = 4.3$ MHz with a maximum bit rate of 10 Mbps. The packet loss rate is 10^{-3}, which is achieved via the fade margin and link-layer retransmissions. The number of retransmission attempts at the link layer is $n = 5$ in the slow fading scenario and $n = 15$ in the fast fading scenario. Other simulation parameters are given in Table 2.

TABLE 1
Link Configuration

Node 1	Node 2	Capacity	Delay	Pk loss rate
S_i	X	100 Mbit/s	$d_i/2$	0
X	R_i	see text	1 msec	10^{-3}

TABLE 2
Simulation Parameters

Parameter	Value
TCP Packet Size	1500 Bytes
CLAMP τ	500 Bytes
CLAMP b	1
CLAMP a	Variable
d_1	30 ms
d_2	130 ms
d_3	10 ms
d_4	180 ms

In the following experiments, we will plot the throughput (and other measures) against the average queuing delay, where the average is taken across all queues. For pure TCP Reno, the average queuing delay is controlled by the choice of buffer size at the AP: The queuing delay is increasing in the buffer size. For TCP Reno + CLAMP, the average queuing delay is controlled by the choice of the parameter a. It will be shown (see Fig. 9) that the queuing delay is an increasing function of the parameter a. By running simulations across a range of values of buffer size for TCP and a for CLAMP, we are able to generate the following graphs. In Section 5.9, we will discuss the problem of choosing the appropriate parameters both for CLAMP and for TCP. As discussed in Section 4.3, we use a large buffer size of 6 Mbytes for CLAMP to avoid any chance of buffer overflow for the CLAMP protocol.

5.5 Fair Resource Allocation

We are particularly interested in the fairness of resource allocation to the long-lived TCP flows since the congestion-avoidance part of TCP tries to reach long-term equilibrium throughputs for these flows. Fig. 4 plots the throughput of each flow against the mean queuing delay (averaged over all flows) both for CLAMP (Figs. 4a and 4c) and for pure TCP New Reno (Figs. 4b and 4d).

In the following, we will denote flows by their propagation delays. For a moderate average queuing delay (50 ms

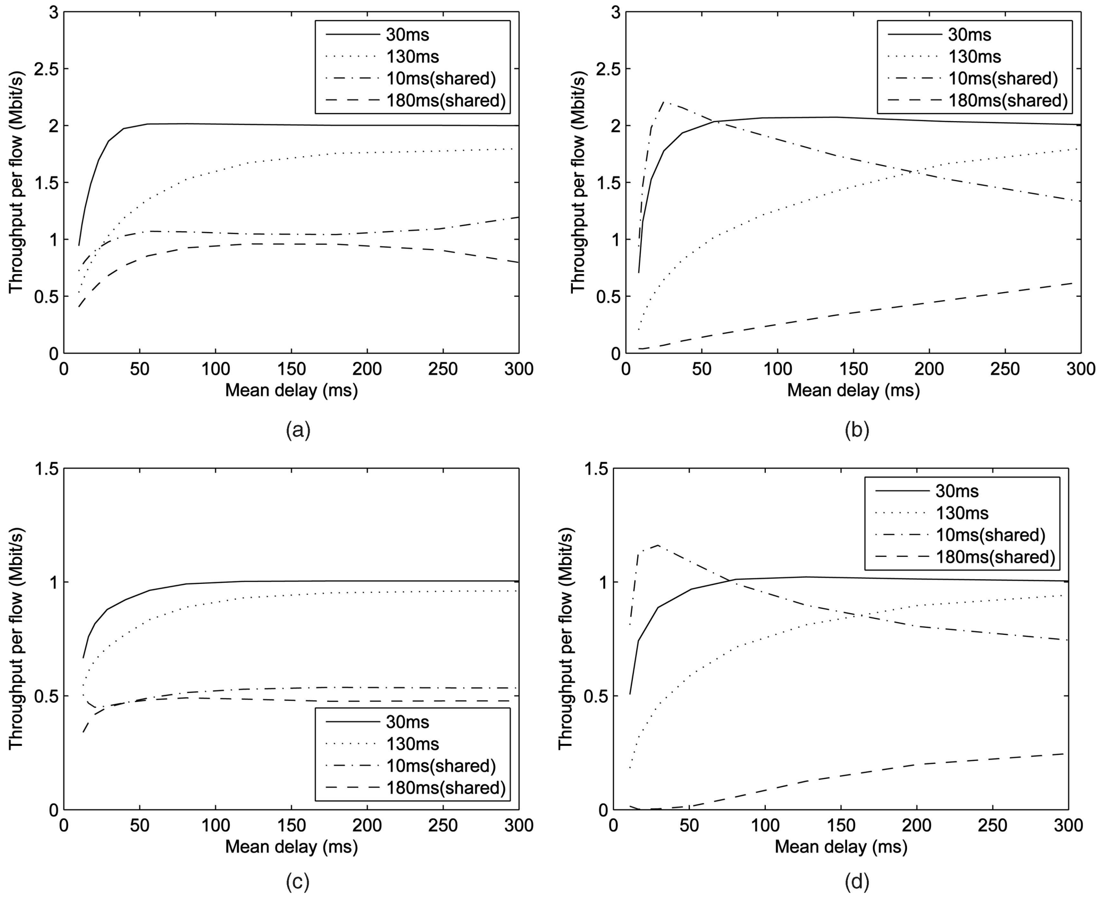


Fig. 4. Throughput per flow for CLAMP and TCP New Reno. Dotted-dashed lines are for flow 3, whereas dashed lines are for flow 4, both sharing a common queue at the AP. (a) CLAMP, with 0.1 m/s. (b) TCP New Reno, with 0.1 m/s. (c) CLAMP, with 10 m/s. (d) TCP New Reno, with 10 m/s.

or less), TCP New Reno is unfair to flows with large propagation delays. RTT unfairness exists, even for flows that do not share the same queue, such as the 30 ms and 130 ms flows, where the 30 ms flow has an unfair advantage. Ideally, the scheduler allocates equal rates to all queues. However, in both scenarios (slow and fast fading), the 30 ms flow receives more bandwidth than the 130 ms flow and significantly more bandwidth at "desirable" queuing delays such as 50 ms (or less).

