Enhancing throughput over wireless LANs using Channel State Dependent Packet Scheduling*

P. Bhagwat† P. Bhattacharya‡ A. Krishna‡ S. Tripathi†

†Computer Science Department
University of Maryland
College Park, MD 20742

‡IBM, T.J. Watson Research Center
Hawthorne, NY 10562

Abstract

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 performance of transport protocols such as TCP. Most wireless LAN link layer protocols recover from packet losses by retransmitting lost segments. When wireless channel is in 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 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.

*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 in the near future. Current generation Wireless LANs (WLANs) operate within the ISM radio spectrum, with a typical range of 50-100 meters. Examples are AT&T Wavelan, Proxim RangeLan, IBM Wireless LAN offerings, etc. From a network layer point of view all WLANs export a connection-less best-effort packet delivery service. However, as will be discussed later, this packet delivery service differs from that of wired networks in many aspects. The environment we 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.

Hosts on the existing wireless LANs (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 more error prone. Within the ISM[1] band, due to multi-path fading, error characteristics of wireless channels are bursty and time varying. In addition, user mobility and frequent handoffs cause burst packet losses. Current generation wireless LANs 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 Transmission Control Protocol (TCP) degrades significantly.

A method for fast recovery from losses during handoff has been proposed in [3]. It exploits TCP's fast retransmit option to reduce TCP's loss detection latency. This method, however, cannot be effectively used to recover from burst errors due to fading. In [2, 16] another 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 [1] achieves the same objective with less book keeping and state maintenance at the base station. Unfortunately, these techniques do not perform well when losses on the wireless channel are bursty.

In this paper we investigate the design space of link layer mechanisms for recovering from burst errors. We begin by carefully studying the effect of burst packet losses on the end-to-end throughput of a TCP connection. We also study the effect of adding link level error recovery on TCP's performance. We have observed that due to the coarse-grained timer (of granularity 500 msec) used in TCP implementations, loss detection latencies are very high. This latency allows sufficient time at the link layer to recover from burst errors. We investigate the performance of multiple TCP connections over a wireless LAN which employs a link level error recovery protocol. This is a likely scenario, in which mobile users gain access to the backbone network via one base station. The channel is subject to burst errors independently for each connection. We observe that under FIFO packet scheduling, the head of the line packet, if encountering burst losses, can block the transmission of other packets. In addition to reducing the overall throughput, this phenomenon causes unfairness, as a particular receiver in a burst error state can reduce the throughput of all other receivers.

We propose link level channel state dependent packet (CSDP) scheduling methods to overcome

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

the problems associated with multiple sessions sharing the wireless link. We show that appropriate packet transmission scheduling can increase overall throughput and reduce the unfairness problem, without interfering with TCP's retransmission schemes. We use TCP as an example protocol to motivate the problem and illustrate the effectiveness of our solution approach. CSDP scheduling methods can be implemented at the base station's wireless device driver. Since this is a link layer solution, not only TCP, but all UDP based applications such as `rpc` and `NFS` also benefit from it. While our experiments were conducted only for a CSMA/CA + priority ACK MAC protocol, we expect that the insights derived from this work will be useful for any link layer protocol design for wireless LANs.

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 head of the line blocking effect that is observed when error-recovery is employed at the wireless link layer. In section 5, we describe the channel state dependent packet scheduling (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 show how these errors degrade throughput and channel utilization.

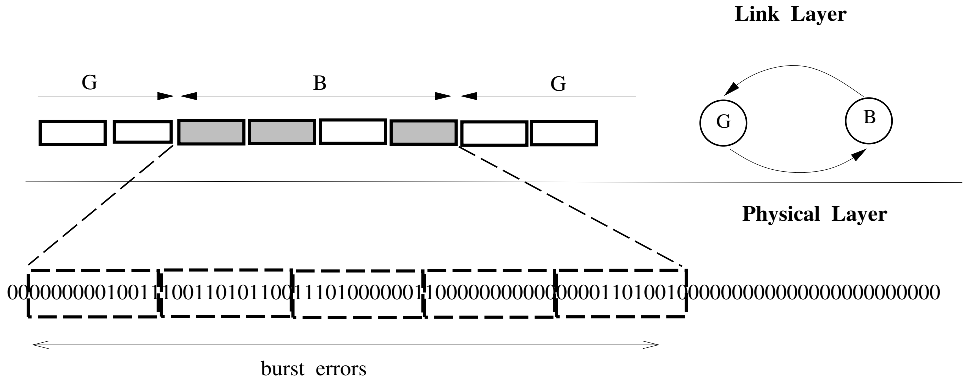

Figure 1: Error Characteristics

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 on to 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, lack of sophisticated signal processing techniques such as adaptive power control, forward error correction, and bit interleaving. Measurements of a particular WLAN system appear in [6]. Their observations 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.

