Baltzer Journals

Using Channel State Dependent Packet Scheduling to improve TCP throughput over wireless LANs

Pravin Bhagwat,[1] Partha Bhattacharya,[1] Arvind Krishna,[1] and Satish K. Tripathi[2] *

[1] *IBM, T.J. Watson Research Center*
Yorktown Heights, NY 10598
E-mail: {pravin,partha,krishna}@watson.ibm.com

[2] *Computer Science Department*
University of Maryland, College Park, MD 20742
E-mail: tripathi@cs.umd.edu

In recent years, a variety of mobile computers equipped with wireless communication devices have become popular. These computers use applications and protocols, originally developed for wired desktop hosts, to communicate over wireless channels. Unlike wired networks, packets transmitted on wireless channels are often subject to burst errors which cause back to back packet losses. In this paper we study the effect of burst packet errors and error recovery mechanisms employed in wireless MAC protocols on the performance of transport protocols such as TCP. Most wireless LAN link layer protocols recover from packet losses by retransmitting lost segments. When the wireless channel is in a burst error state, most retransmission attempts fail, thereby causing poor utilization of the wireless channel. Furthermore, in the event of multiple sessions sharing a wireless link, FIFO packet scheduling can cause the HOL blocking effect, resulting in unfair sharing of the bandwidth. This observation leads to a new class of packet dispatching methods which explicitly take wireless channel characteristics into consideration in making packet dispatching decisions. We compare a variety of channel state dependent packet (CSDP) scheduling methods with a view towards enhancing the performance of transport layer sessions. Our results indicate that by employing a CSDP scheduler at the wireless LAN device driver level, significant improvement in channel utilization can be achieved in typical wireless LAN configurations.

Keywords: Wireless Networks, Link Layer Protocols, Error Control, TCP

*This work was supported in part by NSF grant CCR 9318933

1 Introduction

Growing user demand for tetherless access to computing resources is likely to increase the deployment of Wireless LANs (WLANs) in the near future. The environment we mainly consider is that of a wireless LAN being used to gain access to a wired backbone network. A base station relays packets between wired and wireless links, and all mobile users gain access to the backbone network via a base station. From a network layer point of view all WLANs export a connectionless best-effort packet delivery service. However, as will be discussed later, this packet delivery service differs from that of wired networks in many aspects.

Hosts on the existing WLANs use applications and end-to-end transport protocols that were originally developed for wired networks, where the underlying physical links are fairly reliable and packet losses are random in nature. However, due to radio propagation characteristics, wireless links are significantly error prone. Within the ISM band[1] the error characteristics of wireless channels are bursty and time varying, e.g., due to multi-path fading. In addition, user mobility and frequent handoffs cause burst packet losses. Current generation WLANs operate within the ISM radio spectrum, with typical link rates within a few Mbps at a typical range of 50-100 meters. Under these conditions, the performance of transport protocols such as the Transmission Control Protocol (TCP) degrades significantly.

The performance problems of TCP over wireless channels have been well documented in the literature [1, 2, 3, 4] and a number of solutions have been proposed. Most of these solutions operate at the transport layer. Cacares et. al [2] proposed a method for fast recovery from losses during handoff. Their solution exploits TCP's fast retransmit option to reduce TCP's packet loss detection latency. In [1] a scheme is proposed that splits a TCP connection into two separate connections; one between the mobile host and the base station and the other between the base station and the stationary host. It attempts to isolate the dynamics of TCP's congestion control from the interference of losses on the wireless channel. The solution proposed in [5] achieves the same objective without actually splitting a TCP connection, but by keeping a log of unacknowledged TCP packets at the base station and retransmitting them to the mobile host.

In this paper, we also propose mechanisms to improve the performance of TCP in WLANs, but in contrast to the previous work, we advocate the use of link layer retransmission based mechanisms. The premise of our work is the observation that due to the coarse-grained timer (of granularity 500 msec) used in most TCP implementations, the loss detection latencies are high. This latency allows sufficient time at the link layer to recover from burst errors and link level

[1] Industrial, Scientific, and Medical band (902-928MHz, 2400-2483.5 MHz, and 5725-5850MHz) approved by the FCC for unlicensed use under certain restrictions.

retransmissions do not interfere with TCPs retransmission schemes.

Multiple TCP sessions usually share a wireless link when many mobile users access the backbone network via a base station. In this scenario we have observed that FIFO scheduling of packets in conjunction with a reliable link layer protocol leads to degraded overall throughput and unfair allocation of wireless bandwidth to the mobile users. This is because repeated transmission attempts of the head of line (HOL) packet fail due to channel burst errors, blocking the transmission of packets to other receivers. Since the wireless links to various destinations are statistically independent (as justified by earlier work, see section 2), packets for other destinations could have been successfully transmitted during this interval. None of the proposed solutions [1, 2, 5] address this problem.

We propose link level channel state dependent packet (CSDP) scheduling methods to overcome the problems associated with multiple sessions sharing the wireless link. For each mobile host, the base station maintains a separate queue of packets. Within each queue, packets are served in FIFO order. However, the service amongst queues depends on the specific policy employed; the policy being based upon the current estimated state of the channel between the mobile host and the base station. We demonstrate that appropriate packet transmission scheduling can increase overall throughput and reduce the unfairness problem, without interfering with TCP's retransmission schemes.

CSDP scheduling methods can be implemented at the base station's wireless device driver. Being a link layer solution, this has the advantage that not only TCP, but all UDP based applications such as `RPC`(Remote Procedure Call) and `NFS` [6] also benefit from it. Furthermore, this solution can be deployed in the access points that bridge wired and wireless LANs. Solutions that rely on a transport layer agent [1, 7] or require a soft transport layer state maintenance at the base station [5] cannot be easily used in this scenario. Finally, CSDP scheduling methods can also be used in conjunction with transport layer techniques for improving performance over the wireless channels [1, 2, 5]. Deploying a CSDP scheduler underneath the transport layer solutions will yield the benefits of both the approaches.

This paper is organized as follows. First, we briefly describe packet loss characteristics commonly observed in indoor wireless RF LANs. In section 3, we present some TCP traces collected over a wireless testbed to demonstrate TCP's reaction to burst packet losses. In section 4, we illustrate the HOL blocking effect that is observed when error-recovery is employed at the wireless link layer. In section 5, we describe the CSDP approach and show, through simulations, the interaction of First in First out (FIFO), Round Robin (RR), Earliest Timestamp First (ETF), and Largest Queue First (LQF) schedulers with the dynamics of multiple TCP sessions. We conclude by presenting a summary of our results and observations.

2 Wireless LAN Channel Characteristics

In the following, we briefly discuss various factors that give rise to burst losses on the wireless medium and later show how these errors degrade throughput and channel utilization.