It is well known that TCP (both Reno and New Reno) is unfair to flows with long propagation delays [6]. This assumes that all flows share the same queue, are served in the FCFS order and get the same channel rate when serviced. Certainly, the 10 ms and 180 ms flows share the same queue and are served in the FCFS order so the fact that the 180 ms flow is treated unfairly is well known, but the magnitude of the unfairness, as depicted in Figs. 4b and 4d, is striking. Our results here extend to flows that are in separate queues in spite of a scheduler that tries to allocate bandwidth fairly to the users (queues). The problem is that the scheduler is only fair if the queues are always backlogged and TCP New Reno is unable to satisfy this assumption. Thus, the propagation delays have an impact on rate allocation in the TCP New Reno experiments.

The fairness situation is quite different for CLAMP. Section 3.2 predicted that CLAMP will keep the queues full and share rates fairly between flows that share the same queue. In the high-speed mobility scenario depicted in Fig. 4c, this is shown to be approximately true. All three queues achieve approximately equal rates, and the 10 ms and 180 ms flows, which share the same queue, also get approximately equal rates.

There is a discrepancy in that there is *less* fair sharing between the 10 ms and 180 ms flows as the queuing delay increases, as shown in Fig. 4c, which is not what the fluid model would predict. We investigated this and found that the 10 ms flow is experiencing more time-outs than the other flows and, significantly, more between 25 and 100 ms of queuing delay. The 10 ms flow experiences more variability in its RTT, hence the larger frequency of time-outs for this flow. As the queuing delay increases beyond 100 ms, the frequency of time-outs for the 10 ms flow decreases and its rate improves. The "fairness" depicted at about 30 ms is coincidental: The time-outs have brought the rate for the 10 ms flow down to that of the 180 ms flow, but were it not for the time-outs, the 10 ms rate would be higher and the 180 ms rate would be lower.

CLAMP is less fair in the low-speed mobility scenario, as shown in Fig. 4a, but is still much fairer than TCP New Reno. At an average queuing delay of 50 ms, the 30 ms flow gets a higher rate than the 130 ms flow, and the shared flows get a combined rate between these two rates. Within the shared queue, the 10 ms flow has a small advantage over the 180 ms flow. At queuing delays larger than 250 ms, the 10 ms flow gets a higher rate than at lower delays. This is because CLAMP starts losing control at a high queuing delay (see Section 5.7 for an explanation and illustration).

We might expect that the fast fading scenario is more like the "fluid model" in that there is more averaging. The scheduler averages rates over a period of 100 ms, and in the fast fading scenario, the channel fluctuations are much faster than this: The channel coherence time is 16.67 ms. Thus, we might expect fairness to be achieved when the average queuing delay is significantly larger than the coherence time, for example, at 100 ms of queuing delay. In this case, the queues rarely empty, and the rates achieved per flow are insensitive to the propagation delays.

In the slow fading scenario, the coherence time is significantly longer than the averaging period of the scheduler. In this case, the channel fluctuations do not average out over the queuing delay, and the queue can drain of packets. The larger the propagation delay of a flow, the more this affects the rate allocated to the flow. This effect reduces as the queuing delay increases. Once the queuing delay exceeds the maximum propagation delay of any flow sharing the queue, the queue will not drain, due to self clocking.

This analysis predicts that CLAMP will avoid interacting with the scheduler, provided that the channel fluctuations are much faster than the average queuing delay. The scheduler can then allocate capacity independently of propagation delays. This hypothesis is supported by the above simulation results.

Fast fluctuations happen automatically in high-speed mobility scenarios due to multipath fading, as considered here. However, it is also possible to induce fast fluctuations (at a controlled rate), even when the mobility is slow (or nonexistent), by using the technique of opportunistic beamforming [54]. CLAMP may be fruitfully combined with such techniques to provide fair allocation in a wireless network.

5.6 Total Throughput and Mean Time to Download

Fig. 4 reports on the individual rates per user to illustrate the fairness achieved by the two protocols. In Fig. 5a, the sum rate across flows is depicted. Together, these figures indicate that CLAMP's benefit is of providing fairness rather than of increasing the total throughput: There is not much between the two protocols in terms of the total throughput in the scenario investigated here. The very similar throughput-delay trade-off shows that both protocols are able to exploit multiuser diversity: As the queuing delay increases, there is more multiuser diversity.

Throughput, in bits per second, is only one measure of performance. Since CLAMP allows the scheduler to allocate rates more fairly between flows, we might expect that the average time to download a file of, say, 1 Mbyte, is less under CLAMP than under pure TCP, where the average time to download is dominated by the low rate flows. In Fig. 5b, the average time to download a file of size 1 Mbyte is plotted against the mean queuing delay for both TCP New Reno and CLAMP. As can be seen, at small to moderate mean queuing delays, CLAMP provides a very significant gain with respect to this measure of performance. For example, at 50 ms of average queuing delay, and