2.1 Channel Loss Model

Many earlier studies in the literature [13, 15] have established that finite state Markovian models can be effectively used to characterize bit error pattern observed on RF channels. Each state of the Markov model is associated with a BER value which indicates the probability of an error in bit reception. The channel probabilistically switches between states with high and low BER values. Note that for most systems, a single bit error on the channel translates into the loss of a complete packet (see figure 1. 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 than 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 [6]. At any point in time, we model the channel as being in one of the two possible states, G(good) or B(bad). We use the term, p^S to denote packet loss probability in state S, where $S \in \{G, B\}$. Of course, 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 Packet Losses

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

Figure 2 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 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 ftp was in progress packets on the wireless segments were dropped using a 2-state loss model described in the

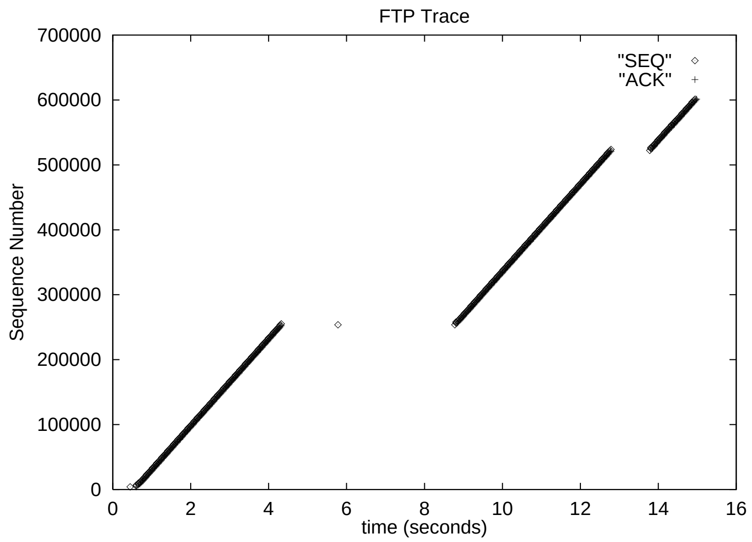

Figure 2: An ftp connection subject to burst losses

previous section. The controlled dropping of packets was accomplished by modifying the wireless device driver at the base station. This trace was collected using tcpdump [11] running on a DEC alpha which is connected to the ethernet.

In the trace collected above, round trip time values are in the 20ms range, the receiver window size is 4K, and the maximum segment size 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%. Below, we enumerate some of the problems that we observed during our experiments.

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 phase. In [3], 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*$O(log_2(W/mss^2))$ time to restore the window size to its original value. 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.

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 occured, and there is no need to wait for a timeout

[2] W is the window size and mss is the segment size.

before retransmitting. 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). Secondly, 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 [9] exponential retransmit back-off algorithm. For example, in Figure 2, 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.

The main reason behind these problems is that TCP was designed around the assumption that links are reliable. It interprets packet losses to be resulting from congestion in the network and consequently reacts by invoking congestion control mechanisms. This justified if underlying links are reliable in which case the true reason behind the packet loss is most likely to be congestion and the only remedy is to reduce input rate. However, triggering a congestion avoidance mechanism in response to each packet loss in wireless medium may lead to unacceptable level of performance. Infact, if wireless link is indeed flaky then TCP's slow start mechanism may repeatedly get triggered, resulting in a a scenario where TCP's window size stays at the minimum possible value [10].

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 capable of recovering from packet losses on the wireless medium, as pointed in the previous section, the latency of timeout based loss detection mechanisms is very high. 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 [5] 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

[3]In [4], 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.

