The Impact of Multihop Wireless Channel on TCP Throughput and Loss

Zhenghua Fu, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang, Mario Gerla
UCLA Computer Science Department,
Los Angeles, CA 90095-1596
Email:{zfu,pzerfos,hluo,slu,lixia,gerla}@cs.ucla.edu

Abstract— **This paper studies TCP performance over multihop wireless networks that use the IEEE 802.11 protocol as the access method. Our analysis and simulations show that, given a specific network topology and flow patterns, there exists a TCP window size W^*, at which TCP achieves best throughput via improved spatial channel reuse. However, TCP does not operate around W^*, and typically grows its average window size much larger; this leads to decreased throughput and increased packet loss. The TCP throughput reduction can be explained by its loss behavior. Our results show that network overload is mainly signified by wireless link contention in multihop wireless networks. As long as the buffer size at each node is reasonably large (say, larger than 10 packets), buffer overflow-induced packet loss is rare and packet drops due to link-layer contention dominate. Link-layer drops offer the first sign for network overload. We further show that multihop wireless links collectively exhibit graceful drop behavior: as the offered load increases, the link contention drop probability also increases, but saturates eventually. In general, the link drop probability is insufficient to stabilize the average TCP window size around W^*. Consequently, TCP suffers from reduced throughput due to reduced spatial reuse. We further propose two techniques, link RED and adaptive pacing, through which we are able to improve TCP throughput by 5% to 30% in various simulated topologies. Some simulation results are also validated by real hardware experiments.**

I. Introduction

TCP is an adaptive transport protocol that controls its offered load (through adjusting its window size) according to the available network bandwidth. It additively increases its congestion window in the absence of congestion and throttles down its window when a sign of congestion is detected. In the wired Internet, congestion is identified by packet loss, which results from buffer overflow events at the bottleneck router. However, it is unclear how well such TCP mechanisms work in a multihop wireless network; this is the focus of this work.

Multihop wireless networks have several characteristics different from wired networks. Firstly, in a typical wireless network that uses IEEE 802.11 MAC, packets may be dropped due to either buffer overflow or link-layer contention caused by hidden terminals. Such losses directly affect TCP window adaptation. Secondly, wireless channel is a scarce, shared resource. Improving channel utilization through spatial channel reuse is highly desirable. Multiple nodes that do not interfere with each other should be encouraged to transmit concurrently. How well TCP utilizes the multihop wireless channel through spatial reuse poses another important issue. A fundamental problem is that, TCP has to interact with ad hoc forwarding and IEEE 802.11 MAC in a multihop wireless network, which exhibits features quite different from the wired or wireless cellular networks. Earlier research on TCP performance over ad hoc networks has been focused on the impact of mobility-induced factors, such as link breakage and routing failures [2], [3]. However, the interaction between TCP and the underlying multihop forwarding with the IEEE 802.11 MAC, is left unaddressed.

In this paper, we study the effect of multihop wireless link on TCP throughput and loss behavior for several simple network configurations and flow patterns. Through both analysis and simulations, we reveal several interesting results. First, given a specific network topology and flow patterns, there exists a TCP window size, say W^*, at which its throughput is highest through improved spatial channel reuse. Further increasing the window size does not lead to further spatial channel reuse, but results in increased link layer contention and perceived packet losses. Second, the standard TCP protocol does not operate around W^*, and typically grows its average window much larger than W^*. Consequently, TCP experiences throughput decrease due to reduced spatial channel reuse. We observe 4% to 21% throughput reduction (from the highest throughput) in our simulated scenarios.

The suboptimal throughput of TCP can be explained by its loss behavior over the multihop wireless channel. In a wired network, all incoming packets are dropped if buffer overflows at a bottleneck. It helps TCP to quickly reduce its window size to release congestion. Multihop wireless networks exhibit different drop features. Unlike wired networks where buffer overflow dominates packet losses, most packet drops experienced by TCP are due to link layer contention, incurred by hidden terminals. Buffer overflow induced packet loss is rare, and the contention-induced packet loss offers the first sign of network overload. Our analysis and simulations further show that contention drops exhibit a load-sensitive feature: as the offered TCP packets exceed W^* and increase further, link drop probability becomes non-negligible and increases accordingly. After the offered TCP packets exceed another threshold $\bar{W}$,

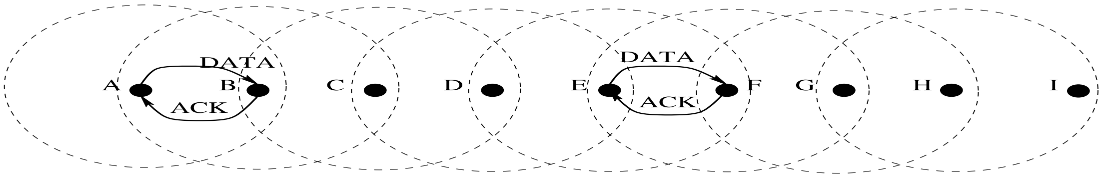

Fig. 1. *Spatial reuse and contention.*An example of 8 hop chain. Optimal spatial reuse is achieved when nodes {A, D, G} and nodes {B, C, E} are scheduled for transmission alternatively. Node D is the hidden terminal for transmission A→B.

the link drop probability saturates and flattens out. It turns out, however, that the link-layer drop probability is not significant enough to stabilize the average TCP window size around W^*. It therefore leads to suboptimal TCP throughput. Some simulation results are also validated with experiments.