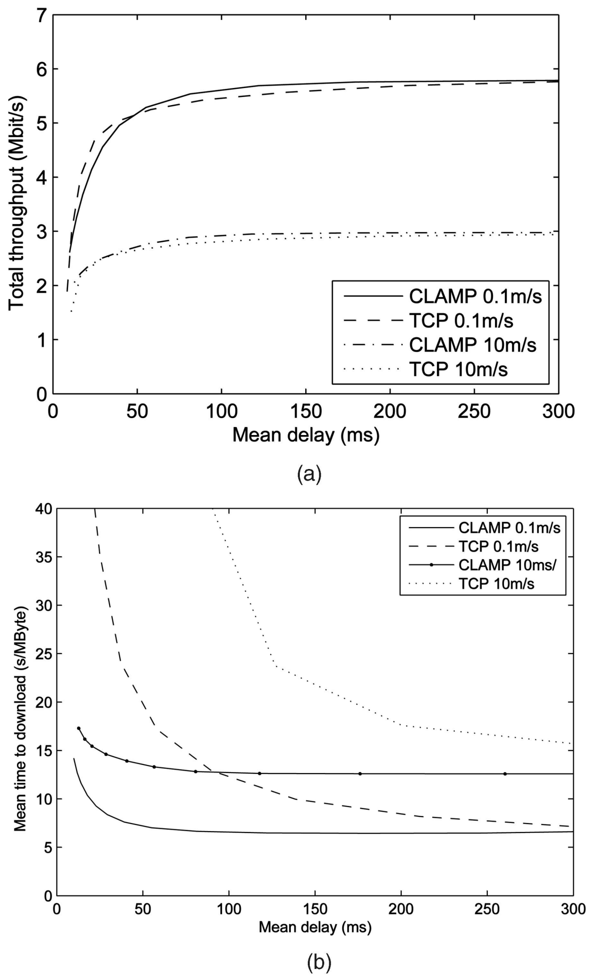


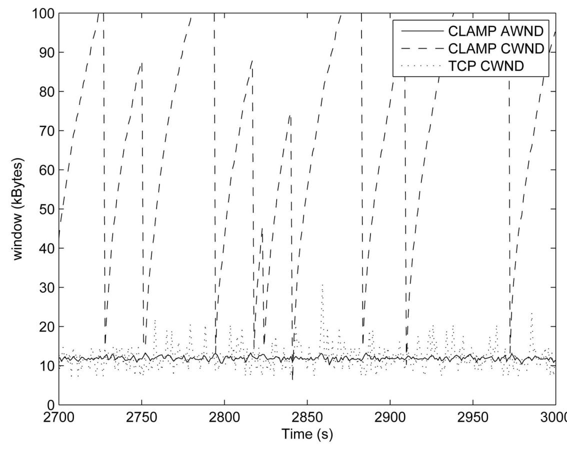
Fig. 5. Performance comparison: CLAMP versus TCP. (a) Total throughput versus delay. (b) Time to download 1 Mbyte versus delay.

at the mobile speed of 0.1 m/s, TCP files take, on the average, more than twice as long to download as they do under CLAMP. The average time to download is a better metric than throughput in capturing how users perceive performance since users experience latency, not bit rate.

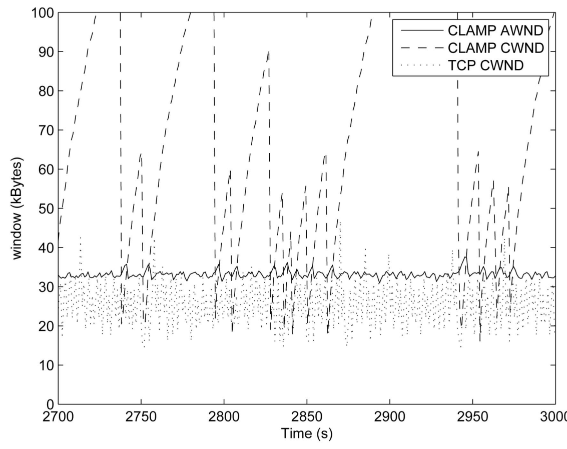
5.7 Window and Queue Dynamics

In this section, we plot the evolution of the window sizes over time for one of the flows in the above experiment. Fig. 6 plots the window dynamics for the 30 ms flow. Space considerations prohibit plotting the other flows, but qualitatively, they are similar. We illustrate with 300 seconds of window evolution, starting after time 2,700 seconds. This is, of course, much longer than a typical file transfer, but we ran this experiment for a long time in order to take the averages needed in Figs. 4 and 5. Also, after 2,700 seconds, the fade margins have long since converged, and we see the equilibrium behavior in Fig. 6. In practice, these margins may be known by the AP at the start of a flow, but the simulator uses an adaptive algorithm to find the fade margin.

We have chosen two different parameter settings for TCP and CLAMP, which correspond to the average queuing delays of 50 and 180 ms, respectively. Also plotted is the TCP CWND for the CLAMP experiment. For a queuing

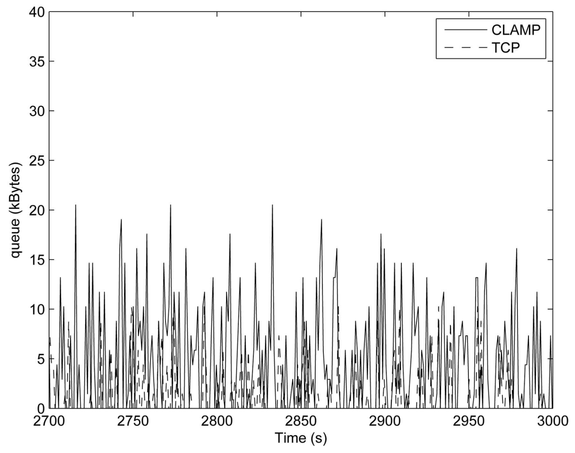


(a)

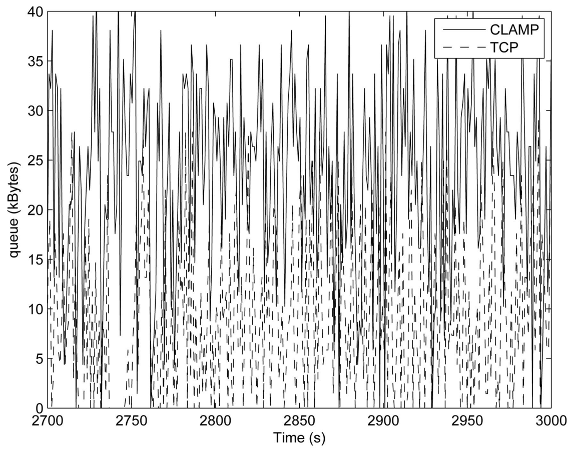


(b)

Fig. 6. Windows versus time for a flow with 30 ms propagation delay, with fast fading (10 m/s mobility). (a) 50 ms queuing delay. (b) 180 ms queuing delay.