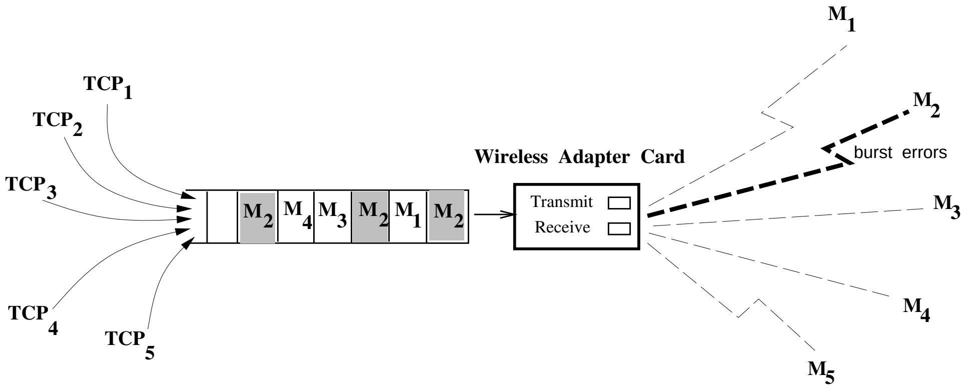

Figure 3: HOL blocking in FIFO scheduler

a medium. To allow detection of a lost frame (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 frame is found correct. If an ACK is not received immediately following a packet transmission, the source quickly times out and retransmits the lost frame 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 frame 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 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 next 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 have observed that in RF wireless LANs typical burst periods are 50 to 100 ms long. During this period, all packet transmission attempts (to a specific destination) fail with a very high probability. The result is that receivers which are located in fade zones claim more wireless resources in comparison to receivers located in good zones. Moreover, packets which are lost during burst periods are interpreted by end-to-end transport layers as signs of congestion [3]. Transport layer mechanisms react to these losses by throttling their transmission rates causing poor throughput. We conducted a simulation study to to perform a careful investigation of this interaction, and to develop solutions to alleviate this problem. We chose TCP as an example end-to-end transport protocol due to its wide spread deployment with the Internet. The observations made below are not specific to TCP, rather they apply to any end-to-end transport protocol, such as those employed in DEC-BIT, RPC etc.

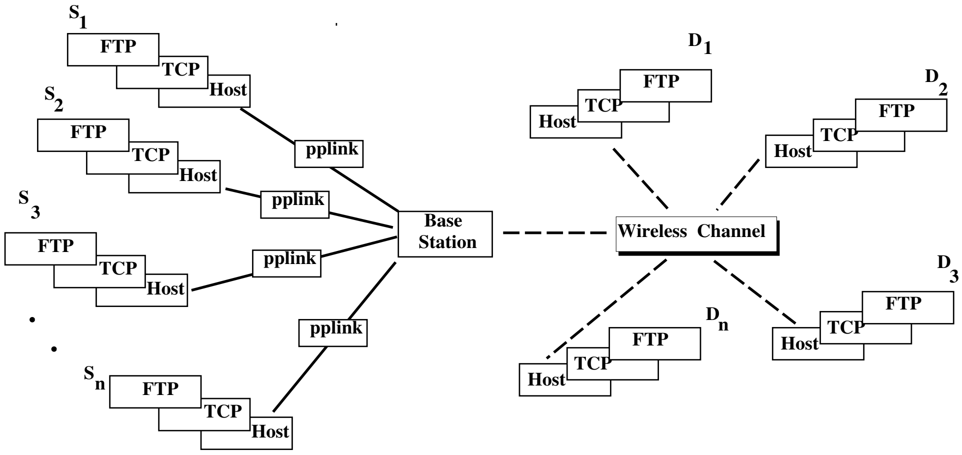

Figure 4: Simulation setup

We study a simple network topology consisting of a single base station providing coverage to N mobile hosts. There are N TCP connections, each starting from a stationary host and terminating at a mobile host. The wireless 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. The transmission line between the source and the base station has a bandwidth of 10Mb/s and a propagation delay of τ.

Each TCP session involves a 0.8Mbyte file transfer. The propagation delay τ is set equal to 16ms 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 wireless link is at least an order of magnitude slower than the speed of the wired segment. We assume sufficient 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, user population within each wireless segment is expected to be small.

Figure 5 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. For these experiments, mean burst length was set to 500ms, mean good period was set to 5 sec and R_{max} was set equal to 8. The following observations became clear from these experiments:

1. **Blocking Effect:** Head-of-the line blocking is observed because the wireless LANs constitute a single wireless channel which is shared on-demand among multiple receivers. When the head of the line packet is destined to a receiver which is located in a fade zone, it prevents other packets in the FIFO queue from being transmitted. 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

