The Impact of Multihop Wireless Channel on TCP Performance

Abstract

This paper studies TCP performance in a stationary multihop wireless network using IEEE 802.11 for channel access control. We first show that given a specific network topology and flow patterns, there exists a window size W^* at which TCP achieves the highest throughput through maximum spatial reuse of the shared wireless channel. However, TCP grows its window size much larger than W^*, leading to throughput decrease. We then explain the TCP throughput decrease using our observations and analysis of the packet loss in an overloaded multihop wireless network. We find out that the network overload is firstly signified by packet drops due to wireless link-layer contention, rather than the buffer overflow as in the wired Internet. As the offered load increases, the probability of packet drops due to link contention also increases, and eventually saturates. Unfortunately, the link-layer drop probability is insufficient to keep the TCP window size around W^*. We model and analyze the link contention, based on which we propose *Link RED* that fine-tunes the link-layer packet dropping probability to stabilize the TCP window size around W^*. We further devise *Adaptive Pacing* to better coordinate channel access along the packet forwarding path. Our simulations demonstrate an improvement of TCP throughput by 5% to 30% using the proposed techniques.

1 Introduction

TCP is the most popular transport protocol that provides reliable end-to-end data delivery. It adjusts its congestion window size in response to detected packet loss, mainly due to buffer overflow at the bottleneck link in the wired Internet. IEEE 802.11 is the dominant technology in building multihop wireless networks. It coordinates the access to the shared wireless channel, and provides the link abstraction to upper layers such as TCP.

Two unique characteristics of IEEE 802.11 multihop wireless networks affect TCP performance. First, contention for the access to the shared wireless channel is *location-dependent*. Packets may be dropped due to consistent link-layer contention, resulted from hidden/exposed terminals [8]. Second, improving channel utilization through *spatial reuse*, i.e., simultaneously scheduling of transmissions that do not interfere with each other, is highly desirable. Both location-dependent contention and the spatial channel reuse are highly dependent on the offered load, managed by TCP congestion control for TCP flows. In this paper, we study the impact of the location-dependent link-layer contention and spatial channel reuse on TCP performance.

We start with several simple network topologies and flow patterns to illustrate the effect of multihop wireless channel on TCP congestion control and throughput. There are several interesting results coming out of our simulations and experiments. First, given a specific network topology and flow patterns, there exists a TCP window size, say W^*, at which its throughput is maximized via maximum spatial channel reuse. W^* is a function of the number of hops the TCP flow traverses, but remains independent from the bandwidth or delay at the "bottleneck" link. Second, current TCP protocol does not operate around W^* but typically grows its average window much larger, resulting in throughput decrease due to degraded spatial reuse and increased packet loss. We observe 4% to 21% throughput decrease from the highest throughput in our simulated scenarios.

Further analysis of the packet loss reveals the reason for the TCP throughput decrease. In a multihop wireless network, link-layer contention happens before buffer overflow. Packet droppings due to link-layer contention offer the first sign of network overload or congestion. The probability of packet dropping due to link contention increases as the offered load (i.e., TCP window size) increases, and finally saturates when every intermediate nodes along the packet forwarding path has a non-empty packet queue. As long as a node has a reasonably large buffer size, e.g., 20 packets, buffer overflow is never observed except for a few pathological cases.

Unfortunately, the gradually increasing packet dropping probability due to link-layer contention is insufficient to stabilize the TCP window size around W^*. Our modeling of the hidden/exposed terminal effects shows that before the offered load reaches optimal operating point, the dropping probability is nearly zero. As the load exceeds such point but before saturation, the packet dropping probability grows accordingly and becomes non-negligible. The probability flattens out if the load further increases.

Our discovery also sheds some light on how to improve TCP performance over multihop wireless networks. In this paper, we propose two link layer techniques to improve TCP throughput: a Link RED algorithm to fine-tune the wireless link's dropping probability to stabilize the TCP window size around W^*, and an adaptive pacing scheme to better coordi-

nate the spatial channel reuse. These simple techniques lead to 5% to 30% throughput increase compared with standard TCP.

The rest of the paper is organized as follows. Section 2 compares with related work. Section 3 reviews the link-layer contention and spatial channel reuse in an IEEE 802.11 multihop wireless network. Section 4 presents a thorough study of the relationship between TCP window size and throughput in several simple topologies and traffic patterns. Section 5 explains the TCP throughput decrease from the packet loss in multihop wireless networks. Section 6 describes and evaluates link RED and adaptive pacing. We discuss a few related issues in Section 7. Finally Section 8 concludes the paper.

2 Related Work

TCP over first/last-hop wireless cellular networks has been an active research field in the literature. Balakrishnan et al [2] present a summary of TCP optimization techniques. The focus of these TCP designs for the single-hop wireless networks is to make random wireless channel errors transparent from upper layer TCP. If IEEE 802.11 protocol is used in such wireless cellular networks, channel-error induced losses would not be a serious issue since seven link-layer retransmissions can hide most of such channel errors. We study TCP performance in a different wireless network model with *multihop wireless channel*.

Holland et al. [3] investigate the effect of mobility-induced link breakage of wireless ad hoc networks over TCP performance. The focus of their study is on the interaction between DSR routing dynamics and TCP window adaptation. Since most packet losses are due to node mobility, congestion control mechanisms of TCP should not be applied to all loss events. Studies in [15][17][18][19] mainly address a congestion detection issue in improving TCP over mobile ad hoc networks. In particular, [15][17][19] use an end-to-end based measurements to detect whether the packet losses are due to congestion or non-congestion conditions. In [18] the network conditions are detected by ICMP (destination unreachable) and ECN messages from the feedback of the intermediate nodes. In [20], Sundaresan et al. also uses the intermediate node's feedback to determine the sending rate and retransmissions. In this paper, we study the interaction of TCP over the MAC layer of a static ad hoc network. We show that even without mobility induced packet losses, TCP performance is still sub-optimal. This is because TCP can not detect the optimal operating point of the underlying ad hoc network by its current congestion control schemes.

Gerla et al [6, 7] study the impact of TCP ACK on TCP performance, as well as the impact of severe unfairness and capture effect caused by the backoff mechanisms in CSMA and FAMA. TCP is observed to have very small throughput when it traverses multiple wireless hops with a window size larger than 1 packet. The authors call for introduction of link-layer ACKs to help reduce packet drops. Our study shows that even though link-layer ACKs are implemented as in IEEE 802.11, TCP performance still suffers from performance degradation due to link-layer contentions. Two recent papers [22][21] study the fairness issue of multiple TCP flows over pure and hybrid ad hoc networks. In particular, Xu et al. [22] enforce a RED dropping among a local domain based on ideal and busy time slots measurements. Our work is different in the following three points. First, instead of fairness, our goal is to improve the bandwidth efficiency of TCP by helping it detect a maximum spatial reuse operating point of the underlying ad hoc network. Second, [22] measures the neighborhood contention level by monitoring the idle and busy time slots. Local observation of idle slots may be inaccurate depending on the underlying traffic statistics. We make use of the contention drops at the link layer to measure the neighborhood congestion level. Due to hidden terminal effects, contention drops at local queue reflect traffic load at the neighborhood areas. From simulation and analysis, we find that such contention drops exhibit RED-like behavior which does not depend on the underlying traffic patterns. Finally, we do not have the message exchange overhead of [22]; due to the contention at local domain as well as hidden terminal effects, the congestion level is propagated among the competing nodes by repeated retransmission failures and eventual packet drops.

