The Impact of Multihop Wireless Channel on TCP Performance

Zhenghua Fu, *Student Member*, *IEEE Computer Society*,
Haiyun Luo, *Student Member*, *IEEE Computer Society*,
Petros Zerfos, *Student Member*, *IEEE Computer Society*, Songwu Lu, *Member*, *IEEE Computer Society*,
Lixia Zhang, *Senior Member*, *IEEE Computer Society*, and
Mario Gerla, *Fellow*, *IEEE Computer Society*

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 an *optimal* window size W^* at which TCP achieves the highest throughput via maximum spatial reuse of the shared wireless channel. However, TCP grows its window size much larger than W^*, leading to throughput reduction. 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 typically first signified by packet drops due to wireless link-layer contention, rather than buffer overflow-induced losses observed 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 behavior, 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 5 to 30 percent improvement of TCP throughput using the proposed two techniques.

Index Terms—TCP performance, multihop networks, congestion and contention control.

1 Introduction

TCP is the most popular Internet 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. As IEEE 802.11-based wireless networking technology gains popularity, TCP is very likely to continue to be the dominant transport protocol in order to reuse the numerous network applications developed so far. In an 802.11-based multihop wireless network, the underlying MAC coordinates the access to the shared wireless channel and provides the link abstraction to upper layers such as TCP. In this paper, we seek to gain understanding on how TCP operates in such a multihop wireless network.

Two unique characteristics of IEEE 802.11 multihop wireless networks may greatly 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., simultaneous scheduling of transmissions that do not interfere with each other, is highly desirable. The window adaptation mechanism of TCP impacts the degree of spatial reuse. In summary, both location-dependent contention and spatial channel reuse are highly dependent on the offered load, managed by the TCP protocol. 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, and have obtained interesting results from 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 of 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 percent throughput reduction 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 typically 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 node along the forwarding path has a nonempty packet queue. As long as each node in the multihop wireless network allocates a reasonably large

• *The authors are with the Computer Science Department, University of California at Los Angeles, Los Angeles, CA 90095-1596.*
E-mail: {zfu, hluo, pzerfos, slu, lixia, gerla}@cs.ucla.edu.

Manuscript received 12 Oct. 2003; accepted 28 Jan. 2004; published online 27 Jan. 2005.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0169-1003.

buffer, 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 the optimal operating point, the dropping probability is nearly zero. As the load exceeds such a point, the packet dropping probability grows accordingly and becomes nonnegligible. The probability flattens out and saturates 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 finetune the wireless link's dropping probability to stabilize the TCP window size around W^*, and an adaptive pacing scheme to better coordinate the spatial channel reuse. These simple techniques lead to 5 to 30 percent throughput increase compared with the standard TCP.

The rest of the paper is organized as follows: Section 2 compares with related work. Section 3 reviews link-layer contention and spatial channel reuse in an IEEE 802.11-based 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 perspective of 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 wireless cellular networks has been an active research topic. Balakrishnan et al. [2] summarized such TCP optimization techniques. The focus of these TCP designs over the single-hop wireless link is to make random wireless channel errors transparent from TCP. If IEEE 802.11 protocol is used in such wireless cellular networks, channel error induced losses would not be a severe issue since seven link-layer retransmissions can hide most of such channel errors. We study TCP performance in a different wireless network setting with *multihop wireless channel*.

Holland and Vaidya [3] investigate the effect of mobility-induced link breakage of wireless ad hoc networks upon 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 such loss events. Studies in [15], [17], [18], [19] mainly address the issue of congestion detection in improving TCP over mobile ad hoc networks. In particular, [15], [17], [19] use end-to-end measurements to detect whether the packet losses are due to congestion or noncongestion conditions. In [18], the network conditions are detected by ICMP (destination unreachable) and ECN messages based on the feedback of the intermediate nodes. In [20], Sundaresan et al. also uses the intermediate node's feedback to decide the sending rate and retransmissions. In this paper, we study the interaction of TCP and the link layer of a static ad hoc network. We show that, even without mobility induced packet losses, TCP performance is still suboptimal. This is because TCP cannot 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 and the effect of unfairness and capture effect 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 one packet. The authors call for the introduction of link-layer ACKs to help reduce packet drops. Our study shows that, even though link-layer ACKs are implemented 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. Xu et al. [22] propose RED dropping in a local network neighborhood based on ideal and busy time slot measurements. Our work is different in three aspects. First, our focus is not fairness, but to improve the bandwidth efficiency of TCP by letting TCP detect an operating point that enables maximum spatial reuse 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 not be accurate enough since it depends on the ongoing traffic statistics. We make use of the contention drops at the link layer to gauge the neighborhood congestion level. Due to hidden terminal effects, contention drops at a local node also reflect traffic load at the neighborhood areas. Finally, our design does not incur the message exchange overhead of [22].

Using pacing as a method of congestion control in packet radio networks has been proposed in [23]. However, pacing in [23] resembles the backoff mechanism which has already been adopted in IEEE 802.11 protocol. In our paper, pacing is different with the standard backoff mechanism and is proposed specifically to reduce the hidden terminal problem.