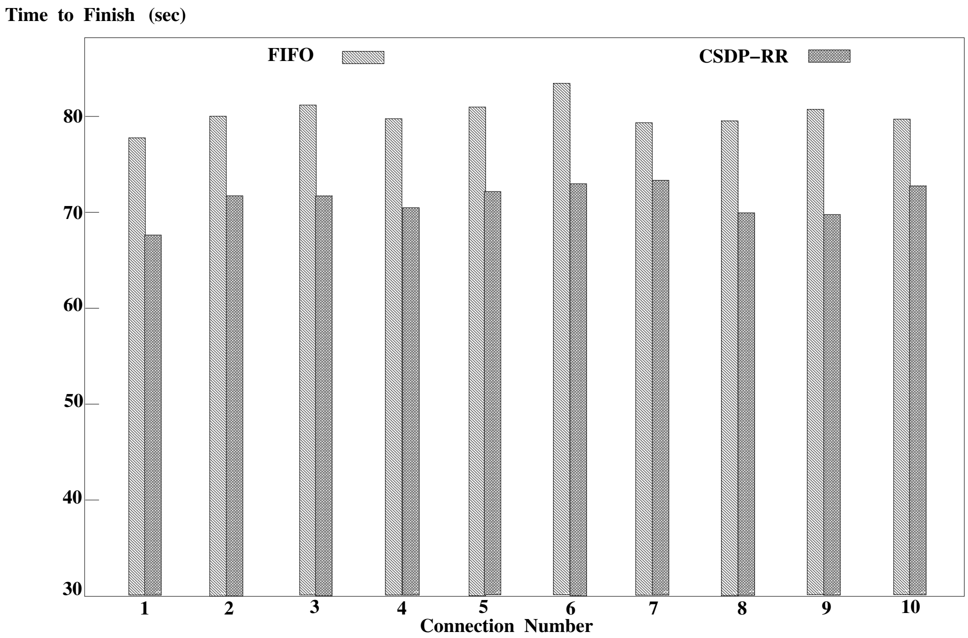

Figure 5: Comparison of FIFO and CSDP-RR, burst=500ms

a packet to its destination. All 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 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 9.

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 earlier been analyzed in the literature [12]. However, these studies do 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

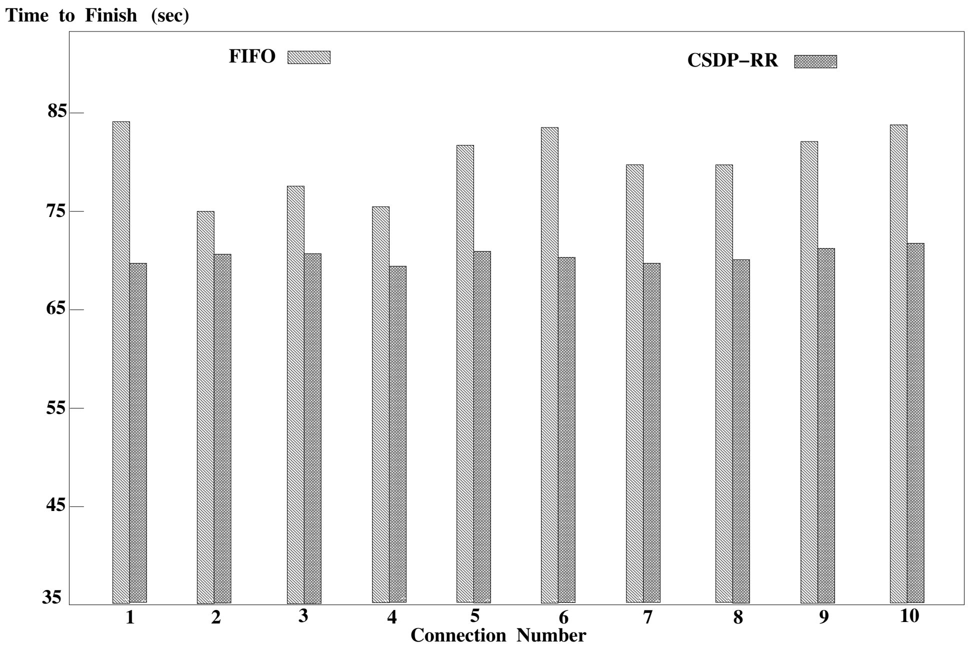

Figure 6: Comparison of FIFO and CSDP-RR, burst=100ms

wireless interface to increase the utilization of the wireless channel while effectively shielding the end-to-end transport sessions from the effects of burst errors on the wireless medium and being fair to all sessions.

5 Channel State Dependent Packet Scheduling

The primary cause of inefficiency is that the CSMA/CA + priority ACK protocol makes repeated attempts to transmit a packet even when the channel is in a burst error state. Moreover, the FIFO packet dispatcher at the WLAN device driver level continues to dispatch packets without taking channel characteristic into consideration. [4] By the time TCP detects packet loss and starts to throttle its transmission, the fade period is usually over. To alleviate this inefficiency, a mechanism is required to defer scheduled transmissions until the start of the next good period. Since the burst periods of different channels are mutually independent, packets for other destinations can be successfully transmitted during this interval.

However, there is a potential risk associated with this approach. If the deferred period length is more than the timeout period of the transport protocol, the source will timeout and retransmit a copy of the delayed packet, thereby unnecessarily increasing the load on the system. In practice, however, we have observed that the timeout period of TCP[5] is on the order of seconds, while the

[4] Prior studies [12] of TCP dynamics have shown that packets from a TCP session have a tendency to arrive in groups. This behavior, in conjunction with FIFO packet dispatcher, makes TCP sessions more susceptible to back-to-back packet losses.

