

Improving TCP/IP Performance over Third-Generation Wireless Networks

Mun Choon Chan, *Member, IEEE*, and Ram Ramjee, *Fellow, IEEE*

Abstract—As third-generation (3G) wireless networks with high data rate get widely deployed, optimizing the Transmission Control Protocol (TCP) performance over these networks would have a broad and significant impact on data application performance. In this paper, we make two main contributions. First, one of the biggest challenges in optimizing the TCP performance over the 3G wireless networks is adapting to the significant delay and rate variations over the wireless channel. We present Window Regulator algorithms that use the receiver window field in the acknowledgment (ACK) packets to convey the instantaneous wireless channel conditions to the TCP source and an ACK buffer to absorb the channel variations, thereby maximizing long-lived TCP performance. It improves the performance of TCP selective ACK (SACK) by up to 100 percent over a simple drop-tail policy, with small buffer sizes at the congested router. Second, we present a wireless channel and TCP-aware scheduling and buffer sharing algorithm that reduces the latency of short flows while still exploiting user diversity for a wide range of user and traffic mix.

Index Terms—TCP, 3G network, long and short flows, delay and rate variation.

1 INTRODUCTION

THIRD-GENERATION wide-area networks (WWANs), which are based on the code division multiple access (CDMA) technology [1], are increasingly being deployed throughout the world. Although voice and, to some extent, short messaging service have been the predominant applications on the low-bandwidth wireless networks to date, the support for high-speed data in 3G networks with bandwidths up to 2.4 megabits per second (Mbps) should enable the widespread growth of wireless data applications. Since the vast majority of data applications use the TCP/IP protocols, optimizing the TCP performance over these networks would have a broad and significant impact on the user-perceived data application performance.

TCP performance over wireless networks has been studied over the last several years. Early research [2] showed that wireless link losses have a dramatic adverse impact on TCP performance due to the difficulty in distinguishing congestion losses from wireless link losses. These results have been one of the main motivations behind the use of extensive local retransmission mechanisms in 3G wireless networks [3], [4]. Although these local retransmission mechanisms solve the impact of wireless link losses on TCP performance, they also result in unavoidable variations in packet transmission delay (due to local retransmissions), as observed by TCP.

In addition to these delay variations, 3G wireless links use channel-state-based scheduling mechanisms [5] that

result in significant rate variations. The basic idea behind channel-state scheduling is to exploit user diversity. As wireless channel quality of different users varies independently due to fading, the total cell throughput can be optimized if the scheduling priority is given to the user with higher channel quality. For example, Qualcomm's proportional fair (PF) scheduler [6] exploits this idea while providing for long-term fairness among different users. Thus, although these scheduling mechanisms maximize the overall link-layer throughput, they can also result in significant variations in instantaneous individual user throughput, as observed by TCP.

Furthermore, Internet traffic consists of a small number of long-lived flows that make up a large part (in bytes) of the total traffic and a large number of the flows (in count) that are short lived. This is especially true with the popularity of applications such as Web browsing. As a result, optimizing the 3G wireless data system for short-lived TCP flows is also important. For short flows, the goal is to maximize the throughput and, for long flows, the goal is to minimize the average transfer latency.

In this paper, we make two contributions for improving the performance of applications in which the servers are attached to the wired network and the clients are attached to the 3G wireless network. First, we design a network-based solution called the *Window Regulator*, which maximizes the throughput of long-lived TCP for any given buffer size at the congested router in the presence of large variations in rate and delay. Second, we present a scheduling and buffer sharing algorithm that reduces the latency for short flows. The algorithm is able to provide short-flow differentiation and exploit user diversity, allowing the wireless channel to be utilized efficiently.

Variations in delay and bandwidth cause throughput degradation of long-lived TCP flows due to the difficulty in estimating the appropriate throughput (that is, congestion window size and round-trip time (RTT)) of the end-to-end

• M.C. Chan is with the Department of Computer Science, School of Computing, 3 Science Drive 2, Singapore 117543, Republic of Singapore. E-mail: chanmc@comp.nus.edu.sg.

• R. Ramjee is with Microsoft Research, "Scientia," 196/36 2nd Main, Sadashivnagar, Bangalore—560080, India. E-mail: ramjee@microsoft.com.

Manuscript received 14 July 2005; revised 30 Nov. 2006; accepted 24 Apr. 2007; published online 23 July 2007.

For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0198-0705. Digital Object Identifier no. 10.1109/TMC.2007.70737.

path at the TCP source. When the source overestimates the available throughput, it causes multiple and frequent packet drops at the congested router buffer, resulting in poor TCP throughput. The Window Regulator algorithm uses the receiver window field in the acknowledgment (ACK) packets to convey the instantaneous wireless channel conditions to the TCP source and an ACK buffer to absorb the channel variations, thereby maximizing long-lived TCP performance. Window Regulator ensures that the source TCP *operates in the window-limited region*, resulting in a congestion-loss-free operation. Although the receiver window field of the ACK packets have been used for ensuring fairness and regulating flows in wired networks [7], we show that this scheme does not perform as well over wireless links with variations. Window Regulator can achieve high goodput and maximum long-lived TCP performance for even small values of the buffer size, reasonably large wired latencies, and small amounts of packet loss in the wired or wireless links. For example, it improves the performance of TCP Selective ACK (SACK) by up to 100 percent over a simple drop-tail (DT) policy.

Although maximizing the throughput is important for long-lived TCP flows, applications like HTTP transfer and text messaging consist of many short-lived TCP flows. For these short-lived flows, minimizing the flow completion time requires giving preferential treatment to short flows and involves a different mechanism from maximizing the throughput. In addition, in a 3G environment, any short flow differentiation scheme has to take into account the wireless channel condition in order to take advantage of user diversity. We show that a scheduling algorithm that provides differentiation but does not fully exploit user diversity can have the adverse effect of increasing short-flow latencies and decreasing long-flow throughput at the same time. We present a TCP-aware scheduling and buffer sharing algorithm that reduces the latency of short TCP flows while still exploiting user diversity for a wide range of user and traffic mix.

The rest of this paper is organized as follows: In Section 2, we review related work. In Section 3, we present the architecture. In Section 4, we present a model for the receiver-window-based algorithms. This model serves as our motivation for the design of the Window Regulator algorithms. We compare its performance to several algorithms, including the ACK Regulator (AR), through extensive simulations in Section 5. In Section 6, we present the buffer sharing and scheduling algorithm for differentiation of short flows and discuss its performance. We finally present the conclusions in Section 7.

2 RELATED WORK

The vast majority of related work on TCP performance over wireless networks [2], [8], [9], [10] has concentrated on reducing the impact of TCP misreacting to wireless losses as congestion losses that result in poor throughput. As mentioned earlier, link-layer retransmission in 3G wireless links [3], [4] have effectively reduced the loss rate of wireless links to well under 1 percent, thereby minimizing the impact of wireless link loss on the TCP performance.

Wireless connections also suffer from short and long pauses due to handoffs and disconnection. Network-based and end-to-end solutions like M-TCP [11] and Freeze-TCP [12], respectively, address this problem by sending an ACK with the receiver window size set to 0. This freezes the connection and timers at the TCP source until the connection is reestablished, thereby avoiding time-outs and improving the TCP performance. However, these solutions require advance warning before link breakage occurs. In addition, they are designed for a complete break in link transmission and are not appropriate for large variations in rate and delay.

There have been several studies that examine the impact of wireless link variations on the TCP performance (for example, [13], [14], [15], [16], and [17]). Large delay variations resulting in delay spikes can cause spurious time-outs in TCP where the TCP source incorrectly assumes that a packet is lost while the packet is only delayed. This unnecessarily forces the TCP into slow start, adversely impacting the TCP performance. In [17], the authors propose enhancements to the TCP timer calculations to better track the RTT of the connection, thereby avoiding spurious time-outs. Another approach to dealing with spurious time-outs, based on a TCP sender algorithm (forward retransmission timeout (F-RTO)), is presented in [18]. The authors in [14] and [15] present several recommendations for TCP hosts such as enabling the time stamp option and using large windows for improving performance over wireless networks. A measurement study of TCP performance over a commercial WWAN testbed can be found in [19].

In [13], the authors present the AR solution for avoiding multiple packet drops at the congested router for long-lived flows. Since we compare our proposal with the AR solution in our simulations, we now briefly present an overview of AR. AR is a network-based solution, implemented at the Radio Network Controller (RNC), that aims at achieving the classic sawtooth congestion window behavior of the TCP source, even in the presence of varying rates and delays. It achieves this by controlling the buffer overflow process at the bottleneck link through the regulation of the transmission of TCP ACKs back to the source. It accommodates packets from the TCP source without buffer overflows until a predetermined threshold (based on an estimated TCP source congestion window size) is reached and then causes a single-packet loss to notify the TCP source to halve its congestion window. This process repeats and thus ensures that the TCP source operates in the congestion-avoidance phase (sawtooth behavior) with high probability. Details of the AR algorithm can be found in [13].