3 Link-layer Contention and Spatial Channel Reuse

We consider a *stationary*, *multihop* wireless network using IEEE 802.11 distributed coordination function (DCF) [1]. A single broadcast wireless channel is shared among all nodes in the network. Only receivers within certain transmission range of the sender can receive the packets. As part of IEEE 802.11 DCF, each packet transmission is preceded by a control handshake of RTS/CTS messages. Upon overhearing the handshake, the nodes in the neighborhood of either the sender or the receiver will defer their transmissions, and yield the channel for the subsequent DATA-ACK transmissions. Since we study stationary network, we do not consider packet loss due to routing breakage. We do not consider packet loss due to channel errors as well, since IEEE 802.11 link-layer retransmission mechanism is sufficient to recovery most of such packet losses.

The link-layer contention, specifically the packet drops due to hidden/exposed terminal problem [9] that is unique to multihop wireless networks, turns out to be a major source of packet loss. A hidden terminal is a sender in the neighborhood of the receiver of another on-going transmission, but out of the transmission range of the sender. Because it may not receive the receiver's CTS broadcast due to various rea-

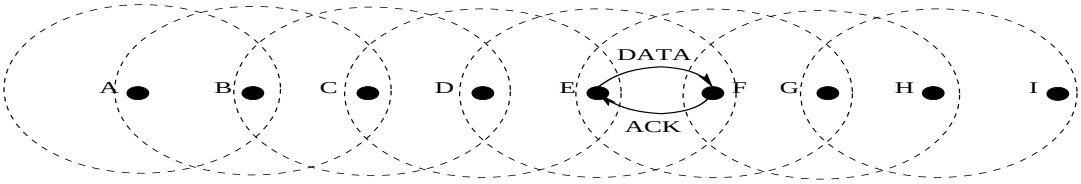

Figure 1: *Location-dependent contention and spatial channel reuse.* 8-hop chain topology. H is a hidden terminal, and C is an exposed terminal for transmission E→F. For optimal spatial channel reuse and maximum end-to-end throughput between A and I, three sets of nodes (i.e. {AE}, {BF}, {CG}, and {DH}) transmit alternatively.

sons such as collisions, a hidden terminal may disrupt the on-going transmission by initiating another transmission. On the other hand, an exposed terminal is a potential receiver in the neighborhood of the sender of another on-going transmission. It cannot receive or respond to another sender's RTS. According to the IEEE 802.11 protocol, a sender drops the a packet after retransmitting DATA four times without receiving an ACK, typically caused by hidden terminals. Besides, a sender drops the packet after sending the RTS message seven times without receiving a CTS, typically caused by exposed terminals.

We illustrate the hidden/exposed terminal problem in Figure 1. In Figure 1, two adjacent nodes are 200m apart. The transmission range of a node is set to 250m, carrier sensing range 550m, and interference range 550m. In this example, node H is a hidden terminal of the on-going transmission E→F. Node H cannot decode F's CTS since it is out of the 250m transmission range. Besides, H cannot sense E's DATA transmission since E is out of H's 550m carrier sensing range. Therefore, node H may transmit to to another node, say node I, at any time, disrupting the on-going transmission E→F. If the DATA transmission between E and F is corrupted four times in a row, node E will drop the packet. On the other hand, node C is an exposed terminal since it is within the 550 carrier sensing range of the transmitting node E. Node C cannot respond to the RTS message from other node, say node B. After seven times of unsuccessful RTS retries node B will drop the packet.

The location-dependence of contention also allows for spatial channel reuse in a multihop wireless network. Specifically, any two transmissions that are not interfering with each other can be scheduled simultaneously. In Figure 1, A→B and E→F can transmit concurrently, reusing the shared wireless channel. Spatial channel reuse can greatly improve the network throughput, especially in a large network that spreads in a wide area.

4 TCP Window Size and Throughput

In this section, we examine the relationship between TCP window size and throughput in multihop wireless networks using various configurations including chain, grid, cross and random network topologies. Our analysis and simulations show that excessive packets in flight (or equivalently, large TCP congestion window size), can only degrade spatial channel reuse and decreased TCP throughput. In fact, the throughput decrease can be as much as 30% in our simulated scenarios. We derive the optimal TCP window sizes at which TCP achieves maximal throughput in simple scenarios. The simulation results for 7-hop chain topology are also verified with experiments.

4.1 Chain topology

We start with the chain topology where packets originate at the first node and are forwarded to the last node. In general, the chain topology represents the packet forwarding path generated by a minimum-hop routing protocol such as DSR [13] and AODV [14].

In a chain topology, the successive transmissions of even a single TCP flow interfere with each other as they move down towards the destination, resulting in link-layer contention and packet drops. Consider the chain in Figure 1 with settings presented in Section 3. It is easy to see that nodes A and E, spaced 4-hop away, can transmit simultaneously. For an h-hop chain, the maximum number of simultaneous transmissions is upper bounded by $\frac{h}{4}$, at which maximum spatial channel reuse is achieved. Because IEEE 802.11 MAC with RTS-CTS-DATA-ACK sequence enforces stop-and-wait for each packet, the pipe size over each hop is 1 packet regardless of the link bandwidth or delay. The total pipe-size over the entire packet forwarding path is therefore $\frac{h}{4}$. Consequently, TCP achieves the highest throughput with its window size being $\frac{h}{4}$ for a h-hop chain, assuming ideal scheduling and identical packet size. If the TCP window size is below this value, it tends to under-utilize the channel; if it is larger, it does not further increase the channel utilization. In fact, as we will show next, it reduces TCP throughput.

The above analysis results match our simulations and experiments, where a perfect scheduler is not available. To obtain the maximum TCP throughput given a chain of specific length, we vary the maximum TCP window size *MaxWin* at

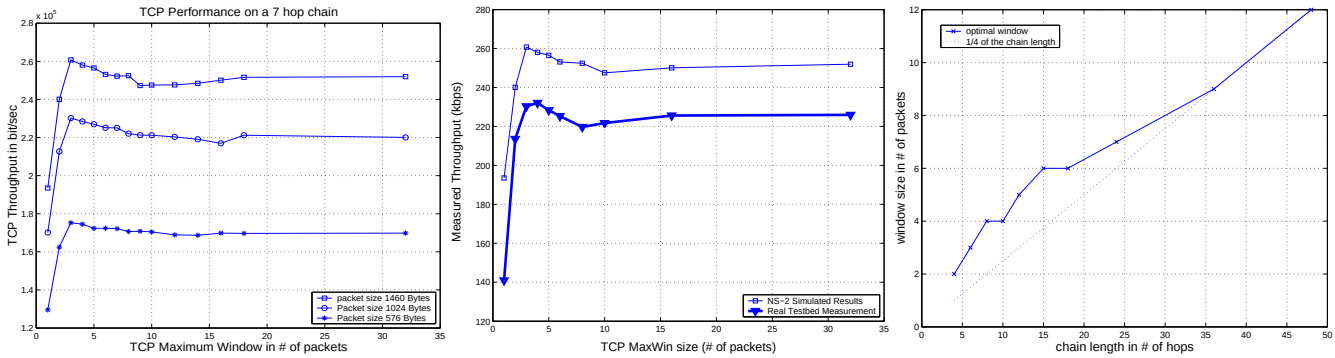

Figure 2: *TCP achieves highest throughput with window size around 3 in a 7-hop chain*. Left: throughput of a single TCP with different packet sizes. Middle: comparisons between ns-2 simulations and testbed experiments with different maximum TCP window sizes. Single TCP, packet size 1460B. Right: TCP optimal window size in chain topologies of different lengths.