[5] Minimum timeout value is often 1 second, and it grows exponentially for each failed retransmission attempt

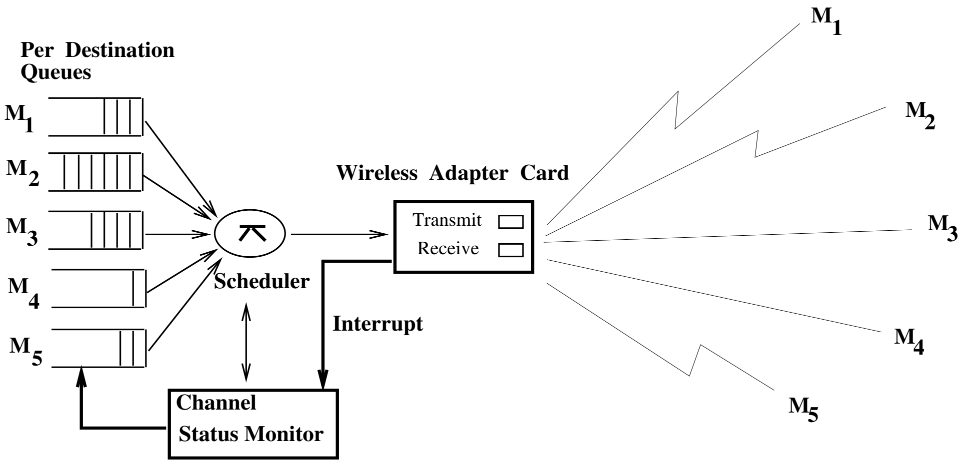

Figure 7: Channel State Dependent Packet Dispatching

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.

$S_i(t)$	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
$\xi_{bi}(t)$	remaining duration in state B, of the channel between host i and the base station, at time t

---

```
Receive(packet𝒫)
        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 T) {
                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
```

5.1 CSDP Scheduling Mechanism

Our solution involves modifying the packet dispatching function at the base station WLAN device driver layer. A scheduler which can take the characteristics of each wireless channel into consideration in making packet dispatching decisions would improve wireless link utilization. We refer to the class of such schedulers as Channel State Dependent Packet (CSDP) schedulers. 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. Within each queue, packets are served in FIFO order. However, service across per 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 station. Whenever the channel between the base b and the mobile station i is in burst error mode, LSM marks the queue (Q_i) for destination i. The channel is assumed to be in burst error state whenever the MAC layer ACK is not received despite multiple transmission attempts on that channel. Q_i is unmarked after an estimated burst period length[6]. It is possible to design more robust mechanisms for detecting burst error periods if the physical layer can inform the MAC layer about packet reception status. For example, since the physical layer manages CRC, it should be able to inform the MAC layer about CRC failures. 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 certain threshold). This notification will help the MAC layer to make a distinction between errors due to collision and errors due to signal attenuation. 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.

5.2 Scheduler Operation

A CSDP scheduler operates by choosing a packet (at the head of the line) from one of the unmarked queues. If all unmarked queues are empty, then it picks up a packet from one of the marked queues. Immediately following the packet transmission, if an ACK is not received within a short timeout period, CSDP returns the packet to the head of the queue Q_i, increments a counter (R_i which records the number of packet transmission attempts), and selects another packet according to scheduling policy. If R_i exceeds R_{max}, the packet is dropped (i.e., it is not put back at the head of Q_i) and the counter R_i is reset to 0. Notice that CSDP mechanism avoids retransmitting the lost packet immediately following an errored transmission. Thus, in 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.

[6] If burst periods are exponentially distributed, one possible estimator could be $mean + 2 * variance$

5.3 CSDP Implementation Complexity

The performance benefit of a CSDP scheduler comes at the marginal cost of 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 8). At any point in time, the number of queues is equal to the number of active [7] 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 1 shows the memory to memory copy latency on a 33Mhz, 486DX processor based machine. 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.

5.4 CSDP Scheduling Policies

CSDP scheduling framework lends an easy implementation of a variety *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 (π). It is important that the scheduling algorithm 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, 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. If packet transmission fails, retry counter is incremented, and the corresponding queues is marked. Retransmission is attempted when this queue is chosen next time by the scheduler. Notice that a Pure-RR scheduler will continue to retransmit HOL packet until it succeeds or the max retry counter expires.

CSDP-ETF Packets arriving from the wired side 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.

[7] By active we mean those stations which have active, on-going data transfer sessions

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

Table 1: Memory to memory copy latency

CSDP-LQF CSDP-LQF scheduler always tries to minimize maximum backlog by always chosing packets from the longest queue. If errors on the wireless channels are statistically identical and packet arrivals follow a poisson distribution then this policy will provide fairness and maximize channel utilization [14].