Although AR was shown to increase the throughput of long-lived TCP flows, it has a few drawbacks. First, AR needs to estimate the number of data packets in transit from the source: Errors in this estimation, for example, due to variations on the wired network, could lead to multiple-packet drops, resulting in lowered throughput. Second, since it intentionally causes single-packet drops to force the TCP source to go into the congestion-avoidance phase, it inherently cannot achieve the maximum goodput. The

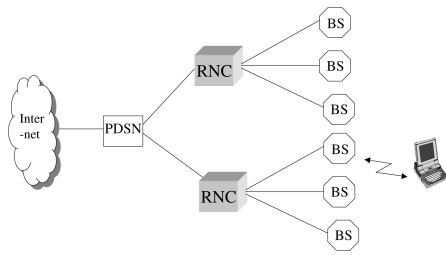


Fig. 1. Wireless network architecture.

Window Regulator presented here can be seen as an improvement to the AR solution.

The use of the receiver window field in the ACK packets to throttle the TCP source is not new. In [7], the authors implement a receiver-window-based technique with ACK pacing at the bottleneck router to reduce burstiness and ensure fairness among different flows. In [20], the authors use the receiver window field to better manage the TCP throughput over a connection that spans both IP and asynchronous transfer mode (ATM) networks. However, as we shall see later, since these solutions do not explicitly cater to significant rate and delay variations, they do not perform well over wireless networks.

Differentiation for short flows over long flows in wired networks has been studied in [21] and [22]. The basic idea is to identify short flows heuristically through the use of a simple threshold for the bytes transmitted and another threshold for an idle period and then give priority to short flows. The authors in [21] and [22] use random early drop (RED) with different weights for short and long flows to provide differentiation. However, tuning RED for wireless links that exhibit significant variation is hard. Furthermore, wireless networks already employ per-user buffering in order to implement reliable link layers with local retransmissions; thus, the utility of an algorithm like RED is reduced since we already have per-user state information available.

Differentiation for short flows in wireless networks has also been studied in [23] and [24]. In [24], Foreground-Background (FB) scheduling [25] is used for scheduling between multiple flows of a single user, and PF scheduling is used for scheduling packets across users. No new algorithm is proposed for interuser scheduling in place of PF. In this paper, we propose both new intrauser and interuser scheduling algorithms. In [23], the goal is to minimize the average stretch (the ratio of the actual job completion time over the minimum job completion time) over all jobs. This approach requires advance knowledge of all job sizes, which may not be available.

3 ARCHITECTURE

A simplified view of the 3G wireless access network architecture is shown in Fig. 1. The base stations are managed by an RNC. The RNC performs handoffs and terminates the Radio-Link Protocol (RLP), which is responsible for improving the reliability of the wireless link through link-layer retransmissions. The Packet Data Service Node (PDSN) terminates the Point-to-Point Protocol (PPP),

performs the function of a Mobile IP Foreign Agent, and interfaces to the public Internet. In this architecture, the RNC receives IP packets that are encapsulated in the PPP from the PDSN. These IP packets, whose sizes depend on the maximum transmission unit (MTU) being supported, are fragmented into smaller radio frames by using the RLP protocol and transmitted to the base station. The sizes of the RLP packets vary from 128 to 512 bytes [3]. The size of the packet being transmitted depends on the current channel condition. Better channel condition (or higher bit rate) allows larger packets to be transmitted in a single time slot. The base station then schedules the transmission of the packet over the air. In the case of a wireless frame loss, the RLP performs retransmission of the radio frames. In this architecture, the RNC maintains a per-user packet buffer and drops packets during congestion when the per-user buffer is full.

We consider only applications where the server resides in the wired network and the client resides in the 3G networks. For the algorithms discussed in this paper, we assume that the RNC can be extended to provide additional classification capability. First, in order to implement the Window Regulator schemes presented in Section 4, the RNC needs to differentiate among flows going to different users. This is already being done in the current RNC. Next, in order to implement the short-flow differentiation scheme presented in Section 6, the RNC needs the additional capability to distinguish between different TCP flows based on the IP addresses and port numbers inside the packet headers.

In general, the RNC must carefully choose how much buffer is to be allocated to a single user. Placing a strict upper limit on the maximum buffer allocated to a single user is necessary because of several reasons: 1) during handoffs, the per-user buffers have to be either quickly moved from one RNC/base station to another or flushed, and large buffers can result in long handoff latencies and/or wasted bandwidth on the access network, 2) scalability and cost considerations also place a limit on the buffer size, as the RNC must scale to the order of 100,000 users or more, and 3) stale data (for example, a user clicking “reload” on a browser or terminating an FTP flow) will still be sent over the wireless link, and large buffers imply a larger amount of stale data, wasting limited wireless bandwidth.

4 WINDOW REGULATOR

Variations in delay and bandwidth cause throughput degradation of long-lived TCP flows due to the difficulty in estimating the appropriate throughput. When the source overestimates the available throughput, it causes multiple and frequent packet drops at the congested router buffer, resulting in poor TCP throughput. One approach to avoid buffer overflow at the congestion (RNC) buffer is to control the amount of data packet that can arrive at the RNC data buffer.

In AR, this control is performed by releasing ACKs only if there are vacancies in the data buffer. However, AR needs to estimate the number of data packets in transit from the source, and it intentionally causes single-packet drops to force the TCP source to go into the congestion-avoidance

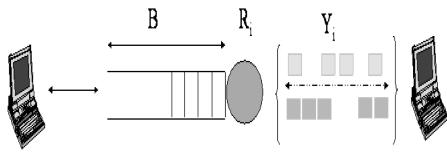


Fig. 2. Simple model of the wireless network.

phase. A different approach is taken by the Window Regulator algorithms. In Window Regulator, the maximum number of packets that can be accommodated is estimated, and this value is written onto the receiver window field in the ACK packets. If the estimated window is less than or equal to the actual bandwidth-delay product (BDP), the Window Regulator ensures that the source TCP *operates in the window-limited region*, resulting in a congestion-loss-free operation. However, if the estimate is too small, the channel will be underutilized. In the following paragraphs, we present three Window Regulator algorithms with strictly increasing estimates of window size.

First, we describe the Window Regulator algorithms by using a simple model of a single flow in a wireless network, as shown in Fig. 2. The base stations and the RNC are collapsed into a single bottleneck link, with the per-flow buffer size at the bottleneck node set to B . For simplicity of elucidation, let us further assume that there are no losses or reordering on the wired or the wireless links and let the latency on the wired network be insignificant compared to wireless delays. Note that the analysis of the performance of TCP has an extensive literature (for example, [26], [27], [28], and [29]). The aim of our analysis is not to propose a new analytical technique but to simply motivate and explain the heuristics behind the Window Regulator algorithms. When we evaluate the performance of the Window Regulator algorithms in Section 5, we will relax the assumptions on delay and packet losses.

In general, the Window Regulator algorithm attempts to maximize the TCP throughput by ensuring that there is always at least one packet available in the bottleneck node to be transmitted (no underflow) and there is no packet loss due to buffer overflow (this leads to retransmission, which is wasted bandwidth, and reduction in congestion window size or time-out, which could then lead to an underflow). In order to understand the performance of Window Regulator, we focus on the *arrival events* on the data queue.

Consider the arrival of the i th packet. Let Y_i represent the number of packets in the “wireless pipe” when packet i arrives. Y_i is a function of the varying rates and delays on the forward and reverse directions on the wireless link.¹ Finally, let $N_i (\leq B)$ be the number of packets in the bottleneck link when packet i arrives. Since we assume that the wired network latency is insignificant, all data packets are buffered in the bottleneck link and all ACKs are acknowledged immediately with no outstanding packets in the wired network. The sum of data packets in the buffer and in the “wireless pipe” equals the TCP window size W_i :

$$W_i = N_i + Y_i + 1. \quad (1)$$

1. Later, in the Window Regulator algorithms, we use the variable Y to estimate Y_i .

On Ack Enque, set $W^r = B$ and transmit Ack to source

Fig. 3. Window Regulator-Static.

Equation (1) is always true since the wired network latency is assumed to be insignificant. In Section 5, we study the impact of varying the wired network latency on the throughput of the different algorithms.

Note that, for ease of explanation, the variables that are used in this section are in units of (MTU) packets. In the actual implementation, since TCP is a stream-oriented protocol, these variables have to be converted into units of bytes by using the sequence numbers available in TCP headers. This is a straightforward translation and the equations derived in this section remain valid even after this translation.

Now, for TCP to operate without any buffer underflow, the window size W_i must obey

$$\forall i \quad Y_i + 1 < W_i \quad (\text{no underflow}). \quad (2)$$

Equation (2) states that, for there to be no underflow, the window size at the arrival of the i th packet must be greater than the instantaneous number of packets in the wireless pipe and at least one packet already waiting in the buffer. In general, this is a necessary but not sufficient condition. However, since we assume that (1) is true, $N_i \geq 1$ when there is no underflow, and thus, (2) is also a sufficient condition for no underflow. For there to be no overflow (packet drop), from (1), $B \geq W_i - Y_i$. In other words,

$$\forall i \quad Y_i + B \geq W_i \quad (\text{no overflow}). \quad (3)$$

Similarly, (3) is necessary and sufficient to prevent overflow. The difficulty in choosing an appropriate TCP window size is in adapting to the variations in Y_i while ensuring that (2) and (3) are not violated.

Next, we consider three Window Regulator algorithms that set the receiver window size W^r in the ACK packets in order to manage the window size at the TCP source. For the purpose of this discussion, we assume that each packet is acknowledged. However, the algorithms can be generalized to handle ACKs that represent more than one packet as well and a brief discussion is given in Section 4.4. Furthermore, we assume that the original value of the receiver window size in the ACK is larger than the value set by the algorithm (that is, the actual receiver window is not a bottleneck). If this assumption does not hold, we can simply send back the ACK with the receiver window size unchanged.