Chain length (hops)	4	7	10	16	48
Queue size deviation	1.45	1.31	1.23	1.10	1.05

Table 1: *Deviation of queue lengths*. Chain topologies of different lengths.

MaxWin (packet #)	1	2	4	8	16	32
Avg. TCP wnd size	1	2	3.9	7.1	9.2	9.6

Table 2: *Average TCP window size w.r.t. different* MaxWin*'s in 7-hop chain.*

the sender[1] from 1 to 32 packets. At each *MaxWin* setting, we run a TCP flow for 300 seconds and measure the achieved throughput. The *MaxWin* settings at which TCP achieves the maximum throughputs (i.e., W^*) are plotted in Figure 2 (Right) for chain topologies of different lengths. The figure shows that W^* and $\frac{h}{4}$ match reasonably well, particularly for longer chains ($h > 20$). For short chains, the simulation value is one or two packets larger than $\frac{h}{4}$. The reason is that TCP packets in flight do not distribute evenly among nodes. As the chain becomes longer, the uneven packet distribution (or equivalently the deviation of queue sizes at intermediate nodes) tends to become smaller, as shown in Table 1.

The result of $W^* = \frac{h}{4}$ is independent of packet sizes as long as sizes of successive packets of the TCP flow are identical. Figure 2 (Left) shows that W^* is identical with different packet sizes of 576B, 1KB, and 1460B. However, the throughput achieved by TCP is different due to different IEEE 802.11 MAC overhead.

If we leave the TCP window size unbounded, we observe that the throughput decreases compared with the maximum achievable. Figure 2 (Left, Middle) shows that the throughput decrease is about 4% in a 7-hop chain. As the chain grows longer, the observed throughput decrease can be as high as 10%. Table 2 shows that for *MaxWin* unbounded (bounded to 32), the average TCP sender window size stabilizes at around 9∼10 packets in a 7-hop chain, more than 4 times $W^* \approx 2$. As we will show in Section 4.2, the throughput decrease for TCP flows with excessive window sizes is more significant in complex topologies. In random and grid topologies the throughput decreases as much as 15% to 21%.

As a rough check on the above simulations, Figure 2 (Middle) shows results measured in a testbed. The experiments were configured to mimic the simulation parameters used in Figure 2 (Left). We use Lucent ORiNOCO wireless cards, operating in the ad-hoc mode at 2Mbps. Eight notebooks form a 7-hop chain network and only two neighboring nodes are within the transmission range. Manual routing is used. The average difference between the measured TCP throughput and the simulated results is less than 10%. More importantly, the W^* of the simulations and experiments match perfectly well. It shows that the simulations are accurate enough to model the reality.

4.2 Complex topologies and flow patterns

We extend our study to scenarios of multiple TCP flows and more complex topologies including cross, grid and random topologies. We keep the simulation parameters the same as that in Section 4.1 unless explicitly specified. In all cases, we observe that there exist a window size for TCP to achieve the highest throughput, and TCP in general experience 15% to 21% throughput decrease from the maximum achievable.

Cross topology In the cross network topology shown in Figure 3, we run two TCP flows: one from node 0 to node 6 and the other from node 7 to node 12. Table 3 shows that W^* for each flow is 2, but our measured aggregate TCP window is 12 packets at steady state. 20% throughput decrease

[1]This is equivalent to enforcing flow control in TCP, where *MaxWin* is the advertised receiver window size.

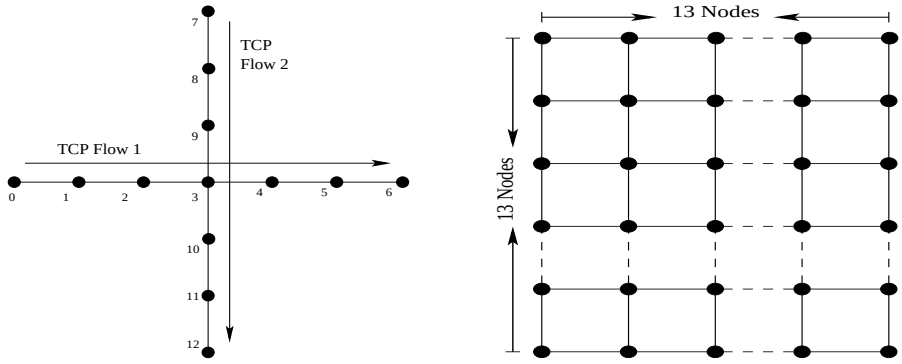

Figure 3: *Complex topologies.* Distance between neighboring nodes is 200 meters. Left: cross topology with 13 nodes and 2 TCP flows. Right: 13x13 grid topology with 4, 8, 12 TCP flows.

Topology	Flow #	Maximum Throughput (Kbps)	Measured Throughput (Kbps)	Optimal Win Size (W^*)	Average Measured Win Size
6-hop Chain	6	298	272	2	22
7-hop Chain	3	255	215	2	16
13-node Cross	2	248	203	4	12
169-node Grid	4	287	241	8	14
169-node Grid	8	957	824	8	19
169-node Grid	12	872	690	8	26
200-node Random	20	1,196	1,015	-	-

Table 3: *TCP throughput and window size.* The data for TCP throughput and window sizes are the aggregation of all flows in topology.

is observed.

Grid topology Figure 3 shows a 13x13 grid topology. We run 4, 8 and 12 TCP flows, respectively. In each case, half the TCP flows go horizontally and the other half go vertically, spaced evenly for either case. The results are summarized in Table 3. In all cases, the measure TCP window sizes are significantly larger than W^*, with throughput decreases up to 21% in 12 flows case.

Random topology We also run extensive simulations with random network topologies generated by the `setdest` tool in *ns-2* distribution. We place 200 nodes are uniform-randomly in a rectangular area of size $1000\text{m} \times 2500\text{m}$. There are 20 TCP flows with their sources and destinations randomly chosen. In our simulations we still observe the existence of a TCP window size at which the maximum aggregate throughput is achieved. TCP throughput suffers up to 15% decrease from the maximum throughput achievable, as shown in Table 3.

4.3 Summary

All our simulations and analysis confirm that for a given topology and traffic pattern, there exists a window size W^* at which TCP achieves the highest throughput through maximum spatial channel reuse. W^* is a function of the number of hops the TCP flow traverses, but remains independent from the bandwidth or delay at any intermediate node. However, if we let TCP *MaxWin* go unbounded as in the normal case, a common observation for all examined topologies and flow patterns is that TCP throughput decreases by 4% to 21%.

5 Packet Loss for TCP Flows in Multihop Wireless Networks

This section studies why TCP throughput decreases at window sizes larger than W^*, and why TCP grows its window size beyond W^*. We start with the examination of the causes to packet loss in multihop wireless networks. Our simulations show that link-layer contention induced packet drop dominates, while buffer overflow is almost never experienced by TCP flows in multihop wireless networks. Since the number of competing nodes for the shared channel increases as the number of in-flight packets increases, a large TCP window size leads to higher degree of the link-layer contention and more packet drops. Therefore, TCP throughput decreases if its window size goes beyond W^*. We finally model the probability of link-layer contention induced packet drop to show that it is insufficient to stabilize the TCP window size at the desired value W^*. The analysis sets the foundation for potential improvements, as we will present in the next section.

Node ID	A	B	C	D	E	F	G	H	I
Max. Q. Size	9	11	13	14	16	15	12	10	6
Avg. Q. Size	0.4	0.8	1	1.9	1.9	1	0.9	0.7	0.3

Table 4: *Queue sizes in packets.* No packet drop due to buffer overflow.