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 wireless channel is shared among all nodes in the network. Only receivers within the transmission range of the sender can receive the packets. In 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 subsequent DATA-ACK transmissions. Since we study a stationary network, we do not consider packet loss due to routing breakage.

In this paper, we assume that multihop contention, i.e., due to hidden/exposed terminal problem [9], is the main source for packet losses. Note that packets can also get dropped due to out-of-band channel errors. In IEEE 802.11 networks, however, the retransmission mechanism hides most uncorrelated channel noises for nonbroadcast traffic. De Couto et al. [24] reported that 1-hop unicast packets losses seen by the higher layers were indeed very low with the link-level retransmission mechanism.

A hidden terminal is a sender in the neighborhood of the receiver of another ongoing transmission, but out of the

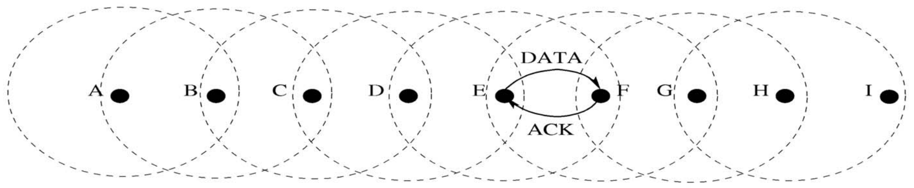


Fig. 1. *Location-dependent contention and spatial channel reuse.* Eight-hop chain topology. H is a hidden terminal and C is an exposed terminal for transmission $E \rightarrow F$. For optimal spatial channel reuse and maximum end-to-end throughput between A and I, four sets of nodes (i.e., {AE}, {BF},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

transmission range of the sender. Because it may not receive the receiver's CTS due to various reasons such as collisions, a hidden terminal may disrupt the ongoing transmission by initiating another transmission. On the other hand, an exposed terminal is a potential receiver in the neighborhood of the sender of another ongoing transmission. It cannot receive or respond to another sender's RTS. According to the IEEE 802.11 protocol, a sender drops the 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 CTS, typically caused by exposed terminals. We illustrate the hidden/exposed terminal problem in Fig. 1. In Fig. 1, two adjacent nodes are 200m apart. The transmission range of a node is set to 250m, the carrier sensing range is 550m, and the interference range is 550m. In this example, node H is a hidden terminal of the ongoing transmission $E \rightarrow F$. 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 another node, say node I, at any time, disrupting the ongoing transmission $E \rightarrow F$. 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 another node, say B. After seven 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 Fig. 1, $A \rightarrow B$ and $E \rightarrow F$ can transmit concurrently, reusing the shared wireless channel. Spatial channel reuse can greatly improve the network throughput, especially in a large network that spreads 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 decrease TCP throughput. In fact, the throughput decrease can be as much as 30 percent 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 real 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 downstream toward the destination, resulting in link-layer contention and packet drops. Consider the chain in Fig. 1 with settings described in Section 3. It is easy to see that nodes A and E, spaced 4-hops 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 one 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 an 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 matches 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 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 Fig. 2c 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. Fig. 2a shows that W^* is identical with different packet sizes of 576B, 1KB, and 1,460B. However, the throughput achieved by TCP is different due to different IEEE 802.11 MAC overhead.

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

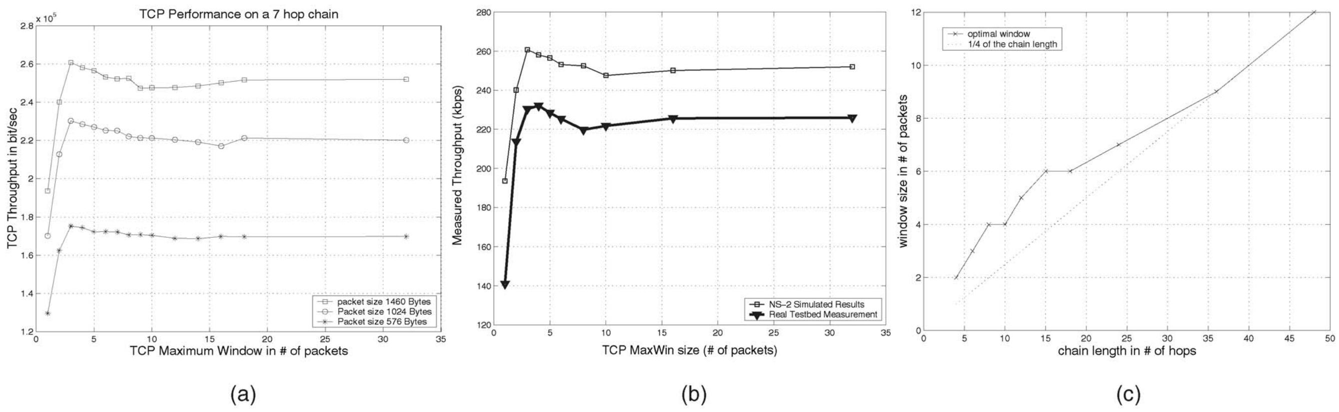


Fig. 2. *TCP achieves highest throughput with window size around 3 in a 7-hop chain.* (a) Throughput of a single TCP with different packet sizes. (b) Comparisons between ns-2 simulations and testbed experiments with different maximum TCP window sizes. Single TCP, packet size 1,460B. (c) TCP optimal window size in chain topologies of different lengths.