4.1 Window Regulator-Static (WRS)

The first algorithm, called WRS, is described in Fig. 3. The receiver window size W^r is set statically to the buffer allocated for that flow. WRS is an existing and well-known algorithm that is used in wired routers for guaranteeing bandwidth and ensuring fairness (for example, see Packeteer [7]). It is not a contribution of our work and is used only as a baseline for comparison purposes.

Note that this algorithm is very conservative as it easily satisfies (3) (no overflow) since $W_i = B$ and $\forall i Y_i \geq 0$ as the number of packets in the wireless pipe cannot be negative. However, depending on the size of the buffer in relation to

| |
|---|
| <p>On Data packet Deque, set $Y = Y + 1$ On Ack Enque, set $Y = Y - 1$ and $W^r = Y + B$, transmit Ack to source</p> |
|---|

Fig. 4. Window Regulator-Dynamic.

the variation of the wireless pipe, the queue can be idle, resulting in a loss of throughput. In other words, the utilization of the queue Q_{WRS} of this algorithm is approximated by

$$Q_{WRS} = \frac{1}{k} \sum_{i=1}^k 1\{B > Y_i + 1\}, \quad (4)$$

where k is the total number of packets arrived and $1\{B > Y_i + 1\}$ is the delta function:

$$1\{B > Y_i + 1\} = \begin{cases} 1, & B > Y_i + 1 \\ 0, & \text{otherwise.} \end{cases}$$

The approximation is exact if the arrival process is Poisson (Poisson arrivals see time averages (PASTA)). For the case of bursty arrivals, which is the expected case here, (4) is an upper bound.

4.2 Window Regulator-Dynamic (WRD)

One simple way to extend the WRS algorithm is to track the changes in Y_i and convey this in each ACK packet flowing back to the sender. We call this the WRD algorithm, and it operates as shown in Fig. 4, where Y is the current estimate of the size of the wireless pipe.

Assuming that the wired latency is insignificant, if a data packet i arrives due to this ACK departure, then $W_i = W^r$ and $Y_i = Y$. Note that this algorithm might end up reducing the receiver window size W_i compared to W_{i-1} , as Y is a varying quantity. However, this algorithm never reduces the receiver window size between consecutive ACKs by more than one (the reception of an ACK reduces the packets in the wireless pipe by one since we assume that every packet is acknowledged). Thus, transmitting an ACK with a reduced window size does not shrink the window but just freezes the window from advancing (no new packets will be transmitted in response to an ACK with the window reduced by one).

This algorithm is also conservative, as it satisfies (3) (no overflow), but it uses a higher window size compared to WRS. Since both algorithms operate in the window-limited region of TCP (where the throughput is given by W/RTT , where W is the window size), the throughput of WRD is as good or is better than the throughput of WRS since

$$\text{If } \frac{W}{RTT} \leq R \text{ then } \frac{W}{RTT} \leq \frac{W+k}{RTT+k/R} \forall k \geq 0,$$

where R is the average rate of the connection. The window size of WRS is B , and the window size of WRD is $B + Y$. Y is the difference in window size between WRD and WRS and is always nonnegative.

However, this algorithm can also result in underflow when the whole buffer is drained before the reception of an ACK (causing a sudden large increase in the number of packets in the wireless pipe). Thus, although Y still tracks

| |
|---|
| <p>On Data packet Deque, $Y = Y + 1$ if there is an Ack stored in the ack buffer, then $W^r = Y + B + B_a$ and transmit Ack to source endif On Ack Enque, set $Y = Y - 1$ and $W^r = Y + B + B_a$ if (new Ack AND($W^r <$previous transmitted W^r)) store Ack in the Ack Buffer and set $B_a = B_a + 1$ else transmit Ack to source endif</p> |
|---|

Fig. 5. Window Regulator with ACK Buffer.

the increase in the size of the wireless pipe, there are no ACKs available to convey this information to the TCP source. The utilization of the queue Q_{WRD} of this algorithm is given by

$$Q_{WRD} = \frac{1}{k} \sum_{i=1}^k 1\{Y_i + B > Y_{i+1}\}. \quad (5)$$

The equation can be rewritten as

$$Q_{WRD} = \frac{1}{k} \sum_{i=1}^k 1\{B > Y_{i+1} - Y_i\}. \quad (6)$$

In other words, the number of packets in the buffer must be larger than the number of packets transmitted between two packet arrivals for there to be no underflow in this algorithm. Comparing (6) to (4), we can again see that the utilization of WRD is always greater than or equal to the utilization of WRS since $Y_i \geq 0, \forall i$.

4.3 Window Regulator with ACK Buffer (WRB)

One of the problems with the WRD algorithm is that, when $Y_{i+1} - Y_i$ increases beyond B and the buffer drains completely, the congested node may not have any ACKs to provide feedback to the TCP source. One way to overcome this is to maintain an ACK buffer in the reverse direction. As mentioned earlier, when the window size is reduced by one in the WRD algorithm, the TCP source does not transmit any packet. Thus, this feedback is not used by the TCP source (other than when resetting its timers for this packet). If, instead, this ACK is stored in an ACK buffer, we can use it to indicate any increase in size of the wireless pipe as soon as it occurs and thereby allow the transmission of a data packet from the source. We call this the WRB algorithm and it operates as shown in Fig. 5. B_a is the size of the ACK buffer and is set to 0 initially.

Note that, in the WRB algorithm, B_a will always increase until it converges to some value Y_{max} and $W_{i+1} \geq W_i, \forall i$. In practice, there is an upper bound for Y_{max} because the number of packets in the wireless channel is limited by the flow-control mechanism implemented in RLP. Equation (2) is obviously true, since $\forall i, Y_{max} \geq Y_i$, and (3) is true because the wired delay is insignificant. The queue utilization Q_{WRB} of this algorithm is given by

$$Q_{WRB} = \frac{1}{k} \sum_{i=1}^k 1\{Y_i + B + B_a > Y_{i+1}\}. \quad (7)$$

If we do not limit the size of the ACK buffer (since ACKs consume very little memory and do not impact the latency of new flows), then B_a will grow large enough to absorb the

maximum variation on the wireless link. Therefore, $Pr(Y_i + B + B_a > Y_{i+1})$ approaches 1. Thus, if the flow lasts long enough, and assuming that the wired latency is insignificant, the WRB algorithm achieves the maximum utilization of 1. The queue utilization Q_{WRB} is

$$Q_{WRB} = \frac{1}{k} \sum_{i=1}^k 1 = 1. \quad (8)$$

4.4 Discussion

Although we strongly believe that the bottleneck is most likely to be in the 3G wireless link, our solutions should be able to handle the low-probability cases where the wired network becomes the bottleneck. By holding ACKs in the ACK buffer, WRB can run into a deadlock in the presence of loss in the wired network. Fortunately, this situation can be easily identified since there will be little queuing on the RNC when the wireless link is no longer a bottleneck. Thus, in WRB, an ACK is released whenever the data queue is empty. The ACK release mechanism works in the following way: On the double-ended queue (deque) of a data packet or enqueue of an ACK packet, if the data queue is detected to be empty the first time, two ACKs are sent. On the subsequent enqueue or deque, if the data queue remains empty, the number of ACKs released is double until all ACKs are released. This process resets when the data queue becomes nonempty. A similar reset mechanism is also found in the AR [13] to handle packet loss. The ACKs are not released all at once because, by releasing too many at the same time, ACK compression occurs. When too many data packets are transmitted by the sender in a short time, congestion can occur.

The next issue concerns the assumption that every data packet is acknowledged by the receiver, that is, a delayed ACK in TCP is disabled. If the mobile host uses delayed ACKs, the responsiveness of the algorithms will be affected, and the performance will degrade, depending on the intensity of delay or rate variation and how long the ACKs are delayed. Typically, the TCP delayed-ACK timer is set to 200 ms, which is not large as compared to the wireless link delays, and thus should not have a significant impact on the performance of the Window Regulator algorithms.

Finally, Window Regulator has no impact on the end-to-end congestion control behavior or fairness and does not change the additive-increase, multiplicative-decrease (AIMD) behavior of the TCP sender. Changing the receiver window size at the RNC in all these algorithms will not result in unfairness to other TCP flows sharing the wired bottleneck link. This is because all TCP senders use the minimum of the congestion window and the receiver (advertised) window values for computing the number of packets to be transmitted. As Window Regulator can only reduce the receiver window, its use will not result in larger bandwidth consumption.

5 PERFORMANCE OF LONG-LIVED TCP FLOWS

In this section, we study the performance of the Window Regulator algorithms through extensive simulation and compare their performance with those of AR [13] and a

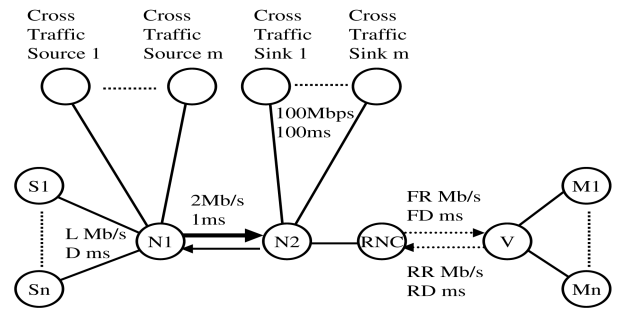


Fig. 6. Simulation topology.

simple DT policy. All simulations are performed using ns-2 with modifications that implement high data rate (HDR) scheduling and variable link delays.