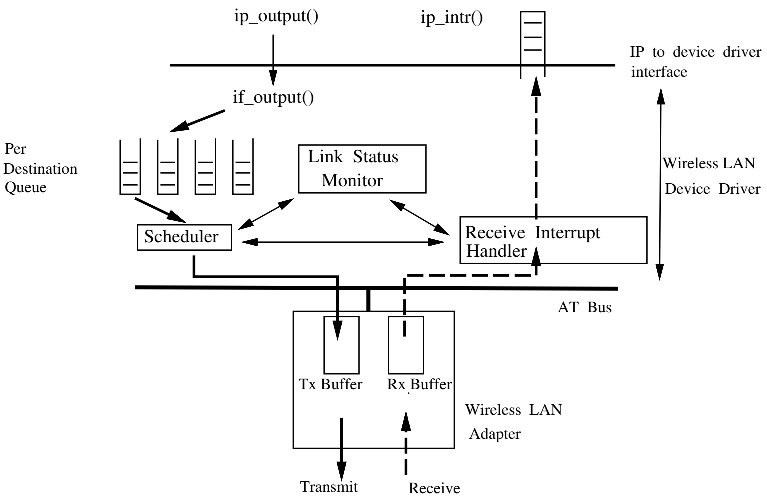

Figure 8: Implementation of per destination Queuing

6 Numerical Results

Mutual interaction of TCP's flow control and the link layer dispatching mechanism in 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 [7] simulator. Each simulation run involves data transfer over N 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 time it takes to complete each TCP session (T_i^π). From the collected data set we compute:

$$T_{min}^\pi = min\{T_1^\pi, \ldots, T_N^\pi\}$$

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

T^{π}_{max} denotes the time when the last TCP session completes. We say that a policy π_1 provides better wireless channel *utilization* compared to policy π_2 if $T^{\pi_1}_{max} < T^{\pi_2}_{max}$. Time difference between the finish time of the first and the last session ($T^{\pi}_{max} - T^{\pi}_{min}$) provides us a measure of *fairness.* In a perfectly fair system, identical sessions should finish at the same time. Below we compare the performance of FIFO, CSDP-RR, CSDP-ETF and CSDP-LQF schedulers using these two evaluation criteria. Results in the next section demonstrate how CSDP schedulers interact with TCP sources. They also provide a basis for making comparison across different CSDP policies.

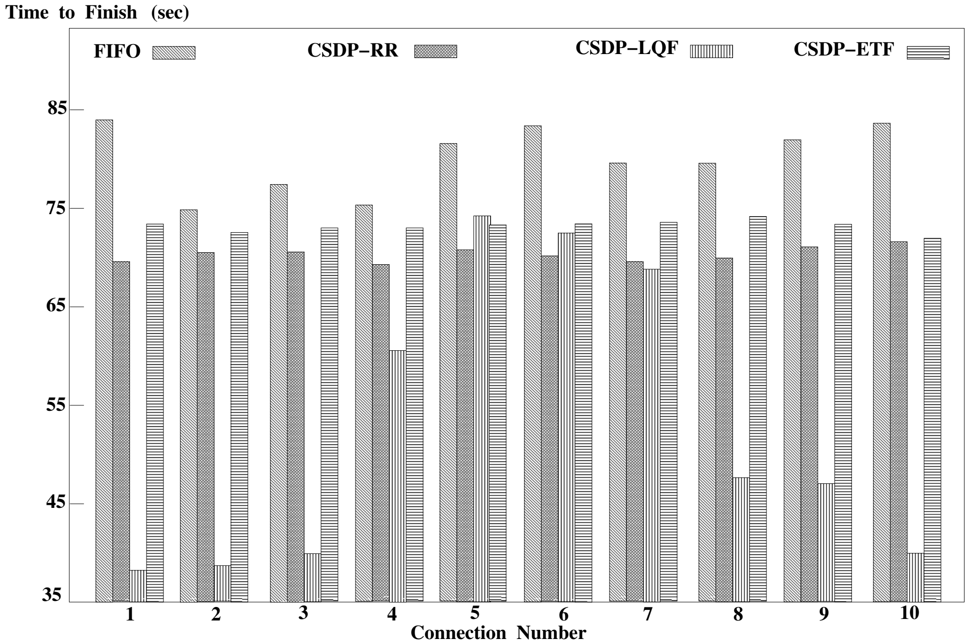

Figure 9: CSDP methods, burst=100ms, R_{max}=8

6.1 CSDP with Perfect Channel Estimation

Figure 9 shows file transfer times for 10 TCP connections. Each session involves a 0.8 Mbyte file transfer. In this experiment, mean burst length was 100 ms, mean good period was 5 sec and max retries limit was set to 8. Packet loss probability in burst period is 0.8. Our results indicate that:

- CSDP schedulers provide better link untilization as $T^{CSDP}_{max} < T^{FIFO}_{max}$.
- Compared to FIFO, sessions under CSDP receive fair service since $|T^{CSDP}_{max} - T^{CSDP}_{min}| < |T^{FIFO}_{max} - T^{FIFO}_{min}|$ (LQF is an exception)

CSDP scheduling policies differ widely in terms of their interaction behavior with TCP sources. Since all TCP sources are identical, relative difference among their transfer times is a good measure of fairness. Under a RR policy all connections finish about the same time. Results for LQF policy