If we leave the TCP window size unbounded, we observe that the throughput decreases compared with the maximum achievable value. Figs. 2a and 2b show that the throughput decrease is about 4 percent in a 7-hop chain. As the chain grows longer, the observed throughput decrease can be as high as 10 percent. Table 2 shows that for *MaxWin* unbounded (bounded to 32), the average TCP sender window size stabilizes at around $9 \sim 10$ packets in a 7-hop chain, more than four 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 percent.

As a rough check on the above simulations, Fig. 2b shows results measured in a testbed. The experiments were configured to mimic the simulation parameters used in Fig. 2a. 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 percent. 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, our observation shows that there exists a window size for TCP to achieve its highest throughput, and TCP in general experiences 15 to 21 percent decrease from the maximum achievable throughput.

Cross topology. In the cross network topology shown in Fig. 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. Twenty percent throughput decrease is observed.

Grid topology. Fig. 3 shows a 13×13 grid topology. We run 4, 8, and 12 TCP flows, respectively. In each case, half of the TCP flows go horizontally and the other half go vertically, spaced evenly. 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 percent in the 12-flow 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 uniform-randomly in a rectangular area of $1,000\mathrm{m} \times 2,500\mathrm{m}$. 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 a 15 percent reduction from the maximum achievable throughput, 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 of the bandwidth or delay at any intermediate node. However, if we let TCP window *MaxWin* grow 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 percent.

TABLE 1
Deviation of Queue Lengths

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

Chain topologies of different lengths.

TABLE 2
Average TCP Window Size with Regard to Different *MaxWin*s in 7-Hop Chain

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

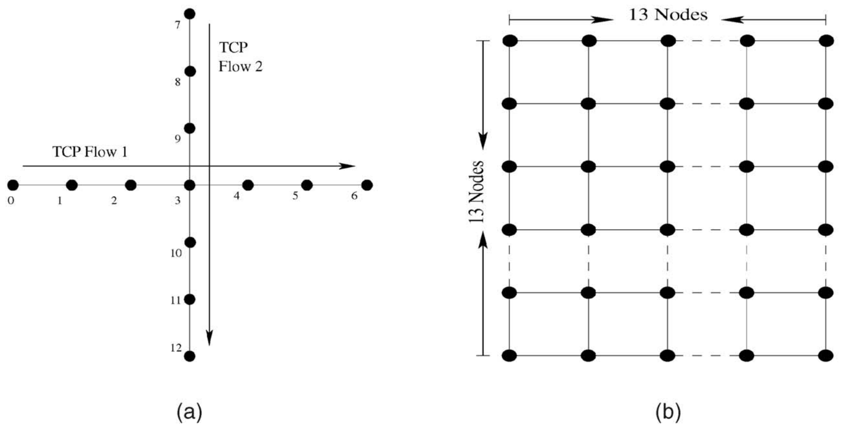


Fig. 3. *Complex topologies*. Distance between neighboring nodes is 200 meters. (a) Cross topology with 13 nodes and two TCP flows. (b) 13×13 grid topology with 4, 8, and 12 TCP flows.

TABLE 3
TCP Throughput and Window Size

Topology	Flow #	Maximum Throughput (Kbps)	Measured Throughput (Kbps)	Optimal Win Size (W^*)	Avg. 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	-	-

The data for TCP throughput and window sizes are the aggregation of all flows in topology.

5 Packet Loss for TCP Flows in Multihop Wireless Networks

This section studies why TCP throughput decreases at window sizes larger than W^* and TCP grows its window size beyond W^*. We start with the examination of the causes of 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 typical multihop wireless networks. Since the number of contending nodes for the shared channel increases as the number of in-flight packets increases, a large TCP window size leads to a higher degree of link-layer contention and more packet drops. Therefore, TCP throughput decreases once 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 forms the foundation for potential improvement, to be presented in the next section.

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 network, 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 Fig. 1, the maximum queue size is 16 packets at node E, as shown in Table 4. The average queue sizes are all less than two packets. In fact, in a 300 second simulation run, all 165 TCP packet drops out of the total 12,349 transmissions are due to link-layer contention. None is caused by buffer overflows.

We also conducted 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

2. Packet loss due to channel errors will be detected by IEEE 802.11 link-layer acknowledgment and recovered by retransmissions.

TABLE 4
Queue Sizes in Packets

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

No packet drop due to buffer overflow.