The simulation topology used is shown in Fig. 6. S_i , $i = 1 \dots n$, corresponds to the set of TCP source nodes sending packets to a set of the mobile TCP sink nodes M_i , $i = 1 \dots n$. Each set of S_i , M_i nodes forms a TCP pair. The RNC is connected to the M_i nodes through a V (virtual) node for simulation purposes. L , the bandwidth between S_i and the router N_1 , is set to 100 Mbps, and D is set to 1 ms, except in cases where D is explicitly varied (in Section 5.4). The forward wireless channel is simulated with a model for 3G1X-EVDO (HDR) system, and the reverse wireless channel has rate $RR = 64$ kilobits per second (Kbps) and delay RD . HDR uses PF scheduling (which exhibits both variable rate and variable delay), and the buffer management scheme used is tail drop. The model for the wireless link used is based on a Rayleigh fading channel model. FD is modeled as having a uniform distribution with a mean of 75 ms and a variance of 30, and RD is modeled as having a uniform distribution with a mean of 125 ms and a variance of 15. These are conservative values and were used in [13]. Even in the presence of variable delays, we ensure that packets are delivered in order. The packet arrival time is computed as the maximum of the scheduled time and the arrival time of the previous packet. Fig. 6 also shows a set of nodes used for generating cross traffic and is used in Section 5.5, where the impact of loss is investigated. The bottleneck link in this case is set to be 2 Mbps, which is double the maximum throughput achievable in the 3G network simulated. Hence, it is only relevant for the simulation in Section 5.5. The links between the cross traffic sinks and the router N_2 have a bandwidth of 100 Mbps and a delay of 100 ms. Unless mentioned otherwise, we use TCP SACK with the time stamp option for long-lived flows that last at least 1,000 seconds in each simulation run. All simulations use a packet size of 1,000 bytes. Assuming that the maximum bandwidth available to each user is 600 Kbps and the average delay is 200 ms, the BDP is 120 Kbits or 15 packets. The default per-user queue size is set to 20 packets, which is about 1.3 times BDP of bottleneck link. The TCP maximum window size is set to 500 Kbytes. Using such a large window size ensures that TCP is never window limited by the TCP source in all experiments.

We next evaluate the performance of the five algorithms, that is, the three Window Regulator algorithms (WRS, WRD, and WRB), AR, and DT. Throughput is measured at

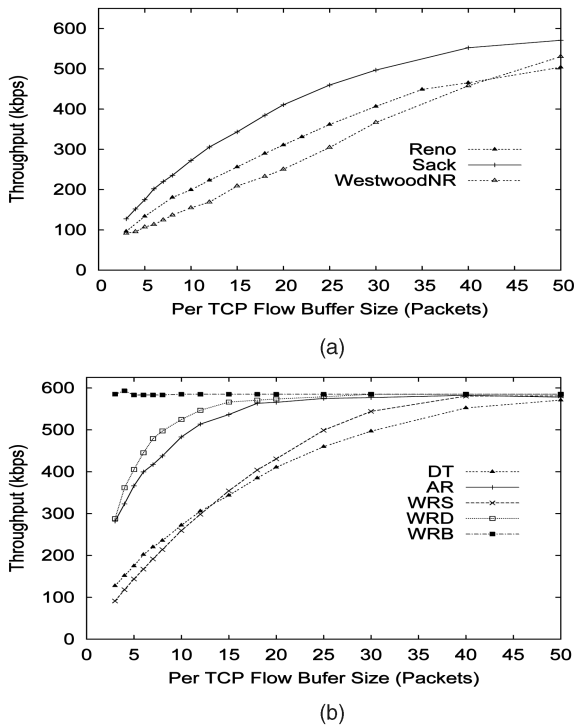


Fig. 7. Throughput versus queue length for a single user/flow. (a) Different TCP variants. (b) TCP sack.

the TCP receiver and is the rate of useful bytes/packets transmitted by the sender. Hence, the throughput measured is the **goodput**. The effect of buffer size is studied in Sections 5.1 and 5.2, the trade-off between throughput and RTT in Section 5.3, the longer wired network latency in Section 5.4, and the wired network loss in Section 5.5.

5.1 Throughput versus Buffer Size (Single User)

In this section, the effect of RNC buffer size on throughput is presented for the case of a single user with a long-lived TCP flow. As a baseline, we compare the performance of TCP Reno, TCP SACK, and TCP Westwood-NR [30] using the simulation topology for a single user. Fig. 7a plots the throughput performance of these three different TCP variants. TCP SACK performs the best among the three variants. In particular, note that, even though TCP Westwood uses a bandwidth estimation technique for rate control, it does not work well in the presence of substantial bandwidth and delay variation that is typical over a wide-area wireless link.

We will focus on the performance of TCP SACK in the rest of the paper since it exhibits the best performance among the different TCP variants. Fig. 7b plots the throughput performance of the TCP flow by using the five algorithms for hosts using TCP SACK. As shown in the figure, DT performs poorly, as the variations over the wireless link cause significant throughput degradation. Interestingly, the WRS algorithm also has very low throughput for a given buffer size and even underperforms DT in some cases. The improvement of WRS over DT ranges from -28 percent (for small buffer sizes) to 10 percent.

AR performs better than WRS and DT, but since it can only use the technique of holding back ACKs to signal the

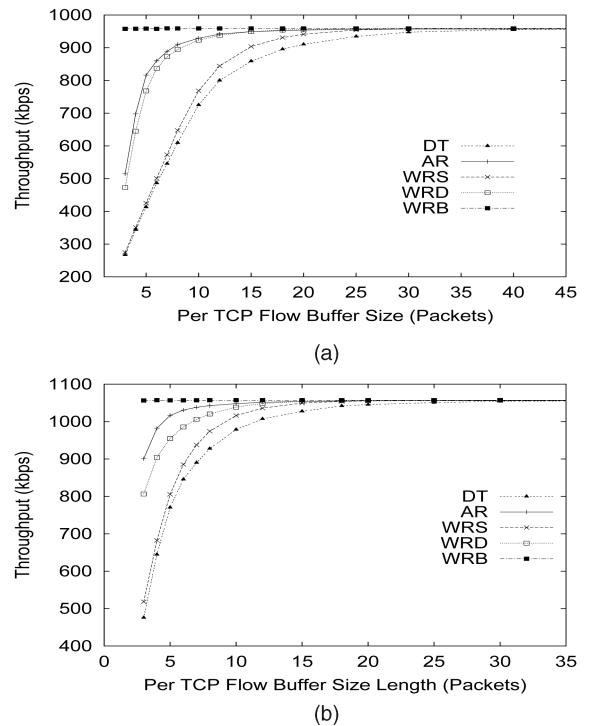


Fig. 8. Aggregate throughput versus queue length for TCP SACK with multiple users/flows. (a) Four users. (b) Eight users.

source to slow down during buffer buildup, it does not achieve the maximum throughput gains. The improvement of AR over DT ranges from 1 percent to 120 percent. The WRD algorithm delivers close to the maximum throughput for reasonable buffer sizes (> 15), but for small buffer sizes, it can lead to underutilization of the link, as the node may not have any ACKs to provide feedback during sudden increases in the number of packets in the wireless channel. The improvement of WRD over DT ranges from 2 percent to 138 percent with the biggest improvement occurring between buffer sizes of 5 to 15.

As expected, WRB outperforms all the other algorithms and delivers the highest throughput with improvement over DT ranging up to 360 percent for very small buffer sizes and close to 100 percent of improvement when one BDP worth of buffer (15) is used. In addition, with the use of ACK buffers, as long as packets continue to be transmitted over the wireless channel and ACKs continue to be sent uplink back to the sender, spurious time-outs caused by ACK storage are not observed to have occurred.

5.2 Throughput versus Buffer (Multiple Users)

Fig. 8 shows the effect of buffer size on throughput for four and eight users. The results are similar to the single-user case in terms of relative performance of the various algorithms, except for the cases of AR and WRD, where AR now outperforms WRD. When the number of users (flows) increases, the gain decreases (but is still substantial) since the likelihood of the buffer being empty is reduced, even under the DT policy.