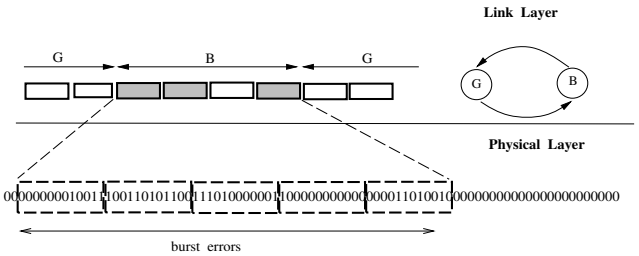

Figure 1: Error Characteristics

Error characteristics of the multi-access wireless channel differ significantly from that of the wired medium. Packet losses on the wired medium are very rare and random in nature. In contrast, the errors on the wireless medium are bursty and the wireless channel is distinct and time-varying for each mobile user. The distinction between the wired and wireless channels arises for many reasons. As users move, the received signal strength varies significantly. In addition, there are effects due to fading, interference from other users, and shadowing from objects, all of which degrade the channel performance [8]. Frequency-hopping WLANs are further subject to a unique error phenomenon. The WLAN may hop onto a frequency channel which is particularly susceptible to interference (e.g., a subset of the mobile users may be close to some other entity transmitting at that frequency). This phenomenon leads to bursty errors, since the effect is likely to disappear with the next change of frequency channel. Digital Cellular channels are also subject to the same problems for the same reasons. However, WLAN channels are more susceptible to losses because of the higher bit rates and lack of sophisticated signal processing techniques such as adaptive power control, forward error correction, and bit interleaving. Measurements of a particular WLAN system [9] show that packet-error rates critically depend on the distance between the transmitter and receiver, and surprisingly, are not monotonically increasing with this distance. Thus, the channel varies with each user, depending on their location with respect to the base station.

Many earlier studies in the literature [10, 11] have established that finite state Markovian models can be effectively used to characterize bit error patterns observed on RF channels. Each state of the Markov model is associated with a bit error rate (BER) value. The channel probabilistically switches between states with high and low BER values. Note that for most systems, a single bit error on

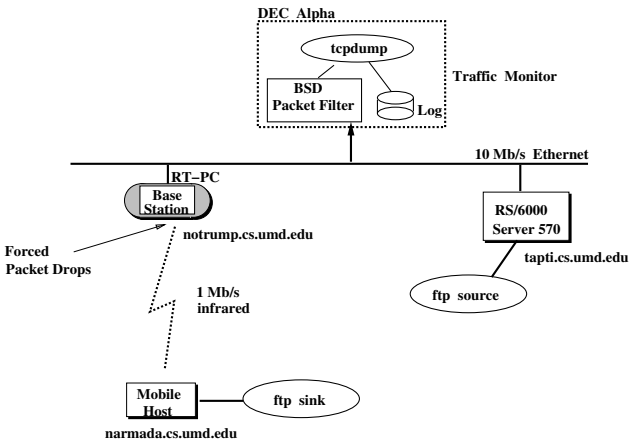

Figure 2: The Trace Collection Environment

the channel translates into the loss of a complete packet (see Figure 1 in which 0 indicates good reception and 1 indicates an error at the bit level).

In practice, we have observed that mean residency time in states with high BER is longer that a single packet transmission time. This results in correlated packet losses, i.e., a single packet loss would be followed by many back-to-back packet losses. We use this observation to capture *packet loss characteristics* by a 2-state Markov model. This model conforms with the observation reported in some of the earlier empirical studies of packet loss behavior on wireless LANs [9]. At any point in time, we model the channel as being in one of the two possible states, $G(\text{good})$ or $B(\text{burst})$. We use the term, p^S to denote packet loss probability in state S, where $S \in \{G, B\}$. We assume that $p^B \gg p^G$. Roughly speaking, a packet transmission is usually successful provided the channel stays in the good state during the packet transmission duration. We assume that the time spent in the Good and Bad periods are exponentially distributed, but with different average values. Note that our objective here is not to derive an accurate characterization of the channel. Rather, it is to illustrate the behavior of transport sessions when packets are subject to burst losses. An approximate characterization of the wireless channel is sufficient to illustrate these effects.

3 TCP's Reaction to Burst Packet Losses

Figure 2 shows the experimental setup that we used for collecting TCP traces. Figure 3 shows the trace of a single TCP connection whose packets were subject to burst losses. It shows a plot of sequence number versus time for an ftp

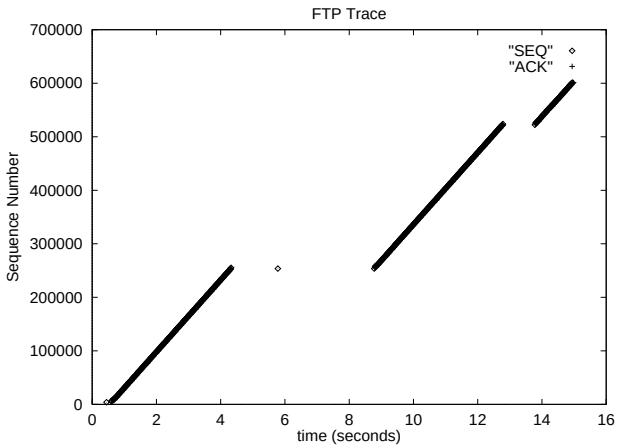

Figure 3: An ftp connection subject to burst losses

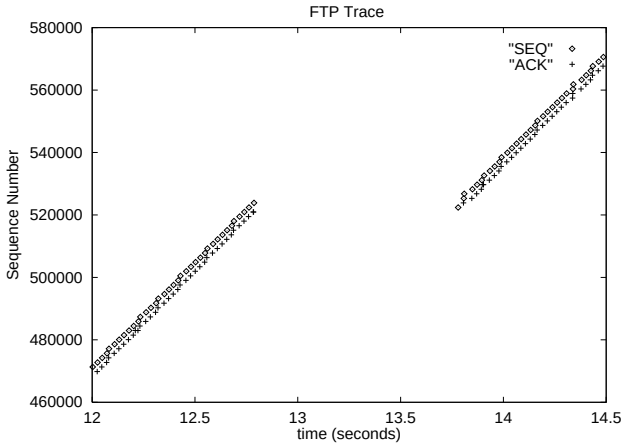

Figure 4: A microscopic view of the ftp trace, $12.0sec \leq t \leq 14.5sec$

connection between an IBM RS/6000 (tapti.cs.umd.edu) and an IBM RT-PC (narmada130.cs.umd.edu). The RS/6000 is connected to an ethernet while the RT-PC is connected to a 1Mb/s infrared wireless LAN. The ethernet and the wireless LAN are connected via another RT-PC machine notrump.cs.umd.edu. While the ftp was in progress, packets on the wireless segments were dropped using a 2-state loss model as described in the previous section[2]. The controlled dropping of packets was accomplished by modifying the wireless device driver at the base station. The trace was collected using tcpdump [12] running on a DEC alpha which is connected to the ethernet.