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 contend 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 flight and the more nodes are backlogged, leading to a higher level of link-layer contention and packet loss. Fig. 4 shows simulations of single TCP or UDP flow over a 7-hop chain. Fig. 4a plots the link drop probability as a function of the TCP window size. The figure shows that contention drop probability gradually increases from 0 up to 5 percent 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 Fig. 4b, there are two knee points on the curve of offered-load versus 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 percent. The probability monotonically increases between those two knee points.

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

The two knee points in Fig. 4b 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

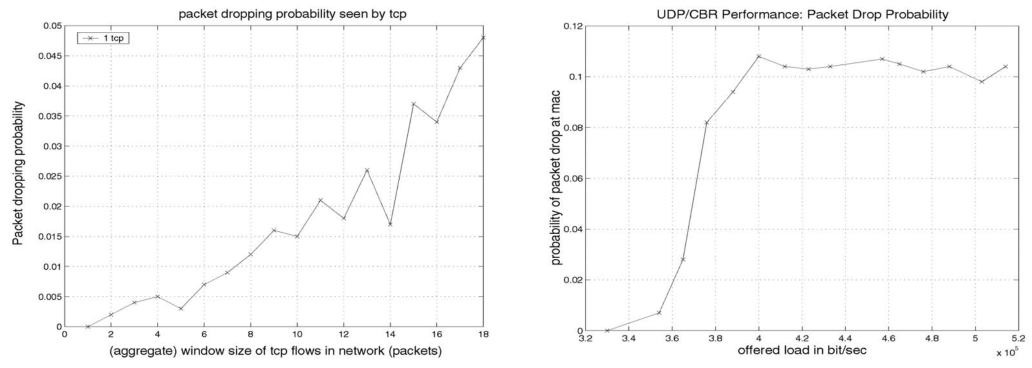
We now consider a generic ad hoc network setting. A node is either in the backoff state or in the process of RTS/CTS handshake and DATA transmission. Note that the RTS/CTS handshake does not guarantee an eventual successful DATA transfer. At the steady state, for a given time slot, we define the probability that a node u initiates RTS for flow f as CS_f, and the probability that a subsequent successful DATA transfer for flow f as B_f. Note that they are the expected value and provide a description of the average behavior of flow f in the steady state.

A successful DATA transfer of flow f requires the sender to initiate the flow first. That is, the sender has to sense an idle channel before its RTS initiation. In addition, in order for the RTS/CTS handshake to be successful, the receiver must not be hidden by signals from terminals outside of the sender's carrier-sensing range. For example, in Fig. 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

$$H_f = 1 - \frac{B_f}{CS_f}. \qquad (1)$$

Without any 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 that f has been initiated. Intuitively, the hidden terminal events are caused by the inefficiency of carrier-sensing at the sender side.



(a) (b)

Fig. 4. *Probability for link-layer contention packet drops for TCP and UDP flows*. (a) A single TCP flow over a 7-hop chain. Contention drops experienced by TCP flow with regard to window sizes. (b) A single UDP flow over a 7-hop chain. Contention drops experienced by UDP/CBR flow with regard to offered load.

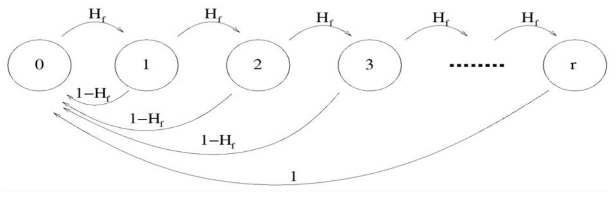


Fig. 5. The diagram of the Markov Chain for calculating the packet drop probability from the hidden probability.

From Fig. 1, a successful DATA transfer requires idle channel at the areas of both sender's and receiver's neighborhood, while the RTS initiation only requires idle channel at the sender side.

The Steady State Contention Drop Probability. In order to relate 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 Fig. 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 an 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, the probability at 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 every other state, it holds that $p_i = p_{i-1} \cdot H_f = p_0 \cdot h^i$ $i = 1, 2, \ldots, r$. Based on the unity condition, we have

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

Since p_r is the probability measure based on the number of RTS initiations, to convert it into time slots, 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

$$L_f = p_r \cdot CS_f = \frac{1 - H_f}{1 - H_f^{r+1}} H_f^r \cdot CS_f. \tag{2}$$

5.3.2 Loss/Load Properties in Random Topology

We have derived the per-flow packet drop probability. We now relate it with the network load in a random topology, in which multiple flows are randomly distributed. Based on the previous observation of the loss/load curves in Fig. 4, we define the traffic load as number of competing nodes (or the backlogged nodes) among the network at a given time instant.

In the following, we first represent CS_f and B_f in 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 two assumptions here. First, we assume that the traffic is 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 $|V|$ denotes the total number of nodes in the network. Second, we assume that the nodes are also randomly distributed in the network, i.e., for a network covering an area of S, the expected space each node occupies is $S/|V|$.