The result for eight users is similar to that of four users, except that the gap between AR and WRD widens. One of the reasons is that, as the number of users increases, the

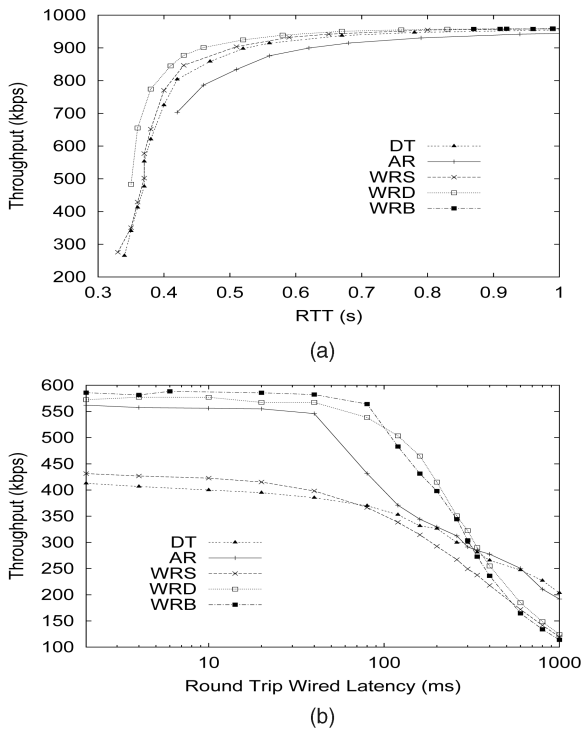


Fig. 9. Throughput versus RTT and wired latency. (a) Throughput versus average RTT. (b) Throughput versus wired latency.

variation in Y , which is the size of the wireless pipe, increases. As a result, the likelihood of an underflow increases, resulting in a decrease in throughput. This clearly illustrates the need for an ACK buffer so that a prompt feedback of the variation can be provided to the TCP source.

5.3 Throughput versus RTT

In this section, we investigate the trade-off between throughput and RTT. This trade-off is of interest because two of the algorithms (AR and WRB) hold back ACK packets to achieve higher throughput, resulting in potentially higher average RTT, whereas the other three do not.

For each buffer size (3 to 50), we measure the average packet RTT seen by the TCP sender across all users. Fig. 9a shows how RTT increases with the throughput (for four active users). The leftmost point of each line in Fig. 9a indicates the average RTT and throughput for a buffer size of three packets, which is the smallest buffer size simulated. As the buffer size increases, RTT increases correspondingly.

It can be seen that, in order to achieve the same throughput, AR incurs the largest average RTT value due to ACK buffering. Therefore, in terms of throughput-RTT trade-off, AR performs the worst. On the other hand, WRD performs the best, as the average RTT value achieved is the lowest for a given throughput for all the buffer sizes simulated. Finally, although WRB provides the highest throughput, it results in high-average RTT values, even with very small buffers. However, note that, for a long-lived TCP flow, large (per-packet) RTT values are acceptable as long as the throughput is high since high throughput will result in lower file transfer latency.

5.4 Throughput versus Wired Latency

In all previous measurements, the latency on the wired network is assumed to be small. In this section, we study

TABLE 1
RTT (in ms) Distribution from Hosts in the US to the World in February 2006

| Server Location | 25% | 50% | 75% | 95% | Num of Pairs |
|-----------------|-------|-------|-------|-------|--------------|
| U.S. | 37.6 | 50.1 | 75.6 | 93.4 | 239 |
| Europe | 124.4 | 141.0 | 169.8 | 200.1 | 68 |
| E. Asia | 115.9 | 184.1 | 193.9 | 229.0 | 29 |
| World | 37.6 | 81.0 | 160.5 | 330.6 | 413 |

the effect of a larger wired latency on the throughput by varying D . In order to obtain ranges of Internet RTTs, we look at recent Internet measurements done by the Internet End-to-end Performance Measurement (IEPM) project to monitor the end-to-end performance of Internet links (<http://www-iepm.slac.stanford.edu/monitoring/general.html>). A snapshot of the measurement results obtained by using ping is shown in Table 1 for a host residing in the US in February 2006. The 95th percentile of the measured round-trip delay is 93.4 ms when servers are located within the US. Partly due to much larger physical distance, for servers around the world, the 95th percentile of the measured round-trip delay is up to 330.6 ms. In this section, D , which is the constant wireline delay shown in Fig. 6, is varied from 1 to 500 ms, which is equivalent to varying the wired RTT from 2 ms to 1 second (the wireless link delay is independent). Thus, this range should easily cover the entire spectrum of wired latencies observed today. Note that the average wireless link latency is 200 ms (forward link with a mean of 125 ms and reverse link with a mean of 75 ms).

Fig. 9 shows how the throughput of the different algorithms vary with increasing wired latency for one user/flow. The performance of all algorithms are expected to drop as the wired latency increases since the TCP throughput is inversely proportional to RTT. For round-trip wireline latency less than 70 ms, WRS is better than DT. Beyond that point, since WRS is operating at the window-limited region and the TCP window remains fixed at 20 packets, the throughput of WRS degrades inversely with RTT and performs worse than DT, which can utilize a large window. AR performs very well for latency below 40 ms. For larger wired latencies, the estimation algorithm in AR becomes less accurate and buffer loss begins to occur more frequently. Beyond latency of 120 ms, the improvement of AR over DT is less than 10 percent, and beyond 200 ms, the throughput is very close to DT. WRD is fairly robust with respect to the increase in wired latency of up to 200 ms. One impact of larger latency on WRD and WRB is that more packets are being buffered in the wired network, increasing the chance of buffer underflow and lower throughput. With a round-trip latency of 200 ms, WRD is still 27 percent better than DT. However, the throughput degrades rapidly beyond latency of 200 ms. With latency larger than 400 ms, WRD has lower throughput than AR and DT. Again, this is due to the fact that WRD is operating at the window-limited region. The performance of WRB is similar to WRD.

To summarize, the performance of AR degrades rapidly at round-trip latency larger than 40 ms, but its performance

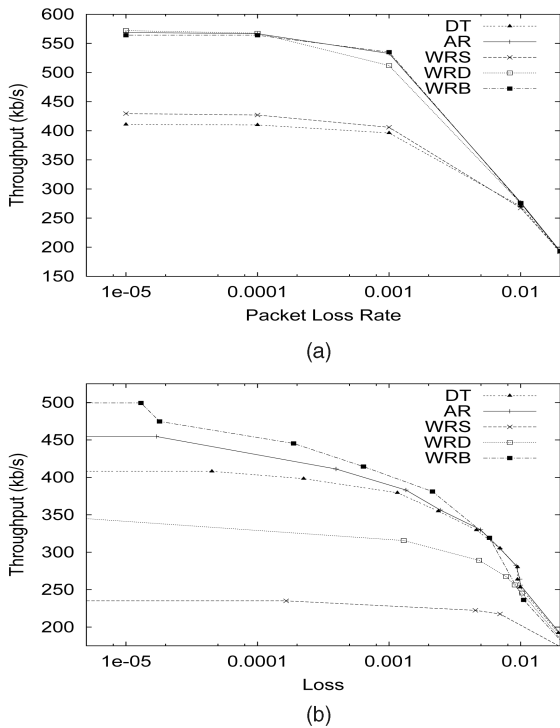


Fig. 10. Throughput versus loss. (a) Random loss. (b) Congestion loss.

is never worse than DT. On the other hand, both WRD and WRB perform well, with round-trip latency below 200 ms, but their performance degrades rapidly at larger latencies since they operate in the window-limited region. Beyond 400 ms RTT on the wired network, they perform worse than DT. Since the window sizes of all the three algorithms (AR, WRD, and WRB) are a function of the buffer size available in the RNC, increasing the RLP buffer size will allow these algorithms to maintain their high throughput for even larger wired latencies.

5.5 Impact of Random and Congestion Losses

In all the simulations so far, we have assumed zero loss in the rest of the network. In this section, we study the impact of loss in two different ways. First, we simulate the effect of random loss over the wireless link, and second, we simulate the effect of congestion loss in the wired network. In these experiments, we consider the case of a single mobile user $M1$ performing a download from source $S1$. Each simulation runs for 10,000 seconds.

The amount of random loss is varied in the link between the virtual node V and the mobile device $M1$ by using the random loss error module available in ns-2. The packet loss rate is varied from 10^{-5} to 10^{-2} . Fig. 10a shows how the throughput varies with packet loss. AR, WRB, and WRD continue to perform well for a very small amount of loss, but the throughput starts to decrease at loss rate of 10^{-3} . With loss rate 10^{-2} or greater, all algorithms have the same low performance since random loss is now the dominant factor determining the TCP throughput. Note that random loss can occur in the wireline network as well if RED is enabled.

In order to generate congestion loss, four FTP sessions using TCP SACK are generated from the cross traffic

sources in Fig. 6 to the cross traffic sinks. The bottleneck link is the link going from routers $N1$ to $N2$ and has a link bandwidth of 2 Mbps with delay of 1 ms. The packet buffer on router $N1$ is set to 100. Since the minimum RTT is 200 ms with a link speed of 2 Mbps and a 1,000-byte packet, 100 packet buffering is about two times the BDP. Different congestion conditions are simulated by varying the link bandwidth between the cross traffic sources and the router $N1$ from 300 to 500 Kbps. The impact of congestion loss on performance is shown in Fig. 10b. The loss rate plotted in the figure is the loss rate experienced only by the traffic going from $S1$ to $M1$.

A number of observations can be made. Both WRB and AR perform better than DT, with congestion loss rate below 10^{-3} , though the difference in performance decreases fairly rapidly from 10^{-5} to 10^{-3} . On the other hand, WRD performs poorly with respect to congestion loss and performs worse than DT for even a very small amount of loss. A similar result is true for WRS. The results for congestion loss, which is different from random loss, can be explained as follows.