In the trace collected above, Round Trip Time (RTT) values are in the 20ms range, the receiver window size (W) is 4K, and the maximum segment size (mss) is 1466 bytes. The trace corresponds to a 600K file transfer which, under a no-loss situation, completes in approximately 10.5 seconds. During the course of the experiment the wireless channel switched into burst mode only 3 times (at $t = 4.3sec, t = 5.8sec$ and $t = 12.8sec$). Yet observed loss in throughput is about 40% (see Table 1). Below, we enumerate some of the problems that we observed during our experiments.

	No Losses	Burst Losses
Throughput	0.508 Mb/s	0.321 Mb/s

Table 1: Effect of burst errors on TCP throughput

Loss Detection Latency

Following the first burst error period, which occurs at $t = 4.3sec$, TCP waits approximately 1.48 seconds before it starts retransmission. Similarly, after the 3^{rd} burst error period, which occurs at $t = 12.8$ seconds, TCP takes 1.01 seconds to resume the *slow start* [13] phase (see Figure 4). In [2], a similar phenomenon was pointed out in the context of losses during handoff. A microscopic view of the ftp trace reveals that the time spent in the slow start phase is negligible compared to the latency of loss detection. This is because it only takes approximately RTT*$O(log_2(W/mss))$ time to restore the window size to its original value [14]. Over local area wireless connections, where round trip times are short and the window is small, this period is negligible compared to the latency of loss detection. For example, in figure 4, the duration of *slow start* phase is less than 40 milliseconds (from $t = 13.78sec$ to $t = 13.82sec$).

[2]mean burst period = 100 ms, mean good period = 5 sec, $p^B = 1$, $p^G = 0$

Failure of the Fast Retransmit Feature

The fast retransmit feature was added in TCP to reduce the latency of loss detection. This feature is triggered whenever a TCP source receives a threshold number (usually 3) of back to back duplicate acknowledgements. Arrival of triplicate ACKs is a good indication that a packet loss has occurred, and there is no need to wait for a timeout before retransmitting. The fast retransmit feature fails on the wireless medium because of two reasons. First, due to the low bit rate available on wireless channels, TCP window size is often chosen to be small (some times even less than 3 packets). Second, back to back packets and ACKs are both likely to be lost during fade periods implying that fast recovery will fail to trigger. Moreover in the case of interactive applications, a TCP source may not even have additional packets ready for transmission which would force the receiver to generate ACKs.

Exponential Retransmit Back-off

If retransmission occurs during a period when the channel is in a burst error state, the retransmit timeout value is doubled by Karn's [15] exponential retransmit back-off algorithm. For example, in Figure 3, the packet retransmitted at $t = 5.8sec$ falls in the burst error zone. Since the ACK for this packet is not received, TCP doubles its retransmit timeout period to 3 seconds and retries only after it expires. During this period the channel is not utilized.

Delayed Connection Setup & Denial of Service

At the connection setup time, TCP starts with an initial RTT estimate of 6 seconds (for lack of any better estimate). If the SYN packet is lost, the connection setup is delayed by 6 seconds. Loss of the 2^{nd} SYN packet further delays connection setup by an additional 24 seconds. The connection setup fails if the 3^{rd} SYN packet is also lost.

4 Loss Recovery at the Link Layer

A natural solution for the problems discussed in the last section lies in adding error recovery at the wireless link layer[3]. Although transport layer protocols are

[3]In [3], the authors point out that error recovery employed at the link layer could potentially interfere with TCP's timeout computation mechanism. This would be true if TCP used a very accurate clock to measure round trip times. In practice, however, TCP implementations use a very coarse timer (granularity of 500ms) to sample round trip time (RTT) values. The coarse grain timer makes TCP's adaptive RTT estimator insensitive to small fluctuations in RTT samples.

capable of recovering from packet losses on the wireless medium, as pointed out in the previous section, the latency of timeout based loss detection mechanisms is very high at the transport layer. Timeouts used in the link layer protocols, on the other hand, use a much smaller timeout value and, therefore, can recover faster from lost segments.

4.1 Adding Reliability at the Link Layer

Due to the high packet loss rate anticipated on the wireless medium, a need for explicit MAC layer acknowledgement for each data packet has been widely recognized within the IEEE 802.11 subcommittee. One of the recommendations [16] is to use CSMA/CA + priority ACK. The CSMA/CA part of the protocol is designed to reduce the collision probability between multiple stations accessing a medium. The priority ACK part of the protocol supports recovery from lost packets. To allow detection of a lost packet (due to collision, fading, or interference) an ACK is returned by the destination station immediately following a successful reception. ACK packets use a smaller carrier sense interval which gives them priority over access to the medium by all other stations which are waiting for the medium to become available. The ACK is transmitted by the receiving station only when the CRC of the received packet is found correct. If an ACK is not received immediately following a packet transmission, the source quickly times out and retransmits the lost packet after a random Retransmission-Backoff. Retransmission is either attempted by the wireless LAN card (if a retry function is supported in the card hardware) or by the link layer protocol (which is implemented as part of the device driver). The packet is eventually dropped if R_{max} successive packet transmission attempts fail.

4.2 FIFO Dispatching:

The current generation device drivers of WLAN cards maintain a FIFO queue of ready-to-transmit packets. In the transmit phase, the driver picks up the packet at the head of the queue, copies it into the WLAN adapter's on-board memory or sets up a DMA channel, and issues a transmit command to the card controller. The WLAN card issues a transmit complete interrupt to the CPU when transmission of this packet is complete. At this point the device driver can pick the next packet at the head of the queue and prepare it for transmission. The FIFO packet dispatching sequence works fine so long as the packet losses on the wireless medium are random. When losses on the wireless medium are bursty, this approach leads to many performance problems.

We study a simple network topology consisting of a single base station providing coverage to 10 mobile hosts. There are 10 FTP/TCP connections, each starting from a stationary host and terminating at a mobile host. The wireless

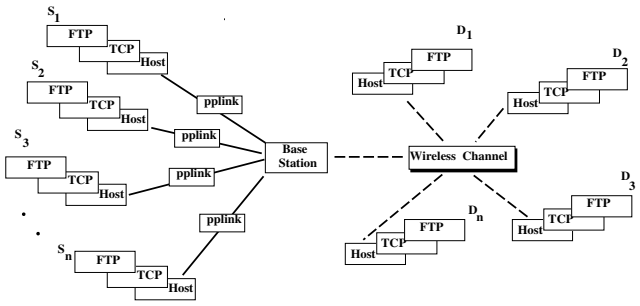

Figure 5: Simulation setup

link between the base station and mobile hosts has a bandwidth of 1Mb/s. This bandwidth is shared among all transmitters by the CSMA/CA + priority ACK protocol that has been implemented as a separate component in the simulator (see Figure 5). The transmission line between the source and the base station has a bandwidth of 10Mb/s and a propagation delay of τ.