5.1 Packet loss in multihop wireless networks

In the wired Internet, packet losses are mainly due to buffer overflows at the bottleneck router. In a stationary IEEE 802.11 multihop wireless networks, packet loss is mainly caused by either buffer overflows or link-layer contentions due to hidden/exposed terminals (Section 3)[2].

A detailed analysis of our simulations shows that almost all packet loss is due to link-layer contentions. Packet loss due to buffer overflow is rare given a reasonable buffer size at each node, e.g., 20 packets. For example in the chain topology of Figure 1, the maximum queue size is 16 packets at node E, as shown in Table 4. The average queue sizes are all less than 2 packets. Actually in one 300-second simulation run, all 165 TCP packet drops out of the total 12349 transmissions are due to link-layer contention. None is caused by buffer overflows.

We also conduct extensive simulations using different flow layouts and more complex network topologies including grid, cross, and random topologies. The simulation results show that, buffer overflows are rare, and most packet drops experienced by TCP flows are due to link-layer contentions. Packet loss in multihop wireless networks is clearly different from that in the wired Internet. This result implies that TCP congestion control, designed to adapt to the packet loss due to buffer overflow in the wired Internet, may not work well in multihop wireless networks where a different type of packet loss dominates.

5.2 TCP window size and link-layer contention level

The level of link-layer contention increases as the number of nodes that are competing for the shared wireless channel increases, as all possible scenarios of hidden/exposed terminals can happen (see Section 3). Although the queue length for each node is small, the link-layer contention and the consequent packet drop probability will be large as long as a large number of nodes have backlogged queues. On the other hand, the larger the TCP window size, the more packets in

[2]Packet loss due to channel errors will be detected by IEEE 802.11 link-layer acknowledgment.

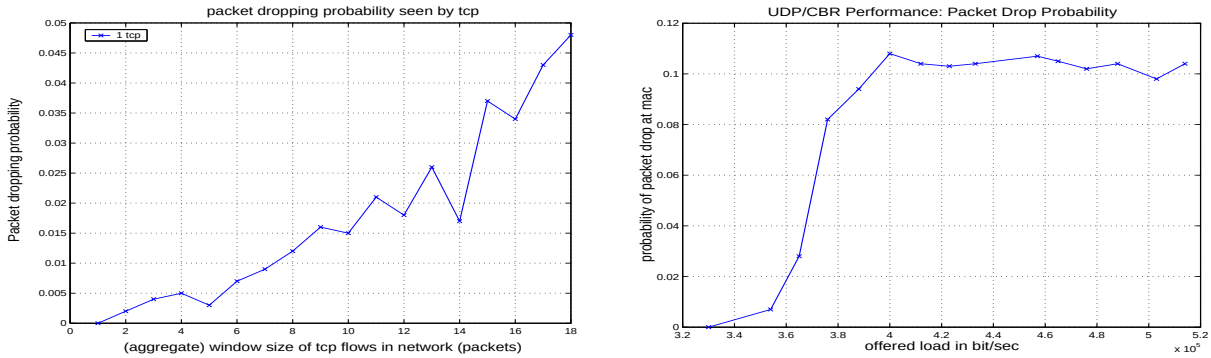

Figure 4: *Probability for link-layer contention packet drops for TCP and UDP flows.* Left: a single TCP flow over a 7-hop chain. Contention drops experienced by TCP flow w.r.t. window sizes. Right: A single UDP flow over a 7-hop chain. Contention drops experienced by UDP/CBR flow w.r.t. offered load.

flight and the more nodes are backlogged, leading to a higher level of link-layer contention and packet loss. Figure 4 shows simulations of single TCP or UDP flow over a 7-hop chain. The left figure plots the link drop probability as a function of the TCP window size. The figure shows that contention drop probability gradually increases from 0% upto 5% as more packets are injected into the chain. To better understand the general case we also simulated CBR/UDP flows with various offered load. As we can see in Figure 4 (Right), there are two knee points on the curve of offered-load-dropping-probability. Before the first knee point, the probability of packet drop due to link-layer contention is nearly zero; after the second knee point, the probability saturates at around 10%. The probability monotonically increases between those two knee points.

Simulation results with more complex topologies, including cross, grid and random topology together also confirm that if we measure the overall contention packet drop probability with respect to the aggregate traffic load level, the curve matches with Figure 4 (Right) with small difference in absolute probability values.

The two knee points in Figure 4 (Right) have clear physical meanings. The first one corresponds to the TCP window size at which the maximum throughput can be achieved through maximum spatial channel reuse, i.e., W^*. The second one corresponds to the maximum contention level when all nodes are backlogged. TCP window size cannot stay around W^*, i.e., the first knee point, since the packet dropping probability is around zero.

5.3 Probability of link-layer contention induced packet drops

5.3.1 A Model for Hidden Terminal Effect

In this section, we consider a general ad hoc network. A node is either in the backoff state, or in the process of RTS/CTS handshake and DATA transmission. Note an initiation of RTS/CTS doesn't guarantee an eventual success DATA transfer. At the steady state, for a given time slot, we define the probability that a node u initiates with RTS for flow f as CS_f, and the probability that a subsequent successful DATA transfer for flow f as B_f. Note they are average value over a long term measurement, and provide a description of average behavior for flow f in steady state.

A successful DATA transfer of flow f requires the sender to initiate the flow first. That is, the sender has to sense idle channel before its RTS initiation. In addition to that, in order for the RTS/CTS handshake to be successful, the receiver must not be hidden by signals from terminals out side of the sender's carrier sensing range. For example, in Figure 1, node D is the hidden terminal of flow A to B. Let H_f be the steady state probability that a flow f is hidden by some terminals, then $B_f = (1 - H_f) \cdot CS_f()$.

Therefore, in steady state, we have

$$H_f = 1 - \frac{B_f}{CS_f} \tag{1}$$

Without loss of generality, we define the term $\frac{B_f}{CS_f}$ as the carrier sensing efficiency of the sender for flow f, which is just the conditional probability of an eventual successful DATA transfer of flow f given f has been initiated. Intuitively, the hidden terminal events are caused by the inefficiency carrier sensing at the sender side.

From the Figure 1, a successful DATA transfer requires idle channel at the areas of both sender's and receiver's neighbor-

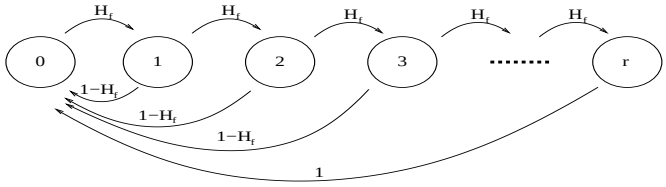

Figure 5: the diagram of the Markov Chain for calculating the packet drop probability from the hidden probability.

hood while the RTS initiation only requires idle channel at the sender side.

The Steady State Contention Drop Probability In order to relate the link layer packet drop with the hidden terminal effect, we note the fact that in 802.11 protocol the packet is dropped after r unsuccessful RTS initiations where r is the $MaximumRetryLimit$ parameter. We consider the discrete Markov model of retrying process in steady state in Figure 5, where H_f is the long term failure (hidden) probability for each RTS initiation. The states represent the number of failed initiations the sender has attempted for a given flow f. Each transition is triggered by RTS initiation. For each initiation, the packet either goes through with probability $1 - H_f$ or fails with H_f. We are interested in the dropping probability p_r, which is just the probability of state r.