During congestion buildup, the buffer in router $N1$, which can buffer up to 100 packets, increases the RTT by up to 400 ms in the worst case. With the increase in RTT, the throughput of WRD, as shown in Section 5.4, decreases rapidly. In fact, the throughput of WRD decreases to below 400 Kbps before any packet loss happens and is caused solely by the increase in wired latency due to congestion. The same is true for WRS, which performs even worse, as it operates with a smaller window. Interestingly, the performance of AR and WRD does not degrade as significantly. This is due to the fact that these schemes have an ACK buffer that can provide fast feedback. Recall that, for both AR and WRD, whenever the data queue (going toward the mobile device) is empty, the reset mechanism is enabled and more ACKs are released (see Section 4.4). This provides quick feedback to the TCP source, and high throughput is maintained even in the presence of congestion losses in the wired network.

5.6 Discussion

Similar to the AR presented in [13], the Window Regulator schemes cannot be used if the flow uses end-to-end IP Security (IPSEC). This is also true for all performance-enhancing proxies. However, we believe that proxies for performance improvement are critical in current wireless networks. In order to allow for these proxies without compromising security, a split security model can be adopted, where the RNC, under the control of the network provider, becomes a trusted element. In this model, a virtual private network (VPN) approach to security (say, using IPSEC) is used on the wireline network between the RNC and the correspondent host, and 3G authentication and link-layer encryption mechanisms are used between the RNC and the mobile host. This allows the RNC to support proxies such as the window or AR to improve performance without compromising security. Additional discussions on such issues can be found at the Internet Engineering Task Force (IETF) Performance Implications of Link Characteristics (PILC) Working Group Web site.

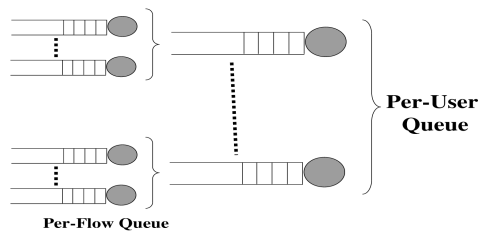


Fig. 11. Queuing structure of a scheduler.

6 SHORT-FLOW DIFFERENTIATION

In the previous section, the problem of improving the performance of long-lived TCP flows is addressed. However, it is well known that Internet traffic consists of a small number of long-lived flows that make up a large part (in bytes) of the total traffic and a large number of the flows (in counts) that are short lived. This is especially true with the popularity of applications such as Web browsing. As a result, optimizing the 3G wireless data system for short-lived TCP flows is also important. The main difference between the performance goals between long and short flows is that, in the former case, the goal is to maximize the throughput, and in the latter case, the performance goal is to minimize the average transfer latency.

In order to improve the performance of short flows, one needs to be able to identify them. Although it is possible to use the Type-of-Service (TOS) bits in the IP header to indicate the type of flows (for example, short or long), these bits are typically not used. Therefore, we assume that it is possible to classify flows at the TCP level by the four tuples in the TCP/IP header, namely, source IP address, destination IP address, source TCP port, and destination TCP port.

In this paper, we consider a two-level hierarchical queuing system, as shown in Fig. 11, where the first level consists of per-flow queue for a given user and the second level consists of a per-user queue. Within each first-level queue, an intrauser scheduler selects a packet to be sent first among all the flows of that respective user. At the second-level queue, a 3G scheduler selects a packet among different users to be sent over the wireless channel. For example, in HDR, the first-level scheduler is a First-in, First-out (FIFO) scheduler, and the second level scheduler is a PF queuing scheduler [6].

The rest of the section is organized as follows: First, we show that a significant improvement in short-flow latency is possible when a simple intrauser scheduler called Short-Flow Priority (SFP) is used instead of FIFO. Next, we study the behavior of three different interuser schedulers and explore the trade-off between fairness, throughput, and short-flow latency. We show that the proposed PF-RP scheduling algorithm is able to introduce elements of flow differentiation into the PF algorithm such that the short-flow latency can be reduced without sacrificing the system throughput.

6.1 Intrauser Scheduling

In this section, we first briefly describe the intrauser scheduler called SFP. SFP is conceptually similar to the schedulers proposed in [21] and [22], and the FB scheduler

in [25], with extensions for flow reclassification (to handle proxies or persistent connections) and the use of a strict-priority-based eviction policy (instead of RED). In SFP, only two classes are defined and strict priority is implemented between the classes. Therefore, if packets from the higher priority class are present, they will always be scheduled first. A flow is identified by the information in the packet header and a flow is classified as either a short or long flow by the amount of bytes sent so far by the scheduler. Initially, all flows are classified as short flows and a counter keeps track of the total number of bytes sent so far for each flow. When that counter increases beyond a predefined threshold, the flow is reclassified as a long flow. This simple reclassification scheme reduces the likelihood of starving long flows of the same user because SFP would eventually promote short flows to long flows. A flow is also reclassified from a long flow to a short flow if the flow is idle (no packet arrival) for a certain amount of time, called the *Reset Duration*. This reclassification has two advantages. First, for interactive applications like telnet, such resets will allow a telnet session to be classified as a short flow, even though the total amount in bytes of a telnet session is large. Second, for the case where Web traffic from a mobile terminates on a proxy or uses persistent connections (HTTP 1.1), it is important to reclassify the flow as short since the idle period likely indicates that the data belongs to a new Web download. A strict-priority buffer eviction is used where high-priority packets always evict lower priority packets if the buffer is full (except for the packet being served). We found that the use of RED/RED with in/out (RIO) schemes, as in [21] and [22], requires careful tuning of parameters, which can be difficult in a wireless environment. The use of a strict-priority eviction policy is simple and provides sufficient differentiation.

6.2 Interuser Schedulers

There are three parameters that can be taken into account by an interuser scheduler, namely, exploiting user diversity in order to improve the overall throughput, maintaining long-term fairness, and minimizing the short-flow latency. Flows to a given user are classified as being a long or short flow based on the first packet in the queue. We consider three different schedulers that take into account different aspects of the above three parameters.

6.2.1 PF Scheduler

PF is the scheduler used in the 3G EV-DO or HDR system [6]. As a standard algorithm used in data-optimized WWAN, PF is used as a baseline for comparison. In order to understand how PF works, we first need to understand the concept of user diversity, which is central to how PF improves the channel throughput. Consider the model where there are N active users sharing a wireless channel. The channel condition seen by each user varies independently. Better channel conditions translate into higher data rate, and vice versa. Each user continuously sends its measured channel condition back to the centralized PF scheduler, which resides at the base station. If the channel measurement feedback delay is relatively small compared to the channel rate variation, the scheduler has a good-enough estimate of all the users' channel condition

TABLE 2
PF-SP and PF with Rate Priority (PF-RP) Algorithms

| The PF-SP Algorithm | The PF-RP Algorithm |
|--|---|
| Let set of short flow user be S | Run the PF, let user i be selected |
| Let set of long flow user be L | Run the PF-SP, let user j be selected |
| if S is non empty | if $R_i > R_j$ OR $R_i < R_{min}$ |
| select user i with largest R_i | select user i |
| else if L is non empty | else |
| select user i with largest $\frac{R_i}{A_i}$ | select user j |
| Update A_i for all users | update A_i for all users |

when it schedules a packet to be transmitted to the user. Since channel condition varies independently among different users, user diversity can be exploited by selecting the user with the best condition to transmit in different time slots. This approach can increase the system throughput substantially compared to a round-robin scheduler. However, such a rate-maximizing scheme can be very unfair, and users with relatively bad channel conditions can be starved. Hence, the mechanism used in PF is to weight the current rate achievable by a user by the average rate received by a user.

At each time slot (every 1.67 ms in HDR), the decision of the PF scheduler is to schedule the user with the largest $\max_i \frac{R_i}{A_i}$, where R_i is the rate achievable by user i and A_i is the average rate of user i . The average rate is computed over a time window as a moving average:

$$\begin{aligned} A_i(t+1) &= (1 - \alpha)A_i(t) + \alpha R_i && \text{if scheduled,} \\ A_i(t+1) &= (1 - \alpha)A_i(t) && \text{if not scheduled.} \end{aligned}$$

PF does not differentiate between short and long flows among users.

6.2.2 PF-SP Scheduler

One possible way to improve the short-flow latency over the PF scheduler is to include a notion of priority in the PF scheduler so that users with short flows are given higher priority than users with long flows. We call this the *PF-SP* scheduler. PF-SP always prefers short flows to long flows. In PF-SP, we select the user with the highest instantaneous rate among users with short flows. When there are only long flows in the system, the default PF is run. The PF-SP is summarized in Table 2.

PF-SP gives strict priority to short flows across all users and, in a channel with no variation, can be expected to provide the lowest latency for short flows. In PF-SP, by selecting the user with the highest rate among all users with short flows, some user diversity is also exploited (limited to users with short flows only, instead of over all flows), and the short-flow latency is expected to be minimized at the

expense of fairness among users. The average rate for each user is maintained over all short and long flows.

6.2.3 PF-RP Scheduler

One of the problems of the PF-SP scheduler is that it always prefers a short flow to a long flow, independent of channel conditions. Short flows, by definition, cannot always be present in the queue, as long flows dominate in terms of byte count. As a result, the amount of diversity available to PF-SP could be reduced in comparison to PF.

We propose an algorithm called *PF-RP*, which attempts to strike a better balance between minimizing the short flow latency, exploiting user diversity, and providing fairness among users. In PF-RP, in each time slot, both PF and PF-SP are run logically. The selection from PF-SP is used if the user selected has a higher R_i than the user selected from PF; otherwise, the user selected from PF is used. This allows us to exploit diversity across all users (with both long and short flows) while retaining differentiation for short flows. The algorithm is summarized in Table 2.