Each FTP session involves a 0.8Mbyte file transfer. The propagation delay τ is set equal to 1.6ms which is representative of the delay in typical LAN environments. We expect the wireless base station to be the bottleneck for each TCP session. This is due to the fact that the wireless link is at least an order of magnitude slower than the speed of the wired segment. We assume sufficient amount of buffering at the base station, implying that buffer overflow is not a problem. This assumption is justified by two reasons. First, due to low bandwidth of the wireless medium we expect TCP sessions to operate with small window sizes. Secondly, the user population within each wireless segment is expected to be small.

Figure 6 shows a comparison of file transfer times for 10 TCP connections when FIFO and CSDP-RR schedulers (see section 5) are used at the base station. Table 2 shows a list of parameter values that were chosen for the simulation runs. The following observations became clear from these experiments:

1. Blocking Effect: Head-of-Line (HOL) blocking is observed because the wireless LANs constitute a single wireless channel which is shared on-demand among multiple receivers. When the HOL packet is destined to a receiver which is located in a fade zone, it prevents other packets in the FIFO queue from being transmitted (see Figure 7). Note that since channels to different receivers are statistically independent, some of the blocked packets destined to other receivers can be successfully transmitted during this time interval. Thus, if the channel to a specific receiver is in the burst error state, all receivers in that cell suffer throughput degradation.

2. Poor Utilization of the wireless medium: When the channel to a specific destination is in a burst error state, the CSMA/CA + priority MAC protocol repeatedly attempts to transmit a packet to its destination. During the burst error period, all

Topology parameters	pplink speed	$10Mb/s$
	pplink delay (τ)	$1.6ms$
	wireless channel	$1Mb/s$
	buffer size at base station	unlimited
Wireless channel parameters	mean good period	1 to 10 sec
	mean burst period	50 to 500 ms
	distribution of {good, burst} period	exponential
	loss probability in burst state (p^B)	0.8
	loss probability on good state (p^G)	0.0
	max retry count at the MAC layer (R_{max})	8
TCP parameters	packet size	512 bytes
	max window size	8K
	socket buffer size (per connection)	8K
	ftp size (file size)	800K
	number of simultaneous sessions	10
Base Station Scheduler	FIFO, RR CSDP-RR, CSDP-ETF, CSDP-LQF	

Table 2: Simulation Parameters

transmission attempts fail with a very high probability causing significant degradation in channel utilization.

3. RTT growth: A direct consequence of HOL blocking effect is seen on the end-to-end level, since the round trip time estimates of all the connections passing through the base station increase. This further increases the timeout values computed by the source TCP. HOL blocking also increases the latency observed by interactive applications.

4. Fairness Problem: Over a large time interval, the FIFO scheduler allocates access to the output link in proportion to the amount of traffic input by various sources. TCP's window flow control is a mechanism for bounding the amount of traffic queued up at the bottleneck node at any point of time. When all connections use identical window sizes, a FIFO scheduler provides fair access to the output link. This property no longer holds when errors are introduced on the output link. Since lost packets are transmitted multiple times, the share of the wireless channel received by different packet streams is a function of the channel loss characteristics. Under these conditions, FIFO scheduling fails to enforce any reasonable interpretation of fairness. See the completion time of various connections in Figure 6.

In summary, FIFO scheduling at the link layer yields poor end-to-end throughput. Interaction of various scheduling policies with the end-to-end flow/congestion control has been analyzed before in the literature [17]. However, these studies do

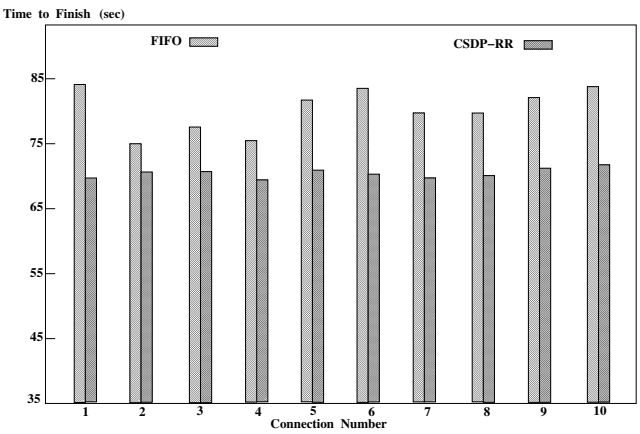

Figure 6: Comparison of FIFO and CSDP-RR, burst=100ms

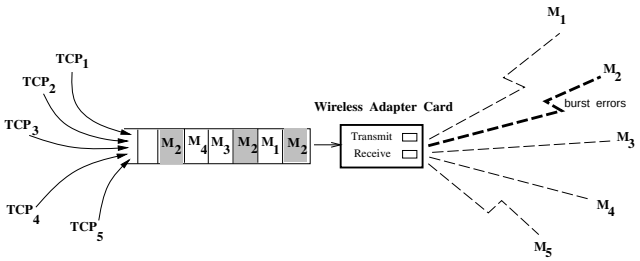

Figure 7: HOL blocking in FIFO scheduler

not consider packet losses on the output link. One of the important design considerations is to ensure that all sessions receive a fair share of the wireless bandwidth. When the medium is error free, a fair MAC protocol is sufficient to ensure this property. However, in the presence of burst errors, additional mechanisms are required. In the next section, we propose a class of packet scheduling policies which can be deployed at the base station wireless interface to increase the utilization of the wireless channel. These schedulers provide fairness while effectively shielding the end-to-end transport sessions from the effects of burst errors on the wireless medium.

5 Channel State Dependent Packet Scheduling

In this section we propose a class of device driver level packet scheduling policies that, unlike the FIFO scheme, takes the wireless channel state information into consideration. We refer to the class of such schedulers as Channel State Dependent

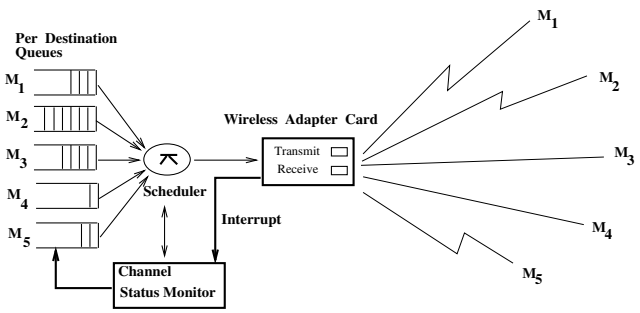

Figure 8: Channel State Dependent Packet Dispatching

Packet (CSDP) schedulers. Upon encountering a packet loss, these CSDP policies defer the transmissions to that destination until the start of the next good period for the wireless channel corresponding to that destination. Since the wireless channels to different destinations are statistically independent, packets for other destinations can be successfully transmitted during this interval.