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

We consider the system in steady state. Given the global backlog ρ, the average number of backlogged senders 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 are evenly distributed in the network, 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 transmissions. 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 (1), we have $H_f(m) = 1 - \frac{b(m)}{c(m)}$ for each flow. According to (2), per flow loss probability is

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

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

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

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

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 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 $L_f(m) \approx 0$.*

The basic idea of the proof is as follows: 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

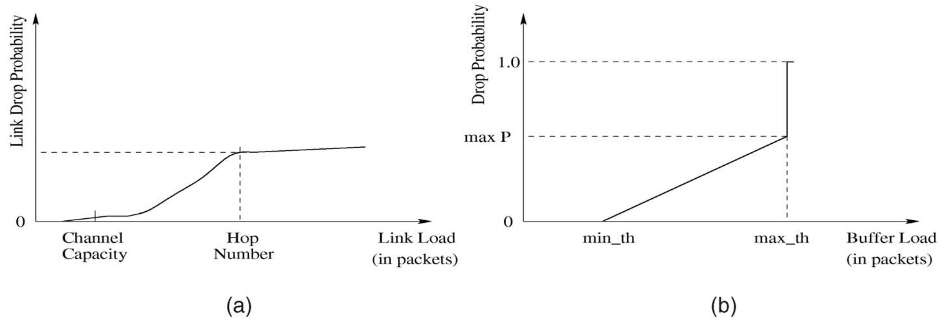


Fig. 6. Comparison between link drop and RED.

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, the 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 collisions. 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 the standard TCP suffers from throughput decrease 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 percent (Fig. 4a). Such a small drop probability is insufficient 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 percent compared with its best operating point [12].

Comparison to RED. RED is an 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 the RED drop behavior (Fig. 6).

Unlike RED, contention drop is a naturally built-in mechanism and is not specifically tuned for any higher-layer protocol. It is not useful for TCP in its current form unless the loss/load curve is appropriately tuned. In particular, 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 percent, which is too small compared with the standard parameters of RED.

However, there exists an important difference that makes contention packet drop an attractive mechanism in the ad hoc network. As shown in Fig. 6, RED drop probability reflects the local queue size, but contention drop probability reflects the 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 two simple link-layer designs to make such contention packet drops beneficial 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 versus link-layer contention to help TCP stabilize its window size around W^* for maximum throughput. In addition, Adaptive Pacing aims at improving spatial channel reuse through better coordination among contention for channel access. The combination of these two techniques is able to improve TCP throughput by as much as 30 percent.

6.1 Link RED

The analysis of 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 the TCP window size 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 its packet dropping probability when the link-layer contention level, measured by 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 an upper bound max_P. The LRED pseudocode is shown in Algorithm 1.

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: $avg\_retry = \frac{7}{8}avg\_retry + \frac{1}{8}retry$
```

LRED integrates 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].

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 pseudocode is 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
```

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 percent 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 Fig. 8c.

Using adaptive pacing alone cannot help TCP window size stays around W^*, as shown in the Fig. 8b. 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. Fig. 9 plots the results for chain topologies of various lengths, with a single TCP flow and six TCP

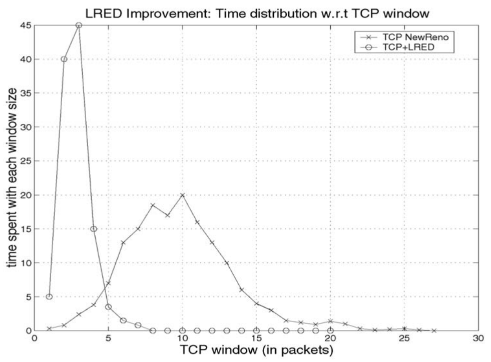


Fig. 7. *LRED stabilizes TCP window size around the optimal value*. The time distribution of instantaneous window size becomes narrower and sharper compared with TCP NewReno.

flows. In all cases, we observe that LRED+pacing enhanced link layer is able to increase TCP throughput up to 30 percent, 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 $10\% \sim 30\%$. The longer the chain, the better the throughput improvement. This is because a 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 Fig. 3a. Table 5 records the throughput and fairness gains for both flows. The fairness results are computed using the fairness index

$$\frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2},$$

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 a large unfairness between these two flows. The reason is that IEEE 802.11 favors the 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 Fig. 3b. Aggregate throughput and fairness results are recorded in Table 6, while more details for the cases of four flows are provided in Table 7. Again, we are able to achieve about $5\% \sim 10\%$ throughput gain in all cases, while significantly improving the fairness index.