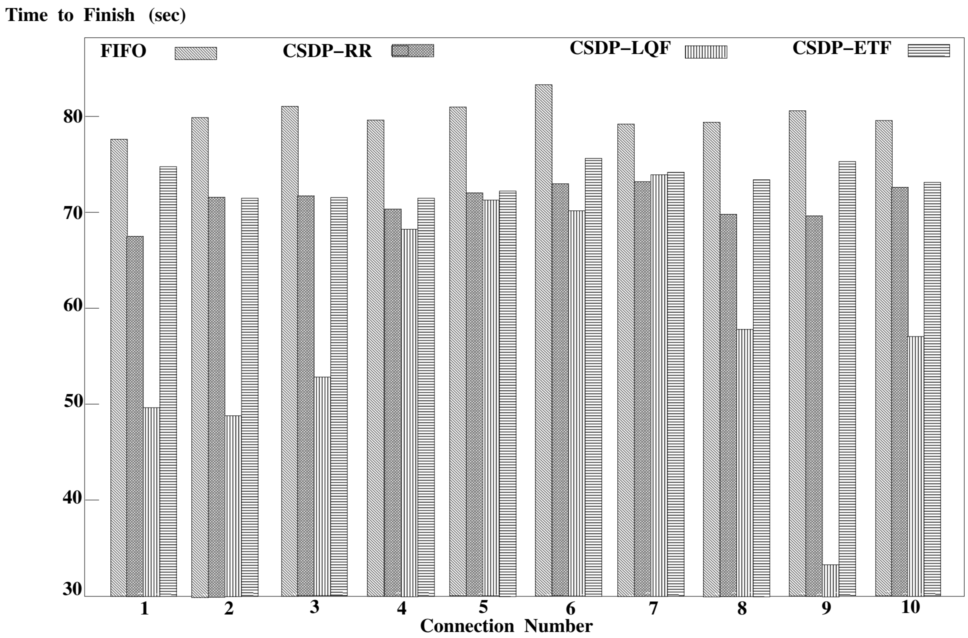

Figure 10: CSDP methods, burst=500ms, R_{max}=8

are rather surprising. Some connections finish very fast while other remain blocked. For Poisson sources and a random error model, LQF is known to be the best policy [14] for this system. This result, however, does not hold when input traffic and loss characteristics on output channels are correlated. As LQF attempts to serve the queue with the maximum number of packets first, the corresponding TCP source receives a bigger share of the bottleneck link. It keeps expanding its window size while packets in other queues wait for their turn for transmission. The result is that some connections 'win' in the beginning while other remain 'blocked' until other connections finish or suffer burst errors. Overall, the round robin CSDP scheduler outperforms all other policies, both in terms of performance and implementation complexity. A summary of our results and a qualitative comparison of various CSDP scheduling methods is shown in the following table.

Policy	FIFO	RR	LQF	ETF
Channel Utilization	worst	best		
Fairness		best	worst	
Throughput	worst	best		
Implementation Complexity	$O(1)$	$O(1)$	$O(n)$	$O(n)$

Data reported in figure 9 is representative of a typical simulation run. We have observed similar performance improvements when the mean length of the burst period was varied from 50ms to 500ms while keeping the ratio between mean good and bad periods equal to 10.

6.2 CSDP with Imperfect Channel Estimation

The level of actual improvement depends on the degree of accuracy in characterizing channel loss behavior. Results reported in the previous section were performed assuming full channel knowledge.

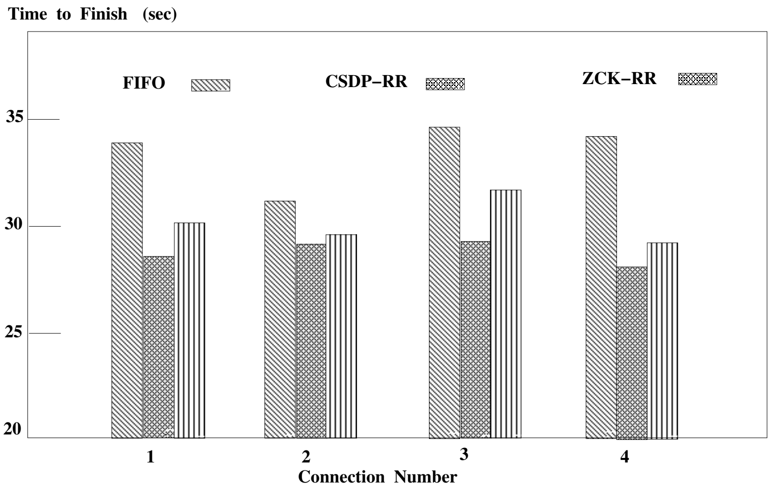

Figure 11: Comparison of FIFO, CSDP-RR and ZCK-RR. Mean burst length = 100ms, $p_B = 0.8$

Under these conditions we have observed up to 10-15 % improvement in channel utilization. In practice, however, this may not always be achievable. To demonstrate the effectiveness of CSDP approach in realistic settings, we operated 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. Deferred period can be thought of as a simple channel estimator, and, as our results show (Figure 11), this simple estimator too works quite well in practice. If errors on the wireless channel are random, the link status monitor will mark all destinations as in "burst" state. In this case, a CSDP-RR scheduler will behave similar to a ZCK-RR scheduler. Thus, CSDP-RR will always outperform FIFO, regardless of what the error characteristics are, and whether a good channel estimator is available or not.