By considering state 0, we have $p_0 h = (1 - H_f) \cdot \sum_{i=1}^{r-1} p_i + p_r$, and for each other state, it holds that $p_i = p_{i-1} \cdot H_f = p_0 \cdot h^i \qquad i = 1, 2, \ldots, r$ By the unity condition, we have

$$p_r = \frac{1 - H_f}{1 - H_f^{r+1}} H_f^r$$

Since p_r is probability measure based on number of RTS initiations, to convert it into time slot, we note that the expected time slots for each initiation is $1/CS_f$. And the average packet loss probability L_f for a given time slot in steady state is

$$L_f = p_r \cdot CS_f = \frac{1 - H_f}{1 - H_f^{r+1}} H_f^r \cdot CS_f \qquad (2)$$

5.3.2 Loss/Load Properties in Random Topology

We have discussed the per-flow packet drop probability. Now we relate such drop with the network load in a random topology with multiple flows randomly distributed among the network. Based on the previous observation of the loss/load curves in Figure 4, we define the traffic load as number of competing nodes (or the backlogged nodes) among the network at a given time slot.

In the following, we first represent CS_f and B_f with the terms of network capacity and load from the perspective of the global spatial reuse. Then we apply them to (1) and (2) to derive the steady state contention drop probability with respect to the network capacity and load. We make 2 assumptions here. First, We assume that the traffic are distributed within the network in a purely random fashion. Specifically, for m backlogged senders in the network, each node has an equal backlog probability of $\frac{m}{|V|}$, where $|V|$ is the total number of nodes in the network. Second, we assume that the nodes are also randomly distributed in the network, that is for a network covering an area of S, the expected space each node occupies is $S/|V|$ on average.

At the network level, the carrier sensing capacity,C^*, is defined as maximum number of concurrent RTS initiation in the network without collision; and the data forwarding capacity, B^*, as maximum number of concurrent successful DATA transmissions. It's easy to see that we always have $C^* \geq B^*$.

Again, we consider the system in steady state. Given the global backlog ρ, the average number of backlogged sender is $m = |V|\bar{\rho}$, where $|V|$ is total nodes in network, and $\bar{\rho} = \frac{1}{|V|} \sum_{i=1}^{|V|} \rho_i$. Since we assume these senders distribute in the network evenly, the expected area covered by each node is S/m. In steady state, at a given time slot, in order for all these nodes to initiate with RTS, the minimum spacing required is S/C^*. Therefore, on average, $c(m) = \lfloor m / \lceil \frac{m}{C^*} \rceil \rfloor$ nodes among total m senders can initiate concurrently. Among them, $b(m) = \lfloor c(m) / \lceil \frac{c(m)}{B^*} \rceil \rfloor$ will succeed in concurrent DATA transmission. Therefore for each flow f the carrier sensing probability is readily given $CS_f(m) = \frac{c(m)}{m}$, and successful data forwarding probability $B_f(m) = \frac{b(m)}{m}$. from equation (1), we have $H_f(m) = 1 - \frac{b(m)}{c(m)}$ for each flow. According to (2), per flow loss probability is

$$L_f(m) = \frac{b(m)/m}{1 - (1 - (b(m)/c(m))^{r+1}} \cdot \left(1 - \frac{b(m)}{c(m)}\right)^r \qquad (3)$$

In IEEE 802.11 networks, $r = 7$ for RTS maximum retry count.

For all the backlogged flows, the aggregated loss probability among all $a(m)$ initiated node is given by

$$L(m) = 1 - (1 - L_f(m))^{c(m)} \qquad (4)$$

The following three properties characterize the behavior of contention packet loss with the two threshold values of B^* and C^* defined as follows. In the random topology of N nodes, B^* denotes the maximum number of nodes that can transmit their DATA packets concurrently without collision. At this value, the network achieves highest channel spatial

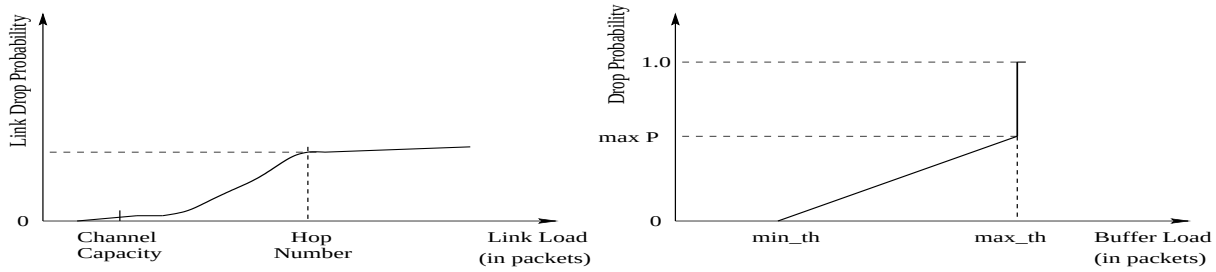

Figure 6: *Comparison between link drop and RED*

reuse. Among N nodes, C^* denotes the maximum number of nodes that can initiate RTS messages, i.e., they perceive clear channel through carrier sensing.

First consider the case when the network is underloaded:

Property 5.1 *Denote the maximum number of nodes (that can concurrently transmit DATA in the given topology) as B^*. When the number of backlogged nodes m is smaller than B^*, i.e., $m < B^*$, then packet drop probability $L_f(m) \approx 0$.*

A detailed proof of this property is pushed to the appendix. Since $m \leq B^*$, on average all m nodes can transmit simultaneously. Therefore, $b(m) \approx c(m) \approx m$ in steady state. According to (3), the drop probability over each link is $L_f(m) \approx 0$. This means that, as long as the network is underloaded, the link drop is negligible.

In the second case when the number of backlogged nodes m is larger than B^*, i.e., the network is overloaded, we have:

Property 5.2 *When the network is overloaded (i.e., the number of backlogged nodes m is greater than B^*), the link drop probability $L_f(m)$ increases as m increases.*

We still use (3) to see why the above is true. In this case, all m nodes can successfully initiate an RTS message but only B^* nodes can transmit their DATA without collisions. That is, $b(m) \approx B^*$ but $c(m) \approx m$. Therefore, $B^* < m < C^*$. It is easy to see that $P_l(m)$ is an increasing function of m since $\frac{dL_f(m)}{dm} > 0$. This shows that link drop probability increases as the network load (as expressed by m) further increases.

Finally, we look at the third case. As the network load further increases, then link drop probability starts to saturate:

Property 5.3 *Once network is heavily loaded in the sense that $m > C^*$, then the link drop probability $L_f(m)$ remains stable in the saturated state.*

In this case, among the m nodes, only C^* out of $c(m)$ nodes can initiate RTS, and only B^* nodes can transmit DATA packets without collision. Therefore, $c(m) \approx C^*$ and $b(m) \approx B^*$. Then long term $L_f(m)$ remains statistically flat according to (3).

5.4 Discussions

Why TCP Suffers from Throughput Decrease Now we use the graceful contention drop behavior to explain why standard TCP suffers from throughput decrease as described in Section 4. TCP achieves highest throughput at the window size W^* that maximizes spatial reuse. The analysis and simulations of Section 5.2 indicate that, the packet drop probability is close to 0 at window size W^*. When the TCP window size grows beyond W^*, the link drop probability starts to increase gradually until it stabilizes around a small value around 5% (Figure 4 Left). Such small drop probability is not sufficient to keep TCP around W^*. Instead, the average window size W_{avg} is much larger than W^*. Thus, TCP flows typically overload the ad hoc network and cause excessive collisions among competing wireless nodes. Having too many packets in flight reduces the network's bandwidth capacity as much as 30% compared with its optimal operating point[12].