(a)



(b)

Fig. 7. Queue versus time for flow with 130 ms propagation delay, with fast fading (10 m/s mobility). (a) 50 ms queuing delay. (b) 180 ms queuing delay.

delay of 50 ms, the *AWND* is almost always smaller than the *CWND* in the CLAMP experiments, only losing control to TCP occasionally and for very brief periods. The frequency and duration that CLAMP loses control, at 180 ms of queuing delay, is higher, due to the higher bandwidth-delay product: It becomes more likely that the *CWND* can drop below the *AWND* value at a high queuing delay. However, for the sake of mice TCP flows (and other delay-sensitive traffic), it is desirable to operate at a low-moderate queuing delay (50 ms or less).

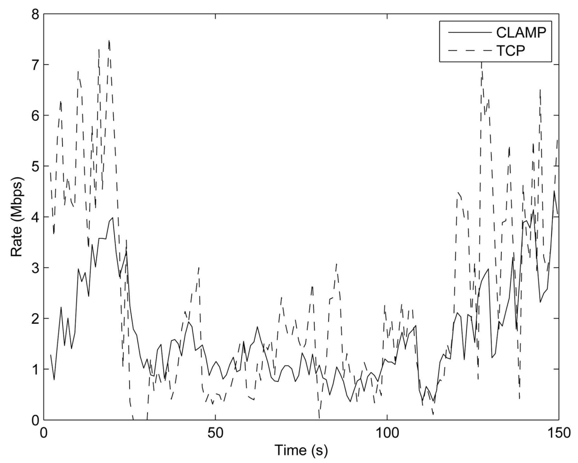
Fig. 7 plots the queue evolution over time, this time for the 130 ms flow. This flow gets less rate under TCP than under CLAMP due to its relatively large propagation delay. This is reflected in the mean queue size, which is lower under TCP, and in the frequency with which the queue drops to zero, which is higher under TCP. One observation is that the queue is very variable, even under CLAMP. CLAMP is not able to control the instantaneous queue size due to the coherence time of the channel being much shorter than the propagation delay of the flow.

Fig. 7 illustrates queue fluctuations when CLAMP is in equilibrium, and it does not indicate that the queue fluctuations are particularly reduced by CLAMP, as compared to TCP. In the next section, we will see a similar queuing behavior in a setting with flow arrivals and departures (see Fig. 8). Thus, we cannot claim that CLAMP can control the queue size at the AP. The wireless channel is highly variable, and the CLAMP protocol must average the queue size and the channel rates of the flows. In doing so, it can influence the *statistics* of the queues at the AP but not the instantaneous values. By changing the statistics, it can reduce the frequency with which the queue empties and thereby improve the performance of the scheduler.

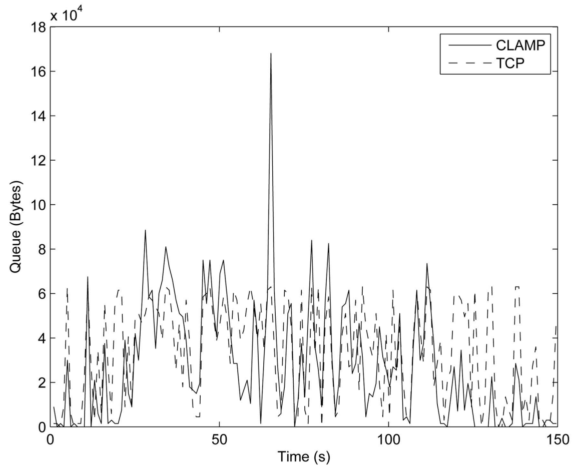
5.8 Flow Dynamics

In Section 5.1, we considered the window and queue dynamics as flows arrive and depart, but there was no fading. In Section 5.5, we have considered the equilibrium behavior of the protocols under dynamic fading but without the dynamics of flows arriving and departing. In this section, we consider both flow dynamics and fading: We study a scenario in which a long-lived TCP flow coexists with a number of short-lived TCP flows, which start and finish at random times.

In this experiment, the wireless channel is as described in Section 5.2, with a maximum rate of 10 Mbps and fast fading (10 m/s mobility), but there is only a single receiver with multiple flows sharing the same queue at the AP. One



(a)



(b)

Fig. 8. Flow dynamics: one long-lived flow and Poisson process of smaller flows. (a) Rate allocated to long-lived flow. (b) Queue dynamics.

flow is long-lived, for example, a file transfer running in the background, which lasts for 150 seconds. In addition, there are short-lived flows, which arrive as a Poisson process with a rate of 0.26 arrivals/s during the first 100 seconds. Each short-lived flow consists of a random number of packets, between 1 packet and 4,000 packets, using the triangle distribution. This models a Web browsing session running in parallel with the large download.

In Fig. 8, we plot the dynamics of the rate allocated to the long-lived flow and the total queue size at the AP (a function of all the flows in the system). We conclude that CLAMP and TCP both track the changing dynamics in a similar manner. The fluctuations in rate are less under CLAMP, but the queue fluctuations are quite similar (apart from a single spike in the queue size for CLAMP). Note that the spike could be avoided if we restricted the buffer size for CLAMP. The spike offsets the otherwise smaller values for the CLAMP queue size. The parameters were chosen to get the same average queue size for CLAMP and TCP. The buffer size for TCP is 65 Kbytes.