It may seem that there is a potential risk associated with this approach. If the deferred period length is more than the TCP's timeout period, the source will timeout and retransmit a copy of the delayed packet, thereby unnecessarily increasing the load on the system. In practice, however, TCP's timeout period is on the order of seconds, while the duration of burst periods is on the order of milliseconds. This time difference is sufficient for link layer mechanisms to attempt loss recovery by retransmission over the wireless link.

5.1 CSDP Scheduling Mechanism

A CSDP scheduler has three components:

- a set of per destination queues
- a link status monitor
- a packet dispatcher.

For each mobile host in the coverage area, the CSDP scheduler at the base station maintains a separate queue of packets (see Figure 8). Within each queue, packets are served in FIFO order. However, service across different destination queues depends on the specific policy employed within the packet dispatcher. The link status monitor (LSM) is responsible for monitoring the state of channels between the base station and each mobile host. Whenever the channel between the base and the mobile host is in burst error mode, LSM marks the queue for that mobile host. The channel is assumed to be in burst error state whenever the

MAC layer ACK is not received following a packet transmission on that channel. The queue is unmarked after an estimated burst period length.

In general, loss of a MAC layer ACK may not necessarily imply an ongoing burst error period. Collisions may also lead to loss of ACKs. The physical layer can help the MAC layer and the LSM to distinguish between these two cases by sending detailed packet reception status. For example, since the physical layer manages the CRC, it should be able to inform the MAC layer about CRC failures. CRC failures are typically caused by collisions. In addition, the physical layer should be able to inform the MAC layer if a packet is received out of range (for example if power reception is below a certain threshold). It is important to note that this information is only available at the receiver end. However, this information can be piggy-backed with ACKs to notify the transmitter as well.

A CSDP scheduler operates by choosing an HOL packet from one of the unmarked queues. If all unmarked queues are empty, then it transmits a packet from one of the marked queues. If an ACK is not received within a short timeout period, CSDP returns the packet to the head of that queue, increments a counter which records the number of packet transmission attempts, and selects another packet according to the scheduling policy. If the total number of retries exceeds R_{max}, then the packet is dropped and its transmission is never attempted again. Notice that CSDP mechanism avoids retransmitting the lost packet immediately following an errored transmission. Thus, in the presence of burst errors, HOL blocking is significantly reduced. Overall, this mechanism yields better wireless link utilization (as results in the next section show) at a marginal cost of software complexity. The pseudocode of the CSDP scheduler is shown in Algorithm 1. The notations are described in Table 3.

$S_i(t)$	the status of the channel between host i and the base station at time t. $S_i(t) \in \{G, B\}$
$A(t)$	Set of destinations such that $\forall i \in A(t), S_i(t) = G$
$P(t)$	Set of destinations such that $\forall i \in P(t), S_i(t) = B$
π	Scheduling policy. $\pi \in \{FIFO, RR, LQF, ETF\}$
Q_i	FIFO queue of packets ready-to-be transmitted to destination i
$\xi_{bi}(t)$	residual period in state B, of the channel between host i and the base station, at time t

Table 3: CSDP Scheduler Notations

Algorithm 1

```
Receive(packet P)
    j ← destination of P
    Enqueue(P, Q_j)
    if (TxFree)
    Schedule Transmit event.

Transmit(t)
    if (Q_A(t) ∪ Q_P(t) = ∅) then {
        Txfree ← TRUE ;
        return;
        }
    if (Q_A(t) ≠ ∅) then
        j ← π(A(t)) ;
    else
        j ← π(P(t));
    P ← Dequeue( Q_j );
    transmit P;
    if (ACK not received within timeout) {
        enqueue(P, Q_j)
        S_j(t) ← B
        A(t) ← A(t) − j
        P(t) ← P(t) ∪ j
        Schedule event LinkToggle(j, t)
        event at t ← t + ξ_bi(t);
    }
    Schedule next Transmit Event;

LinkToggle(j, t)
    S_j(t) ← G
    A(t) ← A(t) ∪ j
    P(t) ← P(t) − j
```

5.2 CSDP Implementation Complexity

A CSDP device driver is required to manage per destination queues, which are accessed and updated whenever either packets are received from the network layer or an interrupt is issued by a WLAN card (see Figure 9). At any point in time, the number of queues is equal to the number of active stations within a single base station coverage area. In typical WLAN environments, this number is expected to be on the order of 10 to 15, implying that the overhead of managing per destination

queues is negligible. Compared to traditional device drivers, a CSDP device driver requires more data copying operations. This is because following each packet loss event a CSDP device driver dispatches a new packet. This requires a single `bcopy` operation since the contents of the transmit buffer can be overwritten. Table 4 shows the memory to memory copy latency on a 33Mhz, 486DX processor based machine with an ISA bus[4]. Since memory to memory copy latency is less than the carrier sense interval (which is on the order of $50\mu s$) it is possible to operate the CSDP device driver without compromising throughput.

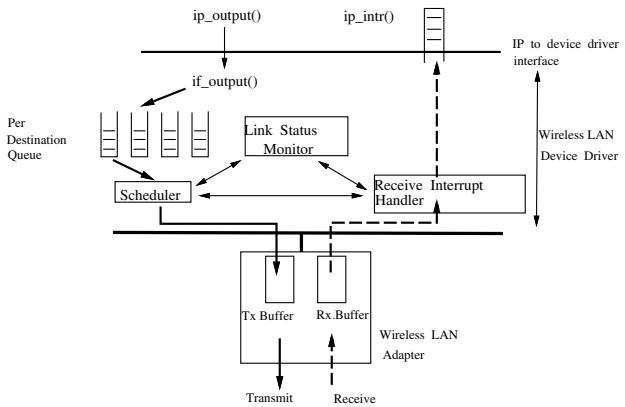

Figure 9: Implementation of per destination Queuing

packet Size	128	256	512	1024
Memory Copy Latency (μs)	5.6	8.4	14.2	26.9

Table 4: Memory to memory copy latency

5.3 CSDP Scheduling Policies

The CSDP scheduling framework lends an easy implementation of a variety of *work conserving* scheduling policies. Since a separate queue is maintained for each destination, it is possible to enforce QoS and fairness constraints on a per-host basis by suitable choice of a scheduling policy. Since the algorithm will be executed at the device driver level, it is important that it be simple so that the required processing can be carried out in real-time. Our results indicate that complex scheduling functions do not always yield good results and, interestingly,

[4]For a machine with a faster processor or a bus the memory copy latency is even lower.

as will be shown in the results section, some of the seemingly good techniques exhibit anomalous behavior.

We experimented with CSDP versions of three well known scheduling methods: Round Robin (RR), Earliest Timestamp First (ETF), and Longest Queue First (LQF).

CSDP-RR Set of unmarked and marked queues are maintained as two separate circular lists. Each set is served in a round robin fashion. Packets are first chosen from the unmarked set. If this set is empty, the scheduler serves packets from the marked set. Notice that a non-CSDP scheduler will continue to retransmit the HOL packet until it succeeds or the maximum retry counter expires.