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 [6] and our experience with RF wireless LANs have shown that error behavior on wireless channels is often bursty. We have also observed that using a FIFO packet dispatcher in conjunction with typical wireless link layer protocols results in poor channel utilization, unfairness, and throughput degradation. We have described a simple technique for link layer error recovery that provides an efficient way to recover from the inefficiencies caused by bursty packet errors. Further, the Channel State Dependent Packet (CSDP) scheduling utilizes the distinct states of the channel to each mobile user to increase the efficiency of the link. Since the burst periods are relatively long, there are many performance benefits in monitoring channel state information.

The strength of the CSDP scheduling approach lies in its ability to hide effects of burst errors from transport and application layer protocols. This is achieved by recovering from packet losses at the link layer rather than relying on higher layer error recovery mechanisms. It allows existing protocols and applications to run unmodified over wireless channels and, at the same time, it provides performance comparable to wired networks. Since it is a layer 2 solution, not only TCP, but all UDP based applications such as `rpc` and `NFS` also benefit from it. A layer 2 solution is attractive due to many reasons. It only requires a software modification to the device driver which makes it easier to deploy and test. It can also be used in LAN-to-wireless bridges. Solutions that rely on a transport layer agent [2, 16] or require a soft-state maintenance at the wireless access point cannot be used in wireless bridges.

The level of achievable performance of the CDPS approach depends on the accuracy of the channel state predictor. We have observed up to 15% improvement in channel utilization when complete knowledge of the channel is assumed. Even in absence of channel state information, we have shown that Round Robin version of CSDP provides significant performance improvement over pure FIFO dispatcher. The ability to differentiate between losses due to collision and losses due to channel errors is crucial for the efficient operation of CSDP scheduler. If a loss due to collision is misinterpreted as occurring due to channel errors, then the CSDP scheduler would unnecessarily defer transmissions for that destination. Based on received signal strength. a wireless receiver might easily be able to distinguish between the two cases. For example, a collision would cause the received signal strength to increase by a few db, while typical errors due to signal fade would cause the received signal strength to decrease by a few db. Many current generation cellular products already use adaptive power control mechanisms to mitigate the effects of fading and other channel fluctuations. It might be also possible to differentiate between collisions and other channel errors by utilizing these sophisticated control mechanisms.

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.

References

[1] E. Amir, H. BalaKrishnan, S. Seshan, and R.H. Katz. Efficient tcp over networks with wireless links. April 1995. preprint.

[2] Ajay Bakre and B. R. Badrinath. I-TCP: indirect TCP for mobile hosts. In *Proceedings of the International Conference on Distributed Computing Systems*, Vancouver, Canada, 1995. IEEE. to appear.

[3] R. Caceres and L. Iftode. The effects of mobility on reliable transport protocols. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 12–20, Poznan, Poland, June 1994. IEEE.

[4] Antonio DeSimone, Mooi Choo Chuah, and On-Ching Yue. Throughput performance of transport-layer protocols over wireless lans. In *Proceedings of the Conference on Global Communications (GLOBECOM)*, pages 542–549. IEEE, 1993.

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

[6] D. Duchamp and N.F. Reynolds. Measured performance of a wireless lan. In *17th Conference on Local Computer Networks*, pages 494–499, Minneapolis, MN, September 1992. IEEE.

[7] Andrew Heybey. NETSIM: The Network Simulator, Sept 1990.

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

[9] Phil Karn and Craig Partridge. Improving round trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9:364–373, November 1991.

[10] 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 *High Performance Networking*, 1994.

[11] Steven McCanne and Van Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *1993 WInter Usenix Conference*. Usenix, January 1993.

[12] Scott Shenker, Lixia Zhang, and Dave Clark. Some observations on the dynamics of a congestion control algorithm. *ACM Computer Communication Review*, pages 30–39, October 1990.

[13] F. Swarts and H.C. Ferreira. Markov characterization of digital fading mobile VHF channels. *IEEE Trans. Vehic. Tech.*, pages 977–985, November 1994.

[14] L Tassiulas and A. Epremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, 39(2):466, March 1993.

[15] H.S. Wang and N. Moayeri. Finite state markov channel – a useful model for radio communication channels. *IEEE Trans. Vehic. Tech.*, pages 163–171, February 1995.

[16] Raj Yavatkar and N. Bhagwat. Improving end-to-end performance of tcp over mobile internetworks. In *Mobile 94 workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.