7 Discussions

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

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

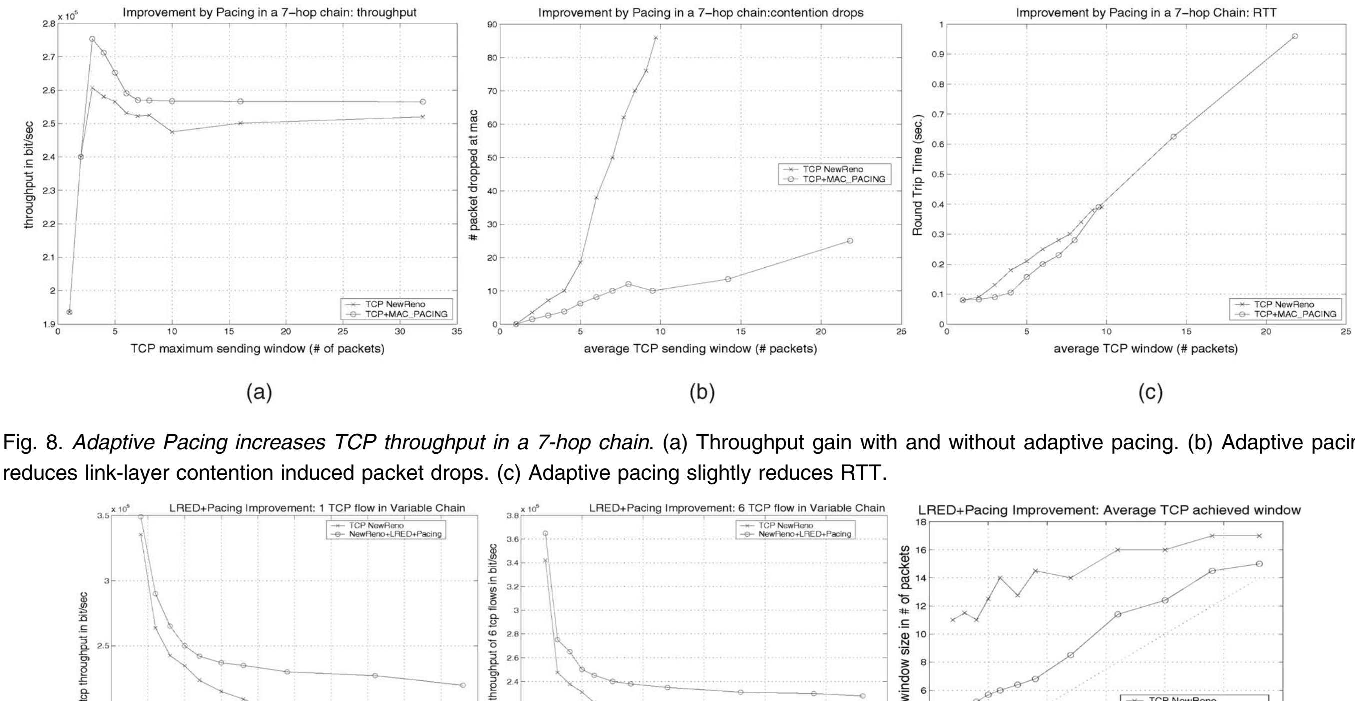


Fig. 8. *Adaptive Pacing increases TCP throughput in a 7-hop chain*. (a) Throughput gain with and without adaptive pacing. (b) Adaptive pacing reduces link-layer contention induced packet drops. (c) Adaptive pacing slightly reduces RTT.

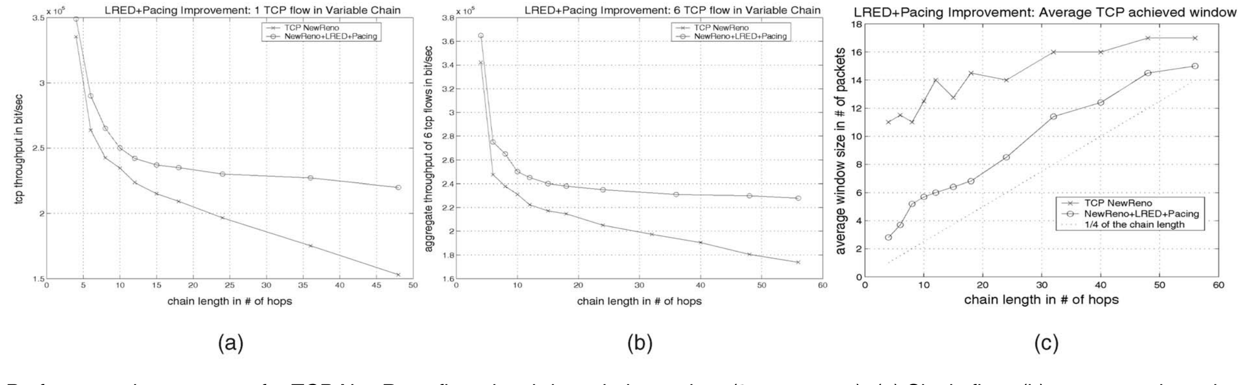


Fig. 9. *Performance improvement for TCP NewReno flows in a h-hop chain topology* ($h = 3, \ldots, 48$). (a) Single flow, (b) aggregate throughput of six flows, and (c) average window size.

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	244 Kbps	166 Kbps
flow 2	0 Kbps	153 Kbps
Aggregate	244 Kbps	319 Kbps
Fairness	0.5	0.9983

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 three 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.

Revisiting the packet loss behavior during network overload. In Section 5, we observe that almost all packet losses are due to link layer contention rather than buffer overflow in a *typical* network setting in which each node has a nontrivial buffer size allocated for the TCP flow. In this section, we further examine this issue. Our interest is on whether buffer overflow will ever happen at all and under what conditions buffer overflow may potentially dominate the packet loss.