CSDP-ETF Packets arriving from the wired interface are first timestamped, and then added to the respective queues. From the unmarked set, if it is non-empty, the packet with the earliest timestamp is chosen for service. Otherwise, the packet with the earliest timestamp from the marked set of queues is selected. This policy is very similar to FIFO, except that it tries to avoid transmitting packets to destinations that are known to be in errored state. FIFO can be viewed as a special case of CSDP-ETF when no feedback from the LSM is available.

CSDP-LQF CSDP-LQF scheduler always tries to minimize maximum backlog by always chosing the unmarked queue with the largest backlog. In [18] it is proved that if errors on the wireless channels are random and packet arrivals follow a Poisson distribution, then the LQF policy maximizes channel utilization among all policies.

6 Numerical Results

The interaction between TCP's flow control and the link layer dispatching mechanism is quite complex. Since the behavior of TCP sources cannot be captured by any closed form analytical expression, it is not possible to analyze this system mathematically. To perform a careful investigation of this interaction we decided to conduct some experiments on the NETSIM [19] simulator. Each simulation run involves data transfer over 10 simultaneous TCP sessions, each starting from a stationary host and terminating at a mobile host. During the course of data transfer, channels between the base station and mobile hosts are independently subject to burst errors. Using the same seed for the random number generator, we repeat this experiment with different CSDP policies. A fixed seed ensures that the same error pattern is repeated in each experiment. It allows us to directly compare results from different experiments. Under each policy π, we measure the

statistics for two kinds of TCP sessions, batch and interactive. For batch sessions we measure the time it takes to complete each TCP session, and for interactive sessions we measure the delay seen by each packet. We use the following three criteria for comparing the policies:

1. Maximum Transfer Time:

   T^{π}_{max} denotes the time when the last batch (FTP) session completes under scheduling policy π. When there are N such sessions,

   $$T^{\pi}_{max} = \max\{T^{\pi}_1, \ldots, T^{\pi}_N\}.$$

   A good scheduling policy should attempt to minimize this value.

2. Fairness: For identical TCP sessions, the difference between the latest and earliest finish times of the TCP sessions, namely $T^{\pi}_{max} - T^{\pi}_{min}$, provides a measure of fairness. T^{π}_{min} denotes the time when the earliest batch TCP (FTP) session completes under scheduling policy π. When there are N such sessions,

   $$T^{\pi}_{min} = \min\{T^{\pi}_1, \ldots, T^{\pi}_N\}.$$

   Since the TCP sessions are identical and the wireless channel characteristics are statistically identical, the sessions should complete at about the same time, that is, $T^{\pi}_{max} - T^{\pi}_{min}$ should be small. A policy π_1 is said to be more fair than π_2 if $(T^{\pi_1}_{max} - T^{\pi_1}_{min}) < (T^{\pi_2}_{max} - T^{\pi_2}_{min})$.

3. Per packet delay: For supporting interactive applications, it is crucial to minimize both the average and the maximum delay.

A desirable scheduler is the one which minimizes the maximum transfer time, fairly distributes the bandwidth among all active batch sessions, and minimizes delay for interactive sessions. First we compare the performance of FIFO and CSDP schedulers using the first two criteria. This is done by measuring the completion time of 10 identical FTP sessions. Later, we experiment with a mix of interactive and bulk transfer sessions to observe the delay behavior of various schedulers.

6.1 CSDP with Perfect Channel Knowledge

When a MAC level ACK is lost and the physical hardware notifies that it is not a CRC error, the scheduler knows that the wireless channel is in a burst state and the queue to that destination gets marked. To determine when the queue will be unmarked and packets from that queue will be transmitted, an estimator for the residual burst period is needed. To probe the destination would defeat the purpose of CSDP scheduling. We first consider the case in which the Link Status

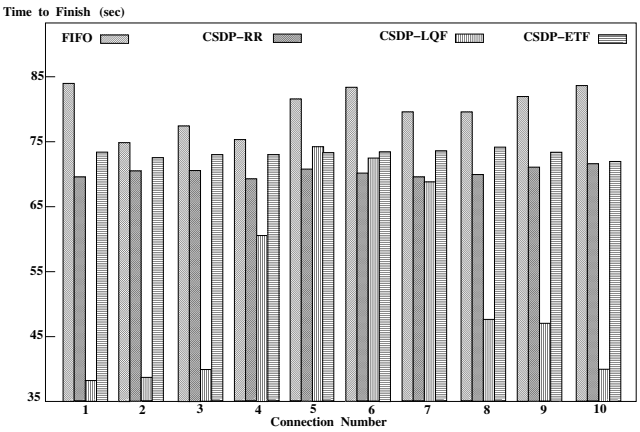

Figure 10: FIFO and CSDP comparison, burst=100ms

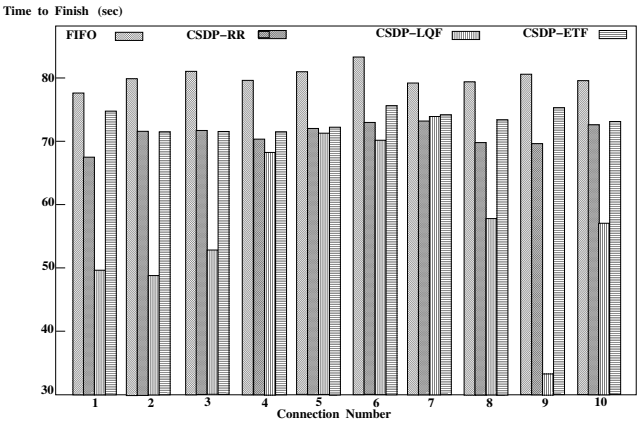

Figure 11: FIFO and CSDP comparison, burst=500ms

Monitor has perfect knowledge of the duration of the residual burst period of the wireless channels. Therefore, the time that the LSM unmarks a queue is in fact, the earliest time that the corresponding channel entered a good state. Results for this case provide upper bounds on the achievable improvement for realistic channel estimator based CSDP schemes.

Identical TCP sessions

Figure 10 shows file transfer times for 10 identical FTP/TCP sessions under various policies. For these experiments, the chosen TCP and the wireless channel parameters are shown in Table 2. Our results indicate that:

- CSDP schedulers provide lower transfer time than FIFO since

$$\max\{T_{max}^{CSDP-RR}, T_{max}^{CSDP-ETF}, T_{max}^{CSDP-LQF}\} < T_{max}^{FIFO}.$$

- Compared to FIFO, sessions under CSDP-RR and CSDP-ETF receive fair service since

$$\max\{T_{max}^{CSDP-RR} - T_{min}^{CSDP-RR}, T_{max}^{CSDP-ETF} - T_{min}^{CSDP-ETF}\} < T_{max}^{FIFO} - T_{min}^{FIFO}.$$

  Observe that CSDP-LQF exhibits a large variation in finish times, even amongst identical TCP sessions. The impact of this variation will be shown in the next section, through the poor performance of interactive TCP sessions under CSDP-LQF.