Comparison to RED RED is a active queue management protocol to be deployed in the Internet Gateways[10]. It drops packet probabilistically according to the queue length at the local buffer. It is interesting to explicitly compare the contention packet drop with RED drop behavior (Figure 6).

Unlike RED, contention drop is a naturally built-in mechanism and is not specifically tuned for any other protocol other than IEEE 802.11. It is not useful to TCP in the current form unless we the loss/load curve is specifically tuned. In particular, the contention drop may happen before the network capacity is reached, due to the randomness in channel contention; and the maximum drop probability is only 5%, which is too small compared with standard parameters of RED.

However, there exist one important difference that make the contention packet drop a more attractive mechanism in the ad hoc network. As shown by Figure 6, RED drop probability reflects the local queue size, but contention drop probability reflects total number of backlogged nodes within the network, which is a better way to indicate the global load level, a network-wise operating point.

In the next section, we present 2 simple link layer designs to make such contention packet drop more useful to TCP flows.

6 TCP Performance Improvement with LRED and Adaptive Pacing

This section describes two link-layer techniques to improve TCP performance in multihop wireless networks. The Link RED (LRED) technique shapes the curve of packet loss probability v.s. link-layer contention to help TCP stabilizes its window size around W^* for maximum throughput. In addition, Adaptive Pacing aims at improving the spatial channel reuse through better coordination among competition for channel access. The combination of these two techniques is able to improve TCP throughput by as much as 30%.

6.1 Link RED

The analysis in Section 5.2 shows that the IEEE 802.11 inherent link-layer packet dropping probability is too small to stabilize the TCP window size around W^*. The main idea behind our Link-layer Random Early Dropping (LRED) is to control TCP window size through by tuning up the link-layer dropping probability according to perceived channel contentions. Similar to the RED algorithm with a linearly increasing drop curve as the queue size exceeds a minimum value min_th, LRED increases the packet dropping probability when the link-layer contention level, measured as the retransmission counts, exceeds a minimum threshold.

In LRED, the link layer maintains a moving average of the number of packet retransmissions. The head-of-line packet is dropped/marked with a probability based on this average retransmission count. At each node, if the average retransmission count is small, say less than min_th, the head-of-line packets are transmitted as usual. When the average retransmission count becomes larger, the dropping/marking probability is set as the minimum of the computed dropping probability and a upper bound max_P. The LRED pseudo-codes are shown in Algorithm 1.

LRED integrate naturally with ECN-enabled TCP flows. Instead of blindly dropping packets, we can simply mark them at the link layer, and thus allow ECN-enhanced TCP flows to adapt their offered load without losing any packets. TCP performance is therefore improved at the cost of a slightly more complex link-layer design.

To summarize, LRED is a simple mechanism that accomplishes three goals. First, by tuning the loss curve, it serves as congestion signals to elastic flows such as long-term TCP to detect the proper offered load for the underlaying network. Second, by dropping packets more aggressively, it enables TCP to adapt its window size around W^* where maximum spatial channel reuse and minimum channel contention are achieved. Finally, LRED improves the fairness among multiple competing flows, since it reduces the channel capturing effect that is observed in [7].

Algorithm 1 L-RED: LinkLayerSend(Packet p)

Require: avg_retry is the average MAC retries for each packet

1: **if** $avg\_retry < min\_th$ **then**
2: $mark\_prob \leftarrow 0$
3: $pacing \leftarrow OFF$
4: **else**
5: $mark\_prob = min\{\frac{avg\_retry - min\_th}{max\_th - min\_th}, max\_P\}$
6: set $pacing$ ON
7: **end if**
8: mark p with $mark\_prob$
9: MacLayerSend(p, $pacing$)
10: $retry$ = GetMacRetries()
11: $avg\_retry = \frac{7}{8} avg\_retry + \frac{1}{8} retry$

6.2 Adaptive Pacing

In the current IEEE 802.11 protocol, a node is constrained from contending for the channel by a random backoff period, plus a single packet transmission time that is announced by its immediate downstream node. However, the exposed terminal problem (see Section 3) still exists due to lack of coordination between nodes that are *two* hops away from each other. Adaptive pacing solves this problem without incurring nontrivial modifications to the IEEE 802.11 or a second wireless channel [8]. The basic idea is to let a node further back-off an additional packet transmission time when necessary, in addition to its current deferral period (i.e. the random backoff, plus one packet transmission time). This extra backoff interval helps in reducing contention drops caused by exposed terminals, and extends the range of the link-layer coordination from one hop to two hops along the packet forwarding path.

When working together with LRED, adaptive pacing is enabled by LRED only when a node finds the average retransmission count be more than min_th. The pseudo-codes are shown in Algorithm 2.

Algorithm 2 Adaptive Pacing

Require: *extra_Backoff* $= 0$

1: **if** received ACK **then**
2: *random_Backoff* $\leftarrow$ *ran_backoff(cong_win)* {DATA transmission succeeded. Setup the backoff timer}
3: **if** $pacing$ is ON **then**
4: *extra_Backoff = TX_Time(DATA) + overhead*
5: **end if**
6: *backoff* $\leftarrow$ *random_Backoff + extra_Backoff*
7: start *backoff_timer*
8: **end if**

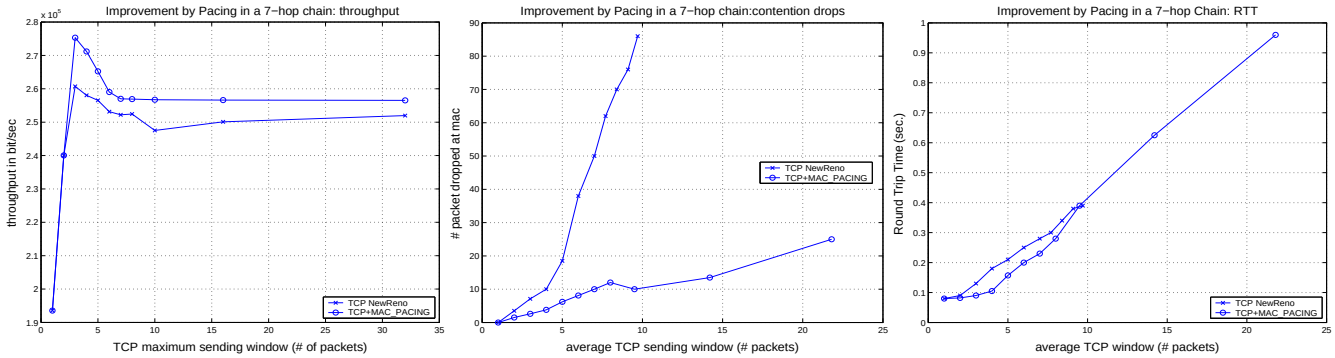

Figure 8: *Adaptive Pacing increases TCP throughput in a 7-hop chain.* Left: Throughput gain with and without adaptive pacing. Middle: Adaptive pacing reduces link-layer contention induced packet drops. Right: Adaptive pacing slightly reduces RTT.

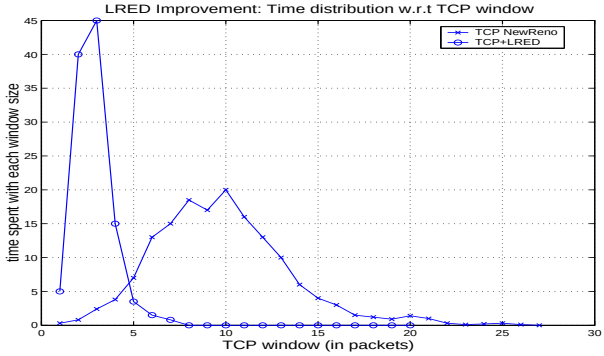

Figure 7: *LRED stabilizes TCP window size around the optimal value.* The Time distribution of instantaneous window size becomes narrower and sharper compared with TCP NewReno.