Since PF is used, except in cases where using PF-SP improves the channel utilization, PF-RP has the property that it decreases short-flow latencies while, at the same time, increasing the overall throughput. However, the PF-RP algorithm sacrifices fairness to users with only long flows, but that is necessary, by definition, in any mechanism that provides differentiation to short flows. As in the case of PF-SP, the average rate for each user is maintained over all short and long flows and provides some measure of the overall fairness to users with little or no short flows. Nevertheless, since the goal of minimizing short-flow latency is taken over all users, it is possible that, in an overloaded system, a single user with only long flows can be starved. Hence, the minimum rate (R_{min}) is defined such that, if the average rate allocated is below R_{min} , the user chosen by PF will be selected instead.

6.3 Evaluation

We used the same ns-2 simulation setup as before, and the simulation time is 10,000 seconds. The parameters of the Web traffic model used are shown in Table 3. Flows are classified as short if the file size is below 15 Kbytes and the reset duration is set to 1 second. The averaging parameter α used in calculating A_i in the PF scheduler is set to 0.001.

In the first simulation, we compare the performance of intrauser and interuser scheduling with eight users. Out of these eight users, four users have only Web traffic, and the other four users have only a single long-lived FTP session. The results are shown in Fig. 12. In Fig. 12a, we see that the biggest improvement is for SFP over the FIFO buffering scheme. Due to selective dropping of packets with lower

TABLE 3
Traffic Model Parameters

| Web Model Elements | Attributes | Distribution and values |
|--------------------|--|--|
| Web Page | Time interval between Pages (s) Number of Web objects per pages | Exponential, mean = 15s (Figure 12, 14) or 300s (Figure 13) Exponential, mean=5 |
| Web Object | Time interval between Web objects (s) object size | Exponential, mean=0.01 Bounded Pareto, mean = 12, shape = 1.2, max=1000 KB |

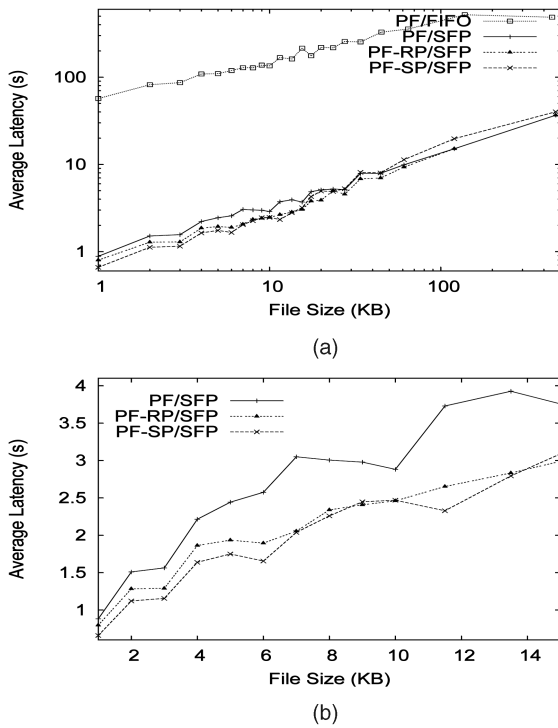


Fig. 12. Average transfer latency for eight users, with four users having only Web traffic and four users having only FTP traffic. (a) All files. (b) Small files below 20 Kbytes.

priority (longer flows), the mean latency is reduced by more than 50 times over all file sizes. Among PF, PF-SP, and PF-RP, for the smallest files, PF-SP performs the best, and PF the worst. For large files (> 100 Kbytes), PF-SP performs the worst (see the crossover in Fig. 12a) because short files are scheduled even when the channel condition is bad. PF and PF-RP have very similar performance, showing that the throughput gain for small files with PF-RP has a minimum impact on the overall throughput. Fig. 12b shows the result for small files, where the performance of PF, PF-SP, and PF-RP are compared for files up to 15 Kbytes. Overall, the result is as expected. Compared to PF, PF-SP improves short-flow latency (up to 15 Kbytes) from 14 percent to 38 percent. For the largest completed flows, the latency increased by 10 percent. For PF-RP, the short-flow latency decreases by 10 percent to 33 percent, and for the large transfer, the latency is the same as PF.

In the next simulation, we experiment with a different traffic mix with eight users, and each user has both Web traffic and a single long-lived FTP. In order to maintain similar overall load with the previous experiment, the Web traffic load used is reduced, as shown in Table 3. The result is shown in Fig. 13. Due to the existence of Web traffic for all eight users, the amount of user diversity available for short flows is higher. As a result, even for small flows, the performance of PF-RP is very close to PF-SP. Compared to PF, PF-SP improves the short-flow latency from 0 percent to 23 percent and increases the latency of the largest files by 12 percent. For PF-RP, the short-flow latency decreases by 7 percent to 33 percent, and for large files, the latency is decreased by 5 percent compared to PF.

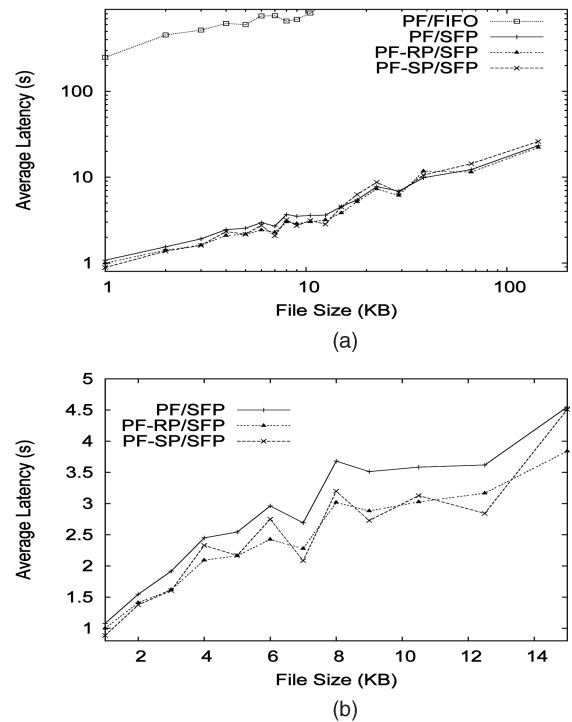


Fig. 13. Average transfer latency for eight users, each user with both FTP and traffic. (a) All files. (b) Small files below 20 Kbytes.

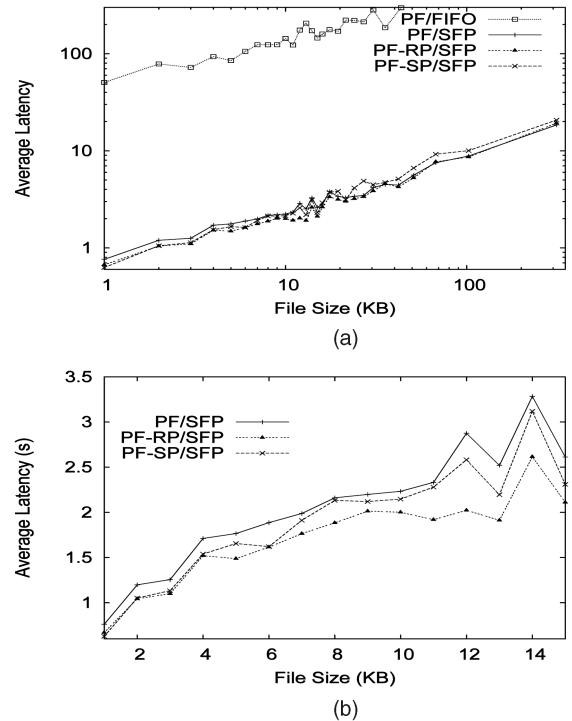


Fig. 14. Average transfer latency for four users, with one user having only Web Traffic and three users having only FTP traffic. (a) All files. (b) Small files below 20 Kbytes.

Surprisingly, in some cases, PF-RP is better than PF-SP for short flows and long flows. Fig. 14 shows such a case with four users. Only one user has Web traffic, and the other three users have a single FTP session each. In this case, for short flows, PF-RP outperforms PF-SP by up to 22 percent. For long flows, PF-RP is comparable to PF (both

TABLE 4
Number of Time Slots Allocated Per User

| | User 1 | User 2 | User 3 | User 4 | User 5 | User 6 | User 7 | User 8 |
|-----------|---------|---------|---------|---------|---------|---------|---------|---------|
| Figure 12 | Web | Web | Web | Web | FTP | FTP | FTP | FTP |
| PF | 123567 | 116764 | 122234 | 128413 | 813063 | 898671 | 893278 | 904010 |
| PF-RP | 128204 | 120279 | 124773 | 132237 | 789613 | 899592 | 898514 | 906788 |
| PF-SP | 158100 | 158409 | 157201 | 172396 | 790855 | 840723 | 859222 | 863094 |
| Figure 13 | Web+FTP | Web+FTP | Web+FTP | Web+FTP | Web+FTP | Web+FTP | Web+FTP | Web+FTP |
| PF | 565914 | 632239 | 633348 | 637268 | 637678 | 629559 | 630153 | 633841 |
| PF-RP | 560573 | 635088 | 634376 | 638709 | 637496 | 628799 | 630606 | 634353 |
| PF-SP | 573389 | 625425 | 634734 | 634764 | 633524 | 633147 | 627962 | 637055 |

of which are better than PF-SP). Note that, in Fig. 14, when there were seven FTP users instead of three, PF-RP could not outperform PF-SP for short flows. This can be explained as follows: First, with only one user with short flows, PF-SP cannot exploit user diversity. Second, with fewer (three) FTP users, the likelihood of PF-RP choosing a short flow with the best channel is higher compared to the case with more (seven) users. The combination of these two factors allows PF-RP to outperform PF-SP for short flows, and since PF-RP is wireless channel aware, it always outperforms PF-SP for long flows.