5.9 Choosing the Parameters

To provide a basis for comparing TCP with TCP+CLAMP, we have plotted the performance (above) as a function of

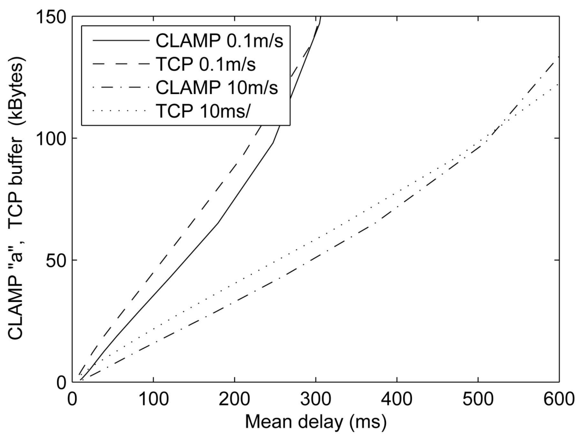


Fig. 9. TCP buffer size and CLAMP a versus queuing delay.

the average queuing delay, but the average queuing delay is a function of the underlying parameters. A question arises as to how these parameters can be set to obtain a particular desired point on, say, the throughput-delay trade-off curve. For TCP New Reno, we vary the buffer size at the AP. For CLAMP, we hold the buffer size fixed and vary the parameter a. Unfortunately, for the fading scenarios considered above, we have no analytical formula to describe the throughput-delay curve. The problem of buffer dimensioning for time-varying channels is an open area of research and is beyond the scope of this present paper.

In Fig. 9, we plot both the TCP buffer size and the CLAMP parameter a against the average queuing delay. This figure suggests that the problems of setting the buffer size (for TCP) and the parameter a (for CLAMP) are quite similar problems. There is no reason to suppose that one is more difficult than the other. It is possible to tune parameter a adaptively, but such algorithms are a topic for future research. In this present paper, the parameter is held fixed during a particular experiment, although we plot the results of different experiments across a range of parameter values.

5.10 Lower Data Rate Scenario

With regard to parameter settings, it is of interest to see if the CLAMP parameters need to be retuned for every network scenario. We have already examined two fading rates (fast and slow), and here, we consider a much lower channel rate. An important issue is the setting of the parameter τ, which is set in the mobile receivers and is not likely to be tunable. In the HDR experiments, we used $\tau = 500\ \mathrm{bytes}$, and we use the same value for the lower data rate scenario.

Identical experiments were undertaken for the same topology, as described in Tables 1 and 2, except that the bandwidth was reduced to $W_0 = 620\ \mathrm{kHz}$, and the maximum wireless link rate was reduced to $C_0 = 1.44\ \mathrm{Mbps}$. Due to the lack of space, we do not provide all the plots, but Fig. 10 presents the results for fairness at a high fading rate.

None of the general conclusions change: CLAMP does not increase the total throughput of the system, but it does allow the scheduler to provide fair service across users, and it provides fairness automatically to flows destined for the same receiver.

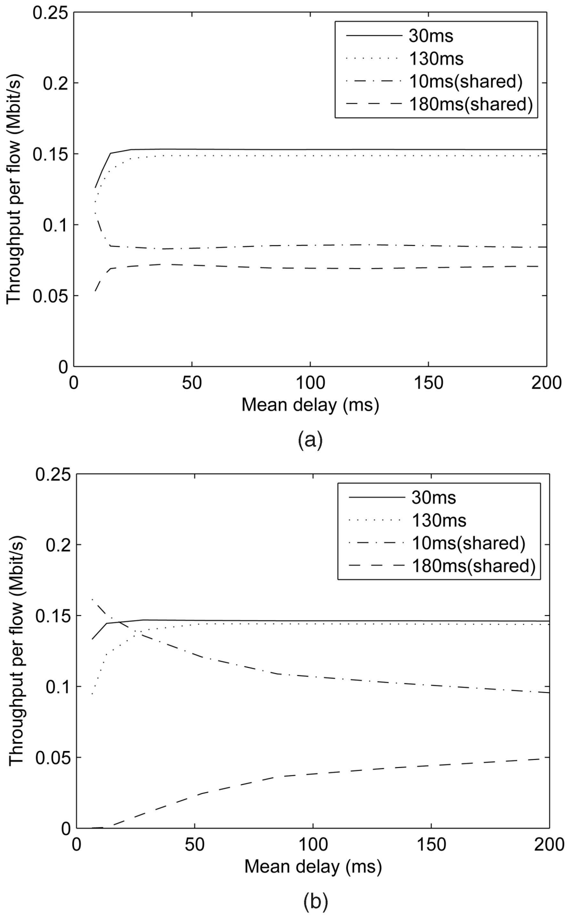


Fig. 10. Throughput per flow in the lower data rate scenario. Dotted-dashed lines are for flow 3, whereas dashed lines are for flow 4, both sharing a common queue at the AP. (a) CLAMP, with 10 m/s. (b) TCP, with 10 m/s.

Although τ is fixed, there is flexibility in the CLAMP protocol for the a parameter to be tuned to the conditions of the network. We observe that, in each scenario (a combination of data rate and fading rate), the total throughput versus delay has a fairly well-defined "knee," but the knee occurs at a different average delay in each case. It might be desirable to operate the system near the "knee," and this would require the AP to make a judicious choice of the parameter a.