6.3 Performance Evaluation

In this section, we first evaluate the TCP throughput gain using LRED and Adaptive Pacing individually. We then apply both techniques and show the throughput gain and fairness among multiple TCP NewReno flows in chain, cross, and grid topologies.

6.3.1 LRED

We first use the 7-hop chain to evaluate whether LRED is able to stabilize the TCP window around the optimal point W^*. We run the simulations where the maximum window size set as 32 packets, with and without LRED. The time distribution of different window sizes is shown in Fig 7. As we can see, with LRED the TCP flow spends most of the time with window size $W^* \sim 3$, while the normal TCP grows its window much larger with an average size around 10 packets.

6.3.2 Adaptive Pacing

We use the same 7-hop chain topology to evaluate the effectiveness of adaptive pacing in terms of throughput gain, link-layer contention induced packet drops and TCP round trip time (RTT). Fig 8 shows the simulation results for TCP NewReno flows with and without pacing. With adaptive pacing, TCP is able to achieve up to 10% throughput gain at the window size W^*. The figure also shows packet drop counts and indicates that at a given average window size, pacing has significantly reduced packet drops due to contention and also slightly reduces RTT as shown by the Figure 8 (Right).

Using adaptive pacing alone can not help TCP window size stays around W^*, as shown in the Figure 8 (Left). When the MaxWin is set to 32, the average window achieved is as large as 26, even larger than the case where adaptive pacing is not used (see Table 2 in Section 4). The reason is that adaptive pacing reduces the link-layer contention induced packet drops, further boosting the TCP window size.

6.3.3 LRED+Pacing

Chain topology Figure 9 plots the results for chain topologies of various lengths, with one single TCP flow and six TCP flows. In all cases, we observe that LRED+pacing enhanced link layer is able to increase TCP throughput up to 30%, while LRED stabilizes TCP window size close to the optimal value. For chains longer than 15 hops, our techniques are able to achieve a throughput gain of 10%∼30%. The longer the chain, the better the throughput improvement. This is because longer chain leaves a larger room for the adaptive pacing mechanism to optimize spatial channel reuse.

Cross Topology In the 13-node cross network topology we run two TCP flows as shown in Figure 3 (Left). Table 5 records the throughput and fairness gains for both flows. The fairness results are computed using the fairness index

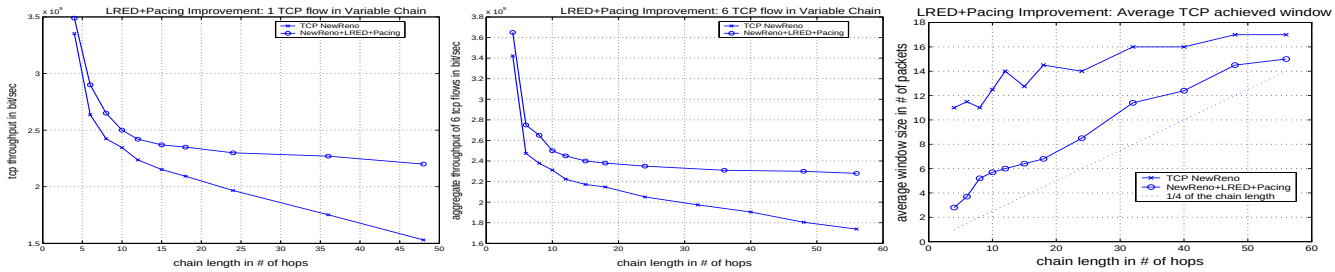

Figure 9: *Performance improvement for TCP NewReno flows in a h-hop chain topology ($h = 3, \ldots, 48$).* Left: single flow; Middle: aggregate throughput of 6 flows; Right: average window size.

	TCP NewReno w/LL	TCP NewReno w/LL+LRED+Pacing
flow 1	244 Kbps	166 Kbps
flow 2	0 Kbps	153 Kbps
Aggregate	244 Kbps	319 Kbps
Fairness	0.5	0.9983

Table 5: *Throughput and fairness comparison between NewReno and NewReno+LRED+Pacing in cross topology*

	TCP NewReno w/LL	TCP NewReno w/LL+LRED+Pacing
flow 1	532 Kbps	85512 Kbps
flow 2	126229 Kbps	90459 Kbps
flow 3	115554 Kbps	70334 Kbps
flow 4	1608 Kbps	47946 Kbps
Aggregate	242923	294251
Fairness	0.51	0.95

Table 7: *Throughput and Fairness Comparisons between NewReno and NewReno+LRED+Pacing.* 4 flows in 13×13 Grid.

$\frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2}$, as defined in [11]. These results show that our design not only increases aggregate throughput, but also improves fairness of both flows. On the other hand, TCP NewReno over the unmodified link layer shows large unfairness between these two flows. The reason is that IEEE 802.11 favors the one flow which captures the wireless channel around the crossing point, as also observed in [6, 7].

Grid Topology Finally, we study the grid network topology with 2, 4, 8 and 12 flows, as shown in Figure 3 (Right). Aggregate throughput and fairness results are recorded in table 6, while more details for the cases of 4 flows are provided in table 7. Again, we are able to achieve about 5%∼10% throughput gain in all cases, while significantly improving the fairness index.

7 Discussions

This section further discusses three important issues of the previous study.

Transition in packet loss during network overload In this paper, we observe that almost all packet losses are due to link layer contention rather than buffer overflow in a *typical* network settings. But will the buffer overflow ever happen for TCP flows? Under what conditions buffer overflow dominates the packet losses? Here we present a simulation study on these two questions.

We start our experiment with a single TCP flow in an 8-hop chain. The traffic source for TCP is a large file; this emulates an FTP connection. We run the 300-second TCP connection multiple times with buffer size of all nodes varing from 2 packets to 19 packets In the presence of packet drop events, we examine the detailed *ns-2* traces to find out the cause for packet loss.

Figure 10 plots the number of each packet loss type as a function of buffer size in each node. It shows a clear transition point (around buffer size of 10 packets) in the dominance of the two loss types.

The figure shows that the loss almost switches from buffer drops to link drops if we change the buffer size from 5 packets to 15 packets. When the buffer size is very small (smaller than 5 packets), buffer overflows dominate and contention loss is almost negligible; when the buffer size grows to 15 packets or more, link-layer contention loss dominates and buffer drops almost vanish. When the buffer size is about 10 packets, both loss types contribute roughly equitable drops. This is not surprising since the larger buffer absorbs more TCP incoming packets and reduces the probability of overflow drops. An interesting observation, shown in Figure 10 also, is that even though the buffer size at each intermediate node has increased to 20 packets, the average number of packets in each node buffer is about 1.2 packets at most. This indicates the average buffer occupancy is quite low, so is its standard deviation (Figure 10 Middle). However, the maximal buffer occupancy of all nodes is very high. This

	NR throughput	NR Fairness	LRED+ Aggregate	LRED+ Fairness
2 flows	203K bps	0.502	252K bps	0.921
4 flows	241K bps	0.508	294K bps	0.952
8 flows	824K bps	0.524	963K bps	0.527
12 flows	690K bps	0.455	880K bps	0.56

Table 6: *Aggregate throughput and fairness comparison between NewReno (NR) and NewReno+LRED+Pacing (LRED+).* 2, 4, 8 and 12 flows in 13×13 grid

clearly shows that highly bursty packet transmissions do happen though infrequently. Two reasons may cause the bursty transmissions: TCP mechanism and 802.11 MAC capture effect. TCP slow-start and the window mechanism may lead to back-to-back transmissions. MAC capture effect [6] due to the binary exponential backoff in contention resolution also incur burst transmissions over a link.