The fairness of PF is such that the PF scheduler allocates an equal amount of transmission time to all queues that are always backlogged. Due to differences in traffic load and TCP dynamics like slow start and congestion avoidance, the user queues will not always be backlogged, and the transmission times allocated will not be equal. However, by comparing the transmission time allocated to different users under PF, PF-SP, and PF-RP, we can obtain an indication of how the various interuser scheduling algorithms impact fairness by using PF as the benchmark. The transmission time allocation for the simulations shown in Figs. 12 and 13 are shown in Table 4. The results show that PF-RP is almost as fair as PF, whereas PF-SP is unfair to users with long flows.

In conclusion, we see that, across a wide range of traffic mix, PF-RP is the most robust scheduling algorithm, delivers high throughput and smaller flow completion latency, and results in fair scheduling time allocation.

7 CONCLUSION

In this paper, we have made two contributions. First, we proposed a network-based solution called the *Window Regulator* that maximizes the TCP performance in the presence of channel variations for any given buffer size at the congested router. We analyzed the performance of various versions of the Window Regulator schemes and make the following observations: WRS, a common algorithm used in wired routers, performs poorly. The WRB scheme, which explicitly adapts to the wireless channel conditions and also performs ACK regulation, improves the throughput by up to 100 percent over a DT scheme. WRB also delivers robust performance gains, even with reasonably large wired latencies and a small number of packet losses.

Next, we presented a scheduling and buffer sharing algorithm that reduces the latency for short flows while exploiting user diversity, thus allowing the wireless channel to be utilized efficiently. For intrauser scheduling, we found that having short-flow differentiation for scheduling and buffer management reduces the short-flow latency significantly and reduces the average normalized latency over a FIFO scheme by up to 50 times. For interuser scheduling, we found that the proposed PF-RP scheme is the most robust and provides good performance over a broad range of user traffic mix.

REFERENCES

- [1] TIA/EIA/cdma2000, "Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular Systems," Telecomm. Industry Assoc., 1999.
- [2] A. Bakre and B.R. Badrinath, "Handoff and System Support for Indirect TCP/IP," *Proc. Second Usenix Symp. Mobile and Location-Independent Computing*, Apr. 1995.
- [3] Third-Generation Partnership Project, "RLC Protocol Specification (3G TS 25.322)," 1999.
- [4] TIA/EIA/IS-707-A-2.10, "Data Service Options for Spread Spectrum Systems: Radio Link Protocol Type 3," Jan. 2000.
- [5] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. Tripathi, "Enhancing Throughput over Wireless LANs Using Channel State Dependent Packet Scheduling," *Proc. IEEE INFOCOM*, pp. 1133-1140, Mar. 1996.
- [6] P. Bender et al., "A Bandwidth Efficient High Speed Wireless Data Service for Nomadic Users," *IEEE Comm. Magazine*, July 2000.
- [7] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP Rate Control," *ACM Computer Comm. Rev.*, Jan. 2000.
- [8] H. Balakrishnan, S. Seshan, E. Amir, and R.H. Katz, "Improving TCP/IP Performance over Wireless Networks," *Proc. ACM MobiCom*, Nov. 1995.
- [9] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," *Proc. ACM MobiCom*, 1999.
- [10] R.K. Balan et al., "TCP HACK: TCP Header Checksum Option to Improve Performance over Lossy Links," *Proc. IEEE INFOCOM*, 2001.
- [11] K. Brown and S. Singh, "M-TCP: TCP for Mobile Cellular Networks," *ACM Computer Comm. Rev.*, vol. 27, no. 5, 1997.
- [12] T. Go, J. Moronski, D.S. Phatak, and V. Gupta, "Freeze-TCP: A True End-to-End Enhancement Mechanism for Mobile Environments," *Proc. IEEE INFOCOM*, 2000.
- [13] M.C. Chan and R. Ramjee, "TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation," *Proc. ACM MobiCom*, 2002.
- [14] H. Inamura et al., *TCP over 2.5G and 3G Wireless Networks*, IETF RFC 3481, Feb. 2003.
- [15] F. Khafizov and M. Yavuz, "TCP over CDMA2000 Networks," Internet draft, work in progress, Jan. 2002.
- [16] R. Ludwig, A. Konrad, and A.D. Joseph, "Optimizing the End-to-End Performance of Reliable Flows over Wireless Links," *Proc. ACM MobiCom*, 1999.

- [17] R. Ludwig and R.H. Katz, "The Eifel Algorithm: Making TCP Robust against Spurious Retransmissions," *ACM Computer Comm. Rev.*, vol. 30, no. 1, Jan. 2000.
- [18] P. Sarolahti, M. Kojo, and K. Raatikainen, "F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts," *ACM SIGCOMM Computer Comm. Rev.*, vol. 33, no. 2, Apr. 2003.
- [19] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, and I. Pratt, "Performance Optimizations for Wireless Wide-Area Networks: Comparative Study and Experimental Evaluation," *Proc. ACM MobiCom*, 2004.
- [20] L. Kalampoukas, A. Varma, and K.K. Ramakrishnan, "Explicit Window Adaptation: A Method to Enhance TCP Performance," *IEEE/ACM Trans. Networking*, June 2002.
- [21] X. Chen and J. Heidemann, "Preferential Treatment for Short Flows to Reduce Web Latency," *Computer Networks*, vol. 41, no. 6, pp. 779-794, Apr. 2003.
- [22] L. Guo and I. Matta, "The War Between Mice and Elephants," *Proc. Ninth Int'l Conf. Network Protocols (ICNP '01)*, 2001.
- [23] N.S. Joshi, S.R. Kadaba, S. Patel, and G.S. Sundaram, "Downlink Scheduling in CDMA Data Networks," *Proc. ACM MobiCom*, 2000.
- [24] Z. Shao and U. Madhow, "Scheduling Heavy-Tailed Data Traffic over the Wireless Internet," *Proc. 56th IEEE Vehicular Technology Conf. (VTC '02-Fall)*, 2002.
- [25] L. Kleinrock, "Queueing Systems," *Vol. II: Computer Applications*, Wiley, 1976.
- [26] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation," *Proc. ACM Ann. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '98)*, 1998.
- [27] V. Misra, W. Gong, and D. Towsley, "Stochastic Differential Equation Modeling and Analysis of TCP-Window Size Behavior," *Proc. Performance*, 1999.
- [28] E. Altman, K. Avrachenkov, and C. Barakat, "A Stochastic Model of TCP/IP with Stationary Random Loss," *Proc. ACM Ann. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '00)*, 2000.
- [29] F. Baccelli and D. Hong, "TCP Is Max-Plus Linear," *Proc. ACM Ann. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '00)*, 2000.
- [30] M. Gerla, B.K.F. Ng, M.Y. Sanadidi, M. Valla, and R. Wang, "TCP Westwood with Adaptive Bandwidth Estimation to Improve Efficiency/Friendliness Tradeoffs," *J. Computer Comm.*, vol. 27, no. 1, Jan. 2004.



Mun Choon Chan received the BS degree in electrical engineering from Purdue University, West Lafayette, Indiana, in 1990 and the MS and PhD degrees in electrical engineering from Columbia University, New York, in 1993 and 1997, respectively. From 1991 to 1997, he was a member of the coordinated multimedia explanation testbed (COMET) Research Group, working on asynchronous transfer mode (ATM) control and management. From 1997 to 2003, he was a member of the technical staff at the Networking Research Laboratory, Bell Laboratories, Lucent Technologies, Holmdel, New Jersey. He is currently an assistant professor in the Department of Computer Science, National University of Singapore. He has published more than 40 technical papers and is the holder of four patents. His current research interests include heterogeneous wireless networks and sensor networking. He is a member of the ACM and the IEEE.



Ram Ramjee received the BTech degree in computer science and engineering from the Indian Institute of Technology, Madras, and the MS and PhD degrees in computer science from the University of Massachusetts, Amherst. He is currently a senior researcher at Microsoft Research, India. Prior to that, he was with Bell Laboratories, Lucent Technologies, for 10 years, where he was a distinguished member of the technical staff and the technical manager of the Wireless Network Elements Research Department, leading a group of researchers examining architecture, protocol, and performance issues in next-generation networks. He has also served as an adjunct faculty in the Electrical Engineering Department, Columbia University, teaching two graduate courses in wireless networks. He served as the technical program cochair of ACM MobiCom 2006 and was a general cochair of the Second Annual International Wireless Internet Conference (WICON 2006). He is currently an associate editor for the *IEEE Transactions on Networking*. He has published more than 50 papers and is the holder of 13 patents. He is a fellow of the IEEE.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.