We observe that the knees occur at lower delays in the lower data rate scenario. It is not surprising that the delay depends on the statistics of the service process, and we can contrast this behavior with what happens in a nonfading wireline scenario when the output rates are deterministic. In that case, the queuing delay should *increase* as the output rate of the queue decreases because the average queue size to meet a particular utilization does not depend on the service rate (this holds for the M/M/1 queue, for example). The present wireless scenario is quite different: The amount of queuing that is required for a particular utilization depends on the rate of fading and, more particularly, on the whole distribution of the service process. In this case, the size of the average queue that is required to keep all queues from emptying increases with the data rate.

6 Conclusions

This paper has studied the interaction between congestion control and scheduling in fading wireless channels. The scheduler attempts to allocate the bandwidth to the users in such a way that a user will only be allocated a rate when its channel is good but subject to fairness constraints. We observe a considerable interaction between TCP New Reno and the scheduler to the extent that rate allocations are determined, in part, by the propagation delays of the flows in the network.

Our approach was to provide a detailed packet-level simulation of a wireless network with a single AP, sending data to a number of mobile receivers. We model the physical-layer fading process, link-layer rate adaptation, scheduling, and flow control. Each mobile is provided with a separate queue at the AP and access to the radio channel is provided by the scheduler. TCP New Reno does the flow control, which is subject to possible intervention by the proposed CLAMP receiver-side algorithm, which allows us to measure the impact of Reno's window fluctuations on the performance of the scheduler.

CLAMP is based on a fluid model algorithm that is designed to avoid interaction between layers 2 and 4. The experiments have tested if this remains true in a more complicated model, in which there is fading, and CLAMP can only intervene at the receiver side. Experiments are mainly focused on a particular scenario with three mobile receivers, one of which is receiving two flows simultaneously. Parameters are selected to provide insight into the effect of propagation delays and rates of fading on performance.

For these experiments, we conclude that the fairness of the scheduler is compromised by TCP New Reno. Flows with large propagation delays get much worse performance, even if the scheduler is attempting to schedule them fairly. When we employ the receiver-side algorithm CLAMP to remove the window fluctuations, we find that the scheduler is able to achieve much better fairness than with pure TCP New Reno. As with TCP New Reno, the CLAMP throughput increases with the AP queuing delay. When the average queue size is larger, the queues empty less often. Nevertheless, for moderate queuing delay, the scheduler is able to provide a much fairer allocation under CLAMP than under TCP New Reno.

In the scenarios tested, better fairness is achievable when the fading is faster, but throughput is reduced at faster fading due to the increased fade margin required. Techniques such as incremental redundancy coding should alleviate this problem, but we did not investigate this. Inducing channel fluctuations in a controlled way [54] is a promising technique and may allow CLAMP + scheduler to operate fairly, even in slow fading scenarios.

Our approach has been to focus on a single-hop wireless downlink, in particular on the issues of time-varying channels and channel-state-aware scheduling. Although we have made specific assumptions about the channels for the simulation, the CLAMP algorithm can, in principle, be applied to any network.

A cleaner solution from a systems-level perspective would be a sender-side implementation in which the feedback signals are sent back to the sender in the ACKs, who then can set an appropriate *CWND* value. In that case, the flow control is primarily achieved via *CWND*. There is no inherent reason that the CLAMP control cannot be affected at the sender side. However, this requires extending CLAMP to the scenario in which all routers in the network provide congestion-indication feedback to the source. There has been much work on price-based congestion control for the core Internet [55], [56], [57], and CLAMP provides one such approach, but others are also possible.

A long-term goal is to generalize CLAMP to handle multiple bottlenecks and to incorporate the principles learned from studies such as the present one into a robust sender-side control that can handle the vagaries of time varying and lossy wireless channels, as well as the scaling problems associated with high-speed networks.

Acknowledgments

This project was supported by the Australian Research Council under grant DP0557611, "Control Protocols for Wireless Networks."

References

[1] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication.* Cambridge Univ. Press, 2005.

[2] S.H. Low, F. Paganini, and J.C. Doyle, "Internet Congestion Control," *IEEE Control Systems Magazine,* vol. 22, pp. 28-43, Feb. 2002.

[3] N.T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B.N. Bershad, "Receiver-Based Management of Low Bandwidth Access Links," *Proc. IEEE INFOCOM,* pp. 245-254, 2000.

[4] L. Kalampoukas, A. Varma, and K.K. Ramakrishnan, "Explicit Window Adoption: A Method to Enhance TCP Performance," *IEEE/ACM Trans. Networking,* vol. 10, pp. 338-350, June 2002.

[5] B. Sardar and D. Saha, "A Survey of TCP Enhancements for Last-Hop Wireless Networks," *IEEE Comm. Surveys,* vol. 8, pp. 20-34, http://www.comsoc.org/pubs/surveys, 2006.

[6] T. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Trans. Networking,* vol. 5, pp. 336-350, June 1997.

[7] T.V. Lakshman, U. Madhow, and B. Suter, "TCP/IP Performance with Random Loss and Bidirectional Congestion," *IEEE/ACM Trans. Networking,* vol. 8, pp. 541-555, Oct. 2000.

[8] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. Networking,* pp. 756-769, 1997.

[9] H.M. Chaskar, T. Lakshman, and U. Madhow, "TCP over Wireless with Link Level Error Control: Analysis and Design Methodology," *IEEE/ACM Trans. Networking,* vol. 7, pp. 605-615, Oct. 1999.

[10] D.A. Eckhardt and P. Steenkiste, "Improving Wireless LAN Performance via Adaptive Local Error Control," *Proc. 11th IEEE Int'l Conf. Network Protocols (ICNP '03),* pp. 327-338, Mar. 2003.

[11] R. Ludwig, A. Konrad, A. Joseph, and R. Katz, "Optimizing the End-to-End Performance of Reliable Flows over Wireless Links," *Kluwer/ACM Wireless Networks J.,* vol. 8, pp. 289-299, Mar.-May 2002.

[12] M. Meyer, "TCP Performance over GPRS," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '03),* pp. 1248-1252, Mar. 2003.