The above simulations are all for single TCP flow. Setting buffer size to 50 packets at each node, we simulated multiple TCP flows over an 8-hop chain and measured the contention/overflow losses (Right of Figure 10). Concurrent flows are introduced with the same starting and ending nodes of the chain topology and are taken as the x-axis. Comparing with the left figure, we observed that the buffer overflow losses increase significantly. However, contention losses still dominate when less than 80 concurrent flows are introduced. This is because that during the period between network gets saturated (each node has a queue size at least 1 packet) and the eventual buffer fill up, contention losses happen with a fixed probability. If this period is sufficiently long, contention packet losses dominate. As more and more concurrent flows are introduced, such buffer fill up period decreases. In addition, the interval between two overflow events also decreases. From the simulation, the overflow losses become majority when more than 100 concurrent flows are introduced.

The above simulations show that contention losses happen before buffer overflow losses. In most cases, they dominate the packet losses even though multiple concurrent TCP flows are introduced.

Other TCP variants Our analysis in Sections 4 and 5 seem to imply that TCP Vegas, which gauges the throughput before increasing its congestion window size, may work better. However, our experiments show that TCP Vegas and TCP NewReno perform comparably in short packet forwarding paths (≤ 6 hops), and TCP Vegas performs 10%$\sim$20% worse than TCP NewReno in longer packet forwarding paths (≥ 9 hops). The main reason is that TCP Vegas keeps its average window size too small (e.g., about 3 packets even in a 16-hop chain). In our simulations we use TCP NewReno, the best existing TCP variants, to compare with.

Variable packet size In most analysis and simulations presented in this paper, we assume identical packet size. If the packet length varies within a flow or among flows, our simulations show that these results still hold: there still exists a TCP window size W^* (in bytes instead of in packets) that achieves maximal throughput, and TCP grows its window size much larger than W^* due to insufficient link-layer packet dropping probabilities.

8 Conclusion

Multihop wireless networks hold great promise in pervasive computing and wireless sensor networks. TCP seems to be the natural choice for reliable data delivery in such networks. This paper systematically studies the impact of the multihop, shared wireless channel on TCP performance. Our results show that only when the buffer is unrealistically small, buffer overflow induced packet drops dominate. In most scenarios packet loss due to link-layer contention dominates. The link-layer contention drop behaves like a RED gateway in terms of graceful drops in the presence of network overload. However, the packet drop provided by link layer is insufficient. This motivates us to design a Link RED which compensates the dropping probability. Through a combination of the LRED and the adaptive pacing at the link layer we can achieve a throughput gain up to 30% for TCP flows, while with the TCP semantics unchanged.

References

[1] IEEE Computer Society. "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," *IEEE standard 802.11*, 1997

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Transactions on Networking*, Dec. 1997

[3] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *ACM MOBICOM* 1999

[4] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing," *IEEE INFO-*

COM 2000

[5] L. S. Brakmo, S. W. O'Malley and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *ACM SIGCOMM* 1994

[6] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, "TCP over Wireless Multihop Protocols: Simulation and Experiments," *IEEE ICC* 1999

[7] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multihop Networks," *IEEE WMCSA* 1999

[8] V. Bharghavan, "Performance Analysis of a Medium Access Protocol for Wireless Packet Networks," *IEEE Performance and Dependability Symposium* 1998.

[9] D. Allen, "Hidden Terminal Problems in Wireless LAN," *IEEE 802.11 Working Group paper 802.11/93-xx.*

[10] S. Floyd and V. Jacobson. "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking* 1(4):397-413, Aug. 1993

[11] R.Jain, D-M. Chiu and W. Hawe. "A Quantitative Measure of Fairness and Discrimination For Resource Allocation in Shared Conputer Systems," *Technical Report TR-301*, DEC Research Report, September, 1984

[12] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris, "Capacity of Ad Hoc Wireless Networks," *ACM MOBICOM*, 2001.

[13] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[14] C. E. Perkins and E. M. Royer. Ad-hoc On Demand Distance Vector Routing. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999.

[15] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," *In Proceedings of ACM MOBICOM'99*.

[16] Zhenghua Fu, et al. "TCP over Multihop Wireless Networks," *UCLA Computer Science Tech. Rep., www.cs.ucla.edu/wing/publication/tech010702.ps*

[17] Zhenghua Fu, Ben Greenstein, Xiaoqiao Meng and Songwu Lu,"Design and Implementation of a TCP-Friendly Transport Protocol for Ad Hoc Wireless Networks", In Proceedings of IEEE ICNP 2002.

[18] J.Liu and S.Singh. "ATCP:TCP for mobile ad hoc networks", IEEE Journal on Selected Areas in Communiations, 19(7):1300–1315 *July 2001*

[19] F.Wang and Y.Zhang, "Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response", In Proceedings of the third ACM international symposium on Mobile ad hoc networking & computing, *2002, Lausanne, Switzerland.*

[20] K.Sundaresan, V.Anantharaman, H.Hsieh and R.Sivakumar, "ATP: A Reliable Transport Protocol for Ad-hoc Networks", In Proceedings of MOBIHOC 2003

[21] L.Yang, W.Seah, Q.Yin, " Improving Fairness Among TCP Flows Crossing Wireless Ad Hoc and Wired Networks", In Proceedings of MOBIHOC 2003

[22] K.Xu, M.Gerla, L.Qi and Y.Shu, "Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED", In Proceedings of MOBICOM 2003

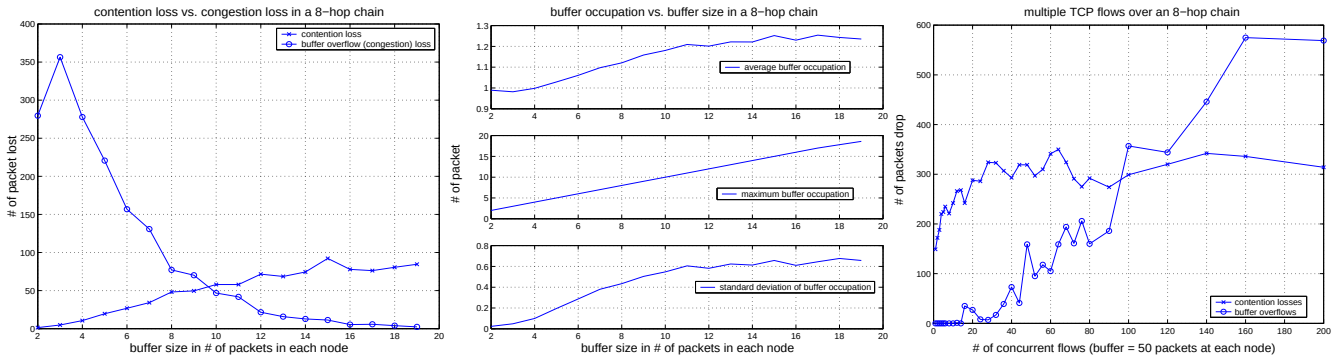

Figure 10: *TCP loss transition* Left: Buffer loss dominates when buffer is small; Contention loss dominates when buffer is large. Middle: Average and Maximum queue size of all nodes in the network. It shows that the average queue size occupancy at each node is very low. Right: Multiple concurrent TCP flows over an 8-hop chain. Buffer size at each node is set to 50 packets. The contention loss still dominates the packet losses.