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 varying from two 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 losses.

TABLE 7
Throughput and Fairness Comparisons between NewReno and NewReno+LRED+Pacing

	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

Four flows in 13×13 *grid.*

Fig. 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 five packets to 15 packets. When the buffer size is very small (smaller than five packets), buffer overflows dominate and contention loss is almost negligible; when the buffer size grows to 15 packets or larger, 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. The result is not surprising since the larger buffer absorbs more TCP incoming packets and reduces the probability of overflow drops. An interesting observation, shown in Fig. 10, 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 (Fig. 10b). However, the maximal buffer occupancy of all nodes is very high. This 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

TABLE 6
Aggregate Throughput and Fairness Comparison between NewReno (NR) and NewReno+LRED+Pacing (LRED+)

	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

Two, four, eight, and 12 flows in 13×13 *grid.*

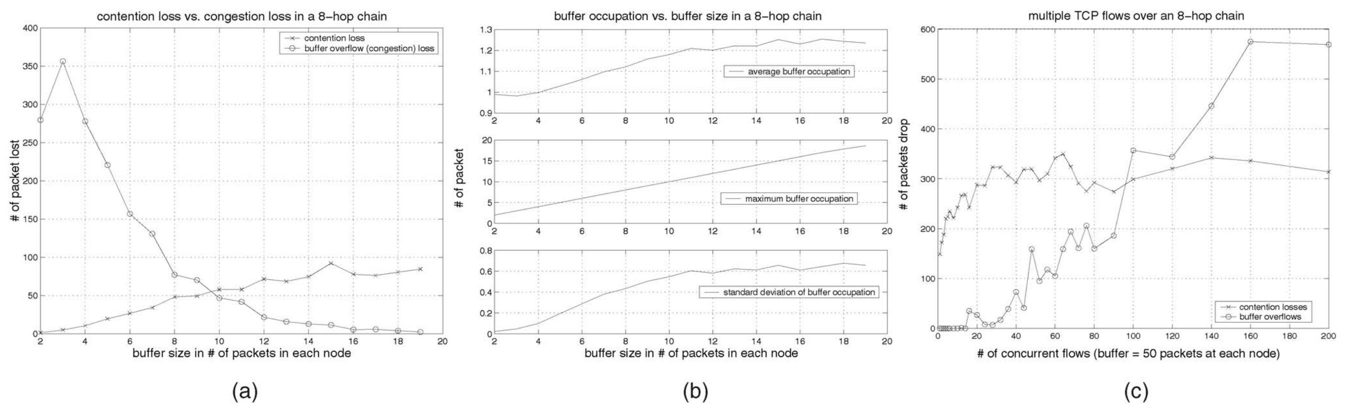


Fig. 10. *TCP loss transition*. (a) Buffer loss dominates when buffer is small; contention loss dominates when buffer is large. (b) Average and Maximum queue size of all nodes in the network. It shows that the average queue size occupancy at each node is very low. (c) Multiple concurrent TCP flows over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

may lead to back-to-back transmissions. MAC capture effect [6] due to binary exponential backoff also incurs burst transmissions over a link.

We now examine the case of multiple flows. Our interest is on the loss behavior when there are a very large number of TCP flows such that the buffer size (shared by all flows) at an intermediate node is insufficient to accommodate in-flight packets from all flows. Setting the shared buffer size to 50 packets at each node, we simulated a number of TCP flows, ranging from 20 to 200, over an 8-hop chain and measured the contention/overflow losses (Fig. 10c). Concurrent flows are introduced with the same starting and ending nodes in the chain topology. Compared with Fig. 10a, we observed that the buffer overflow losses increase significantly. However, contention losses still dominate when less than 80 concurrent flows are introduced. This happens because, during the period between network saturation (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 a buffer buildup period decreases. In addition, the interval between two overflow events also decreases. From the simulation, the overflow losses become dominant when more than 100 concurrent flows are introduced.

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

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 to keep TCP operating around the best throughput point. 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 percent for TCP flows, while the TCP semantics remain 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 Trans. Networking,* Dec. 1997.

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

[4] A. Aggarwal, S. Savage, and T. Anderson, “Understanding the Performance of TCP Pacing,” *Proc. IEEE INFOCOM,* 2000.

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

[6] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, “TCP over Wireless Multihop Protocols: Simulation and Experiments,” *Proc. IEEE Int’l Conf. Comm.,* 1999.

[7] M. Gerla, K. Tang, and R. Bagrodia, “TCP Performance in Wireless Multihop Networks,” *Proc. IEEE Workshop Mobile Computing Systems and Applications (WMCSA),* 1999.

[8] V. Bharghavan, “Performance Analysis of a Medium Access Protocol for Wireless Packet Networks,” *Proc. IEEE Performance and Dependability Symp.,* 1998.

[9] D. Allen, “Hidden Terminal Problems in Wireless LAN,” IEEE 802.11 Working Group paper 802.11/93-xx, 1993.