[13] H.-K. Shiu, Y.-H. Chang, T.-C. Hou, and C.-S. Wu, "Performance Analysis of TCP over Wireless Link with Dedicated Buffers and Link Level Error Control," *Proc. IEEE Int'l Conf. Comm. (ICC '01),* pp. 3211-3216, 2001.

[14] K. Ratnam and I. Matta, "WTCP: An Efficient Mechanism for Improving TCP Performance over Wireless Links," *Proc. Third IEEE Symp. Computers and Comm. (ISCC '98),* pp. 74-78, June 1998.

[15] M. Sagfors, R. Ludwig, M. Meyer, and J. Peisa, "Queue Management for TCP Traffic over 3G Links," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC '03),* pp. 1663-1668, Mar. 2003.

[16] J. Ma, J. Ruutu, and J. Wu, "An Enhanced TCP Mechanism—Fast-TCP in IP Networks with Wireless Links," *Wireless Networks,* vol. 6, pp. 375-379, Nov. 2000.

[17] M. Chan and R. Ramjee, "TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation," *Proc. ACM MobiCom,* pp. 71-82, Sept. 2002.

[18] M. Chan and R. Ramjee, "Improving TCP/IP Performance over Third Generation Wireless Networks," *Proc. IEEE INFOCOM,* vol. 3, pp. 1893-1904, 2004.

[19] S. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with Adaptive Pacing for Multihop Wireless Networks," *Proc. ACM MobiHoc,* pp. 288-299, May 2005.

[20] S. Pilosof, R. Ramjee, D. Raz, and P. Sinha, "Understanding TCP Fairness over Wireless LAN," *Proc. IEEE INFOCOM,* vol. 2, pp. 863-872, Mar. 2003.

[21] D. Leith, P. Clifford, D. Malone, and A. Ng, "TCP Fairness in 802.11e WLANs," *IEEE Comm. Letters,* vol. 9, pp. 964-966, Nov. 2005.

[22] V. Rughunathan and P. Kumar, "A Counterexample in Congestion Control of Wireless Networks," *Proc. Eighth ACM/IEEE Int'l Symp. Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '05),* pp. 290-297, Oct. 2005.

[23] V. Kawadia and P. Kumar, "Experimental Investigations into TCP Performance over Wireless Multihop Networks," *Proc. ACM SIGCOMM Workshop Experimental Approaches to Wireless Network Design (E-WIND '05),* pp. 24-29, Aug. 2005.

[24] R. Oliveira and T. Braun, "A Dynamic Adaptive Acknowledgement Strategy for TCP over Multihop Wireless Networks," *Proc. IEEE INFOCOM,* vol. 3, pp. 1863-1874, Mar. 2005.

[25] Y. Bai, A. Ogielski, and G. Wu, "Interactions of TCP and Radio Link ARQ Protocol," *Proc. 50th IEEE Vehicular Technology Conf. (VTC '99–Fall),* vol. 3, pp. 1710-1714, 1999.

[26] Z. Kostic, X. Qiu, and L.F. Chang, "Interactions between TCP and RLP Protocols in a Cellular System," *Proc. 53rd IEEE Vehicular Technology Conf. (VTC '01–Spring),* vol. 3, pp. 2244-2248, May 2001.

[27] M. Malkowski and S. Heier, "Interaction between UMTS MAC Scheduling and TCP Flow Control Mechanisms," *Proc. IEEE Int'l Conf. Comm. Technology (ICCT '03),* pp. 1373-1376, 2003.

[28] N. Samaraweera, "Non-Congestion Packet Loss Detection for TCP Error Recovery Using Wireless Link," *IEE Proc. Comm.,* vol. 146, no. 4, pp. 222-230, 1999.

[29] S. Biaz and N. Vaidya, "'De-Randomizing' Congestion Losses to Improve TCP Performance over Wired-Wireless Networks," *IEEE/ACM Trans. Networking,* vol. 13, pp. 596-608, June 2005.

[30] P. Cheng and S. Liew, "TCP Veno: Enhancement for Transmission over Wireless Access Networks," *IEEE J. Selected Areas in Comm.,* vol. 21, pp. 216-228, Feb. 2003.

[31] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," *Proc. ACM MobiCom,* pp. 287-297, 2001.

[32] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi, and R. Wang, "TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks," *Wireless Networks,* vol. 8, pp. 467-479, Sept. 2002.

[33] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for Wireless IP Communications," *IEEE J. Selected Areas in Comm.,* vol. 22, pp. 747-756, May 2004.

[34] V. Tsaoussidis and H. Badr, "TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains," *Proc. Eighth Int'l Conf. Network Protocols (ICNP '00),* pp. 12-21, Nov. 2000.

[35] C. Parsa and J. Garcia-Luna-Aceves, "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media," *Proc. Seventh Int'l Conf. Network Protocols (ICNP '99),* pp. 213-221, Oct. 1999.

[36] V. Tsaoussidis, H. Badr, and R. Verma, "Wave & Wait Protocol (WWP): An Energy-Saving Transport Protocol for Mobile IP Devices," *Proc. Seventh Int'l Conf. Network Protocols (ICNP '99),* pp. 301-308, Oct. 1999.

[37] X. Lin, N. Shroff, and R. Srikant, "A Tutorial on Cross-Layer Optimization in Wireless Networks," *IEEE J. Selected Areas in Comm.,* vol. 24, pp. 1452-1463, Aug. 2006.

[38] A. Eryilmaz and R. Srikant, "Fair Resource Allocation in Wireless Networks Using Queue-Length-Based Scheduling," *Proc. IEEE INFOCOM,* vol. 3, pp. 1794-1803, Mar. 2005.