CSDP policies result in 10-15% improvement in maximum transfer time over FIFO. Figures 10 and 11 are representative of typical simulation runs. We have observed similar performance improvements for the range of parameters shown in Table 2. This can be attributed to facts that HOL blocking is avoided and that the deferral of link layer retransmissions does not interfere with TCP's retransmissions schemes.

Of the three CSDP policies CSDP-RR has lower maximum transfer time than CSDP-ETF and CSDP-LQF, although the difference is not significant. However, CSDP-RR exhibits the least variation in finish times. This can be explained as follows. As CSDP-LQF attempts to first serve the queue with the maximum number of packets, the corresponding TCP source receives a bigger share of the bottleneck wireless link. It keeps expanding its window size while packets in other queues wait for their turn for transmission. As a result some connections 'win' in the beginning while others are starved until the favored connections complete or suffer burst errors. This causes sessions to be served in batches under CSDP-LQF, resulting in lower average transfer time. In contrast, CSDP-RR attempts to evenly distribute the available bandwidth among all active sessions causing all sessions to finish at about the same time.

Overall, CSDP-RR scheduler outperforms all other policies, both in terms of performance and implementation complexity. It has been shown [18] that CSDP-LQF maximizes throughput for a similar system under Poisson sources and a random error model. It is interesting to note that this result does not apply in our case since the input traffic here is not Poisson and, in fact, is deeply correlated to packet delays. Also, CSDP-LQF is undesirable when we have a mix of interactive and bulk transfer traffic in the system. Since LQF gives higher priority to queues with more backlog, the interactive traffic suffers unbounded delays. This phenomenon is illustrated in the next section.

A mix of interactive and non-interactive sessions

In the setting of the previous experiment, we now consider a mix of interactive and bulk transfer traffic. Session number 1 generates interactive traffic while the remaining 9 sessions continuously generate FTP traffic. The interactive source generates a new packet only after the previous packet has been successfully received and acknowledged. We model it using a 5K size file transfer with window size set to one packet. Figure 12 shows the delay seen by each packet of the interactive session and Figure 13 shows the average and the maximum of this delay distribution.

Interactive packets encounter much larger delays under CSDP-LQF than under CSDP-RR. The reason is that the interactive session always operates with a window size of one, while the FTP sessions attempt to maximize their window sizes. As a result, an interactive packet typically encounters many packets from each one of the FTP sessions at the base station. Since CSDP-LQF tends to give preference to FTP sessions, the interactive packets are held up at the base station causing TCP to timeout and double its retransmit timeout value. After a point, either the queue grows large enough or FTP sessions hit burst errors, so that the interactive session receives service. This cycle repeats over and over again, causing interactive packet delay to grow exponentially under CSDP-LQF, as evident by the linear slope on the logarithmic delay plot in Figure 12. Observe also that in contrast to CSDP-LQF, CSDP-ETF provides bounded delays for interactive traffic. The delay under CSDP-ETF is proportional to the number of active sessions and their respective window sizes, while that under CSDP-RR is only proportional to the number of active sessions.

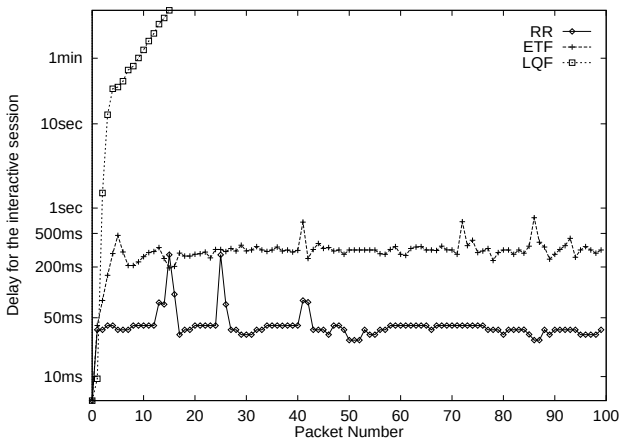

Figure 12: Packet number vs delay plot

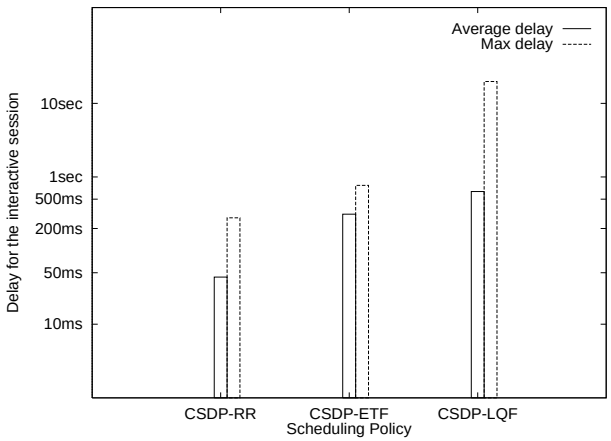

Figure 13: Average and maximum delay per packet

6.2 CSDP with Imperfect Channel Knowledge

We now consider a more realistic setting in which the LSM is not aware of when the burst period ends. To demonstrate the effectiveness of the CSDP approach in this case, we operated the CSDP-RR scheduler without any feedback from the Link Status Monitor. We refer to this scheduler as Zero Channel Knowledge Round Robin (ZCK-RR) scheduler. ZCK-RR serves all queues in round robin sequence without differentiating between marked and unmarked destinations. Following a packet loss, it simply defers retransmission of the lost packet until the next round. The deferred period can be thought of as a simple channel estimator, and, as our results show (in Figure 14), this simple estimator also works reasonably compared to FIFO, but not quite as well compared to CSDP-RR.

The above comparison of FIFO and ZCK-RR policies indicates that even in cases when channel errors are unpredictable CSDP schedulers will outperform FIFO schedulers. For example, if errors on the wireless channel are random, the link status monitor will mark all destinations as in "burst" state. In this case, the operation of a CSDP-RR scheduler will be identical to a ZCK-RR scheduler. Thus, CSDP-RR will outperform FIFO, regardless of what the error characteristics are and whether a good channel estimator is available or not.