[10] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” *IEEE/ACM Trans. Networking,* vol. 1, no. 4, pp. 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, Sept. 1984.

[12] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee, and R. Morris, “Capacity of Ad Hoc Wireless Networks,” *Proc. ACM MOBICOM,* 2001.

[13] D.B. Johnson and D.A. Maltz, “Dynamic Source Routing in Ad Hoc Wireless Networks,” *Mobile Computing,* vol. 353, pp. 153-181, Kluwer Academic Publishers, 1996.

[14] C.E. Perkins and E.M. Royer, “Ad-Hoc on Demand Distance Vector Routing,” *Proc. IEEE Workshop 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,” *Proc. ACM MOBICOM ’99,* 1999.

[16] Z. Fu et al., "TCP over Multihop Wireless Networks," UCLA Computer Science technical report, 2002.
[17] Z. Fu, B. Greenstein, X. Meng, and S. Lu, "Design and Implementation of a TCP-Friendly Transport Protocol for Ad Hoc Wireless Networks" *Proc. IEEE Int'l Conf. Network Protocols,* 2002.
[18] J. Liu and S. Singh, "ATCP:TCP for Mobile Ad Hoc Networks," *IEEE J. Selected Areas in Comm.,* vol. 19, no. 7, pp. 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," *Proc. Third ACM Int'l Symp. Mobile Ad Hoc Networking & Computing,* 2002.
[20] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad-Hoc Networks" *Proc. MOBIHOC,* 2003.
[21] L. Yang, W. Seah, and Q. Yin, " Improving Fairness among TCP Flows Crossing Wireless Ad Hoc and Wired Networks" *Proc. MOBIHOC,* 2003.
[22] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED" *Proc. MOBICOM,* 2003.
[23] N. Gower and J. Jubin, "Congestion Control Using Pacing in a Packet Radio Network" *Proc. MILCOM,* 1982.
[24] D.S.J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing" *Proc. MOBICOM,* 2003.

Zhenghua Fu (ACM S 2001) received the BS degree from Fudan University, China, in 1995 and the MS degree from the University of California, Irvine, in 2000, both in computer science. He is currently a PhD candidate at the University of California, Los Angeles (UCLA). His research interests include wireless networks, distributed systems, and performance evaluation. He is a student member of the IEEE Computer Society.

Haiyun Luo (ACM S 2001) received the BS degree from the University of Science and Technology of China and the MS degree in computer science from the University of California at Los Angeles. He is currently a PhD candidate in the Computer Science Department, University of California at Los Angeles. His research interests include wireless and mobile networking and computing, security, and large-scale distributed systems. He is a student member of the IEEE Computer Society.

Petros Zerfos (S '99) received the BE degree from the National Technical University of Athens, Greece, and the MS degree in computer science from the University of California at Los Angeles. He is currently a PhD candidate in the Computer Science Department, University of California at Los Angeles. His research interests include wireless and mobile networking and computing, security, and large-scale distributed systems. He is a student member of the IEEE Computer Society.

Songwu Lu (M '96/ACM '99) received both the MS and PhD degrees from the University of Illinois at Urbana-Champaign. He is currently an assistant professor at the University of California at Los Angeles in the Computer Science Department. He received the US National Science Foundation CAREER award in 2001. His research interests include wireless networking, mobile computing, wireless security, and computer networks. He is a member of the IEEE Computer Society.

Lixia Zhang (SM '95/ACM '84) received the PhD degree in computer science from the Massachusetts Institute of Technology. She was a member of the research staff at the Xerox Palo Alto Research Center before joining the faculty of the University of California at Los Angeles in the Computer Science Department in 1995. In the past, she has served on the Internet Architecture Board, as cochair of the IEEE Communication Society Internet Technical Committee, as vice chair of ACM SIGCOMM, as editor for the *IEEE/ACM Transactions on Networking*, and as associate editor for *ACM Computer Communication Review*. Her research interests include large scale systems, network security, and resiliency. She is a senior member of the IEEE Computer Society.

Mario Gerla received a graduate degree in engineering from the Politecnico di Milano in 1966, and the MS and PhD degrees in engineering from the University of California at Los Angeles (UCLA) in 1970 and 1973, respectively. After working for Network Analysis Corporation from 1973 to 1976, he joined the faculty of the Computer Science Department at UCLA where he is now a professor. His research interests cover the performance evaluation, design and control of distributed computer communication systems, high-speed computer networks, wireless LANs, and ad hoc wireless networks. He has worked on the design, implementation, and testing of various wireless ad hoc network protocols (channel access, clustering, routing, and transport) within the DARPA WAMIS, GloMo projects. Currently, he is leading the ONR MINUTEMAN project at UCLA, and is designing a robust, scalable wireless ad hoc network architecture for unmanned intelligent agents in defense and homeland security scenarios. He is also conducting research on QoS routing, multicasting protocols, and TCP transport for the Next Generation Internet (see www.cs.ucla.edu/NRL for recent publications). He is a fellow of the IEEE Computer Society.