[39] L. Ying, R. Srikant, E. Eryilmaz, and G. Dullerud, "Distributed Fair Resource Allocation in Cellular Networks," *IEEE Trans. Automatic Control,* vol. 52, pp. 129-134, Jan. 2007.

[40] S. Bae, K. Xu, S. Lee, and M. Gerla, "Measured Analysis of TCP Behavior across Multihop Wireless and Wired Networks," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '02),* vol. 1, pp. 153-157, Nov. 2002.

[41] Z. Fu, H. Luo, P. Zerfos, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Performance," *IEEE Trans. Mobile Computing,* vol. 4, pp. 209-221, Mar. 2000.

[42] L.L.H. Andrew, S.V. Hanly, and R.G. Mukhtar, "Analysis of Rate Adjustment by Managing Inflows," *Proc. Fourth Asian Control Conf. (ASCC '02),* pp. 47-52, 2002.

[43] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt, "Flow Aggregation for Enhanced TCP over Wide-Area Wireless," *Proc. IEEE INFOCOM,* pp. 1754-1764, 2003.

[44] L.L. Andrew, S.V. Hanly, and R. Mukhtar, "CLAMP: A System to Enhance the Performance of Wireless Access Networks," *Proc. IEEE Global Telecomm. Conf. (GLOBECOM '03),* vol. 7, pp. 4142-4147, Dec. 2003.

[45] T. Goff, J. Moronski, D. Phatak, and V. Gupta, "Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments," *Proc. IEEE INFOCOM,* pp. 1537-1545, 2000.

[46] J.E. Marshall, H. Górecki, K. Walton, and A. Korytowski, *Time-Delay Systems: Stability and Performance Criteria with Applications.* Ellis Horwood, 1992.

[47] D.-M. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems,* vol. 17, pp. 1-14, 1989.

[48] F. Paganini, J.C. Doyle, and S.H. Low, "Scalable Laws for Stable Network Congestion Control," *Proc. 40th IEEE Conf. Decision and Control (CDC '01),* pp. 185-190, 2001.

[49] L.L.H. Andrew, S.V. Hanly, and R.G. Mukhtar, "CLAMP: Maximizing the Performance of TCP over Low Bandwidth Variable Rate Access Links," technical report, Univ. of Melbourne, http://www.ee.unimelb.edu.au/people/hanly/T_Reports/clamp_tr-2004-02-01.pdf, 2007.

[50] *Transmission Control Protocol,* IETF RFC 793, Information Sciences Inst., Univ. of Southern California, 1981.

[51] L.L.H. Andrew, S.V. Hanly, and R.G. Mukhtar, "CLAMP: Differentiated Capacity Allocation in Access Networks," *Proc. 22nd IEEE Int'l Performance Computing and Comm. Conf. (IPCCC '03),* pp. 451-458, Apr. 2003.

[52] S. Floyd, T. Henderson, and A. Gurtov, *The NewReno Modification to TCP's Fast Recovery Algorithm,* IETF RFC 3782, 2004.

[53] E. Chaponniere, P. Black, J. Holtzman, and D. Tse, *Transmitter Directed Code Division Multiple Access System Using Path Diversity to Equitably Maximize Throughput,* US Patent 7123922, Patent and Trademark Office, 2002.

[54] P. Viswanath, D. Tse, and R. Laroia, "Opportunistic Beamforming Using Dumb Antennas," *IEEE Trans. Information Theory,* vol. 48, pp. 1277-1294, June 2002.

[55] F. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability," *J. Operational Research Soc.,* vol. 49, pp. 237-378, 1998.

[56] S.H. Low and D.E. Lapsley, "Optimization Flow Control I: Basic Algorithm and Convergence," *IEEE/ACM Trans. Networking,* vol. 7, pp. 861-875, Dec. 1999.

[57] J. Mo and J. Walrand, "Fair End-to-End Window-Based Congestion Control," *IEEE/ACM Trans. Networking,* vol. 8, pp. 556-567, Oct. 2000.

Lachlan L.H. Andrew received the BSc degree in computer science in 1992, the BE degree in electrical engineering in 1993, and the PhD degree in engineering in 1996 from the University of Melbourne, Victoria. Since 2005, he has been a senior research engineer in the Department of Computer Science, California Institute of Technology (Caltech). Prior to that, he was a senior research fellow at the University of Melbourne and a lecturer at RMIT, Australia. His research interests include the performance analysis of congestion control and other resource allocation algorithms for wireless and wired networks. He is a member of the Institution of Engineering and Technology (IET) and a senior member of the IEEE.

Stephen V. Hanly received the BSc (Hons) and MSc degrees from the University of Western Australia and the PhD degree in mathematics from Cambridge University, UK, in 1994. From 1993 to 1995, he was a postdoctoral member of technical staff at AT&T Bell Laboratories. He is currently an associate professor and reader in the Department of Electrical and Electronic Engineering, University of Melbourne, where he has been involved in teaching and research since 1996. He is an associate editor of the *IEEE Transactions on Wireless Communications*. He was the technical cochair of the 2005 IEEE International Symposium on Information Theory (ISIT). He is a corecipient of the Best Paper Awards in INFOCOM 1998 and the 2001 Joint IEEE Communications Society and Information Theory Society, both for his work with David Tse. His research interests are information theory and wireless networking. He is a member of the IEEE.

Rami G. Mukhtar received the BSc, BE, and PhD degrees from the University of Melbourne in 2003. He then worked for NEC Australia, participating in the development of chipsets for Universal Mobile Telecommunications System High-Speed Downlink Packet Access (UMTS HSDPA) cell phone handsets. From 2004 to 2005, he also represented NEC in the 3GPP RAN4 Standardization Working Group. In 2005, he joined VaST Systems Technology Corporation. He is currently a principle engineer with the Model Constructor Tools Group, VaST, leading the development of tools for developing high-speed timing accurate microprocessor models. He is a member of the IEEE.