The construction of good channel state estimators for general channel error processes is a difficult problem, but for some special channel error models, simple estimators can be provided. For example, if the error process is an on-off process with arbitrary good and burst period distributions, renewal theory can be used to determine the steady state distribution of the residual burst period at any time t. From this distribution, the mean and the variance of the residual burst period can be computed and a simple formula *mean + 2 * variance* can be used

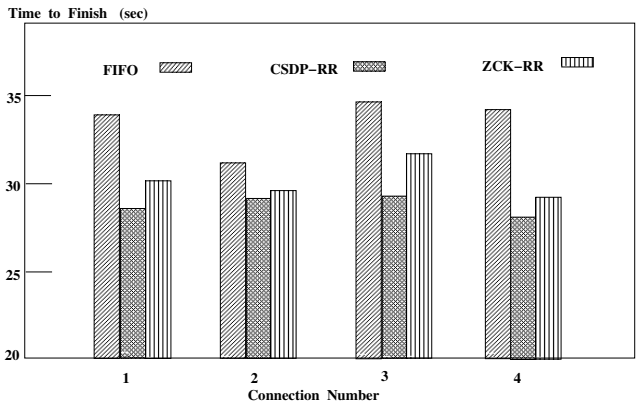

Figure 14: FIFO and ZCK-RR comparison, burst = 100ms

as an estimator. Since the mean duration of a good period is at least an order of magnitude larger than that of a burst period, it is unlikely that minor estimation errors will have a significant effect on the overall link utilization. Our results have demonstrated the benefit of using channel information. The design of better and more effective estimators for general error processes is an area of further study. A summary of our results and a qualitative comparison of various CSDP scheduling methods is shown in Table 5.

Policy	FIFO	RR	LQF	ETF
Transfer time	worst	best		
Fairness		best	worst	
Throughput	worst	best		
Delay for interactive traffic		best	worst	
Implementation Complexity	$O(1)$	$O(1)$	$O(n)$	$O(n)$

Table 5: Comparison of CSDP schedulers

7 Conclusion

Link layer error recovery mechanisms in current wireless LANs, such as those in the forthcoming IEEE 802.11 standards, are effective only in the presence of random errors. Recent performance studies [9] and our experience with RF wireless LANs have shown that error behavior on wireless channels is often bursty. When

multiple TCP sessions share a wireless link encountering burst errors, we have observed that FIFO packet scheduling at the base station leads to poor wireless channel utilization and unfair bandwidth allocation among the various sessions. We have proposed channel state dependent packet (CSDP) scheduling methods to overcome these problems.

A desirable scheduler is the one that minimizes maximum session completion times, fairly distributes bandwidth among all active sessions and minimizes delay for interactive sessions. With these requirements in mind, we have compared CSDP-RR, CSDP-ETF, CSDP-LQF and FIFO schedulers. The degree of performance benefit achievable by CSDP schedulers depends on the accuracy of the channel state predictor. Under the assumption of full channel knowledge, we have observed up to 15% improvement in maximum session completion times for CSDP schedulers. Among the CSDP schedulers, CSDP-RR outperforms the two other policies, both in terms of performance and implementation complexity. It minimizes the maximum session completion time and allocates bandwidth evenly among active sessions, thereby also minimizing the delay for interactive traffic. CSDP-ETF and CSDP-LQF provides comparable maximum session completion times as long as all the sessions are similar (e.g. all FTP sessions). However in the presence of ongoing bulk transfer traffic, interactive traffic suffers large delays under CSDP-ETF and CSDP-LQF policies. Even in the absence of channel state information, we have shown that the round robin version of the CSDP scheduler provides significant performance improvement over a pure FIFO dispatcher.

Further work is required to completely understand the interplay of link level error recovery and scheduling with the transport level mechanisms. Our initial results, however, show promise for these techniques with possibly reduced complexity at the base stations as compared to higher level techniques. In the future, the interaction of heterogeneous users multiplexed on a single wireless link, as well as the impacts of real time traffic should be studied.

Acknowledgements

Many thanks are due to Cynthia Davis who provided TCP traces for the experiments described in Section 3. Also, we would like to thank the reviewers for suggesting improvements to the presentation of the results.

References

[1] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in *Proceedings of the International Conference on Distributed Computing Systems*, (Vancouver, Canada), IEEE Press, 1995.

[2] R. Caceres and L. Lftode, "The effects of mobility on reliable transport protocols," in *Proceedings of the International Conference on Distributed Computing Systems*, (Poznan, Poland), IEEE Press, June 1994, pp. 12–20 .

[3] A. DeSimone, M. C. Chuah, and O.-C. Yue, "Throughput performance of transport-layer protocols over wireless lans," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, IEEE Press, 1993, pp. 542–549.

[4] P. Manzoni, D. Ghoshal, and G. Serazzi, "Impact of mobility on TCP/IP: an integrated performance study," IEEE Journal on Selected Areas in Communications, 13(5)(1995), 858–867.

[5] E. Amir, H. Balakrishnan, S. Seshan, and R. Katz, "Efficient TCP over networks with wireless links," in *The First International ACM Conference on Mobile Computing and Networking 1995*, (Berkeley, CA), Dec. 1995.

[6] Dan Walsh et. al., "The overview of the sun network file system," in *Proceedings of Winter USENIX*, (Dallas, TX), Jan 1985, pp. 117–124.

[7] R. Yavatkar and N. Bhagwat, "Improving end-to-end performance of TCP over mobile internetworks," in *Workshop on Mobile Computing Systems and Applications*, (Santa Cruz, CA), IEEE Press, Dec. 1994, pp. 146–152.

[8] W. Jakes, *Microwave Mobile Communications*. Piscataway, NJ: IEEE Press, 1994. Reissue of 1974 edition.

[9] D. Duchamp and N. Reynolds, "Measured performance of a wireless lan," in *17th Conference on Local Computer Networks*, (Minneapolis, MN), IEEE Press, Sept. 1992,pp. 494–499.

[10] F. Swarts and H. Ferreira, "Markov characterization of digital fading mobile VHF channels," IEEE Trans. Vehic. Tech., Nov. 1994, 977–985.

[11] H. Wang and N. Moayeri, "Finite state markov channel – a useful model for radio communication channels," IEEE Trans. Vehic. Tech., Feb. 1995, pp. 163–171 .

[12] S. McCanne and V. Jacobson, "The BSD packet filter: a new architecture for user-level packet capture," in *1993 WInter Usenix Conference*, Usenix, Jan. 1993.

[13] V. Jacobson, "Congestion avoidance and control," ACM Computer Communication Review, 18, Aug. 1988, 314–329. *Proceedings of the Sigcomm '88 Symposium.* Stanford, CA, August, 1988.

[14] T. V. Laksman and U. Madhow, "Performance analysis of window-based flow control using TCP/IP: the effect of high bandwidth-delay products and random loss," in *Performance 94*, 1994.

[15] P. Karn and C. Partridge, "Improving round trip time estimates in reliable transport protocols," ACM Transactions on Computer Systems, 9, Nov. 1991, 364–373 .

[16] W. Diepstraten, "IEEE 802.11: Wireless Access Method and Physical Specification," May 1993. Doc:IEEE P802.11-93/70.

[17] S. Shenker, L. Zhang, and D. Clark, "Some observations on the dynamics of a congestion control algorithm," ACM Computer Communication Review, Oct. 1990, 30–39 .

[18] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," IEEE Transactions on Information Theory, 39, Mar. 1993, 466.

[19] A. Heybey, "NETSIM: The Network Simulator," Laboratory of Computer Science, Massachusetts Institute of Technology. Sept 1990.