Our observations also shed light on how to improve TCP performance over multihop wireless networks. In this paper, we propose two link layer techniques: a Link-RED algorithm to tune the wireless link's drop probability, and an adaptive link-layer pacing scheme to increase the spatial channel reuse. The goal is to let TCP operate in the *contention avoidance* region. These simple techniques lead to 5% to 30% throughput increase compared with standard TCP.

The rest of the paper is organized as follows. Background information is provided in Section II. A thorough study of the TCP throughput on several simple topologies and traffic patterns is in Section 3. Section IV explains the TCP throughput reduction from its loss behavior in multihop wireless networks. Section V describes and evaluates mechanisms that improve TCP performance. Section VI discusses a few issues. Section VII summarizes the related work and Section VIII concludes the paper.

II. Background

We consider a static, multihop, wireless ad hoc network. A single wireless channel is shared for transmissions, and only receivers within the transmission range of the sender can receive the packets. The IEEE 802.11 Distributed Coordination Function, the *de facto* access method used in ad hoc networks, serves as the wireless MAC protocol. In IEEE 802.11, each packet transmission is preceded by a control handshake of RTS/CTS messages. Upon overhearing the handshake, the nodes in the neighborhood of both the sender and the receiver defer their transmissions until the subsequent DATA-ACK transmissions are completed.

Failures in the transmission of control and data packets are usually caused either by the effect of the Hidden Terminal [9], or by channel errors. Specifically, the sender drops the DATA packet after sending the RTS message seven times and does not receive a CTS from the receiver. DATA packets are dropped after four retransmissions without receiving an ACK. In this work, we do not consider the wireless channel errors, as the current link-layer retransmission is sufficient to hide such errors from upper layer protocols.

Even though the RTS/CTS handshake is employed, hidden terminal problem still persists in an IEEE 802.11 ad hoc network. A hidden terminal is a potential sending node in the receiver's neighborhood, which cannot detect the sender and may disrupt the current packet transmission. We use figure 1 as an example, in which two adjacent nodes are about 200m apart. The current hardware specifies that for each wireless node, its transmission range is about 250m, its carrier sensing range is 550m, and its interference range is about 550m. The potential sending node D is a hidden terminal of the current transmission pair $A - B$. When A and B are initiating RTS-CTS handshake, D cannot hear CTS since it is out of the 250m transmission range of node B. Besides, D cannot sense A's DATA transmission since A is out of D's 550m carrier sensing range. Therefore, D may transmit to its intended receiver E at any time. When D is transmitting to E, it will cause collisions at B, since D is within the 550m interference range for B. Therefore, hidden terminal D will cause contention loss at node B.

Location-dependent contention, together with multi-hop packet forwarding, also allows for spatial channel reuse. Specifically, any two transmissions that are not interfering with each other can potentially occur simultaneously; this improves aggregate channel utilization. Figure 1 illustrates an example for spatial reuse, in which pairs of A-B and E-F may transmit simultaneously, but simultaneous transmissions from pairs of A-B and C-D will collide. Improving spatial reuse will result in increased TCP throughput.

III. TCP Throughput in Multihop Wireless Networks

In this section, we examine TCP throughput to see how well it achieves spatial channel reuse using several simple configurations including chain, grid, cross and random topologies. Our analysis, simulations[1] and real experiments show that excessive packets in flight (or equivalently, large TCP sender window size), which is the typical case for the current TCP protocol, degrades spatial channel reuse and reduces TCP throughput. In fact, the throughput reduction can be as high as 21% in the simulated scenarios. We identify the TCP window size at which TCP achieves highest throughput, for some

[1]All simulations are conducted in *ns-2*.

simple configurations. In addition, we show the quantitative difference between the highest TCP throughput and its steady-state performance. We also verified our simulation results with real hardware experiments, and found a good match between these two.

A. Chain topology

In a chain topology, TCP packets travel along a chain of intermediate nodes toward the destination. The successive packets of a single connection interfere with each other as they move down the chain, resulting in link layer contention. We study the performance of the TCP flow whose source and destination are placed at both ends of the chain respectively.

In a h-hop chain of IEEE 802.11 wireless nodes where adjacent nodes are 200 meters apart, the following analysis shows that TCP's optimal throughput is achieved when its window size is $\frac{h}{4}$, with identical packet size. For example in the chain of Figure 1: it is easy to see that nodes A and E, 4-hops away, can transmit concurrently. Therefore, for a h-hop chain, the maximum number of concurrent senders can not exceed $\frac{h}{4}$. Furthermore, since each wireless node uses IEEE 802.11 MAC that follows the DATA-ACK sequence for each packet transmission, the MAC exhibits a feature similar to the stop-and-wait protocol. Hence, the pipe size over each hop is one packet, and the total pipe-size (i.e., the packets in flight) that achieves best channel utilization is $\frac{h}{4}$. In general, if the TCP window size is below this value, it under-utilizes the channel; if it is larger, it does not further increase the channel utilization. In fact, as we show later, it reduces TCP throughput.

Our simulations show a good match between our analysis and the measured TCP throughput. Figure 2 (right) shows simulation results for a single chain. For this set of simulations, each node is 200 meters away from its immediate neighbors. The figure plots the best TCP throughput measured at chain topologies of variable lengths. To obtain the largest throughput given a chain of certain length, we artificially bound the maximum *allowed* sender window size $MaxWin$ for TCP, in the range of 1 to 32 packets[2] At each $MaxWin$, we run a TCP flow for 300 seconds and measure the achieved throughput. The highest TCP throughput is selected out of different $MaxWin$ values. The results plotted in Figure 2 show that the TCP window size W^* (that achieves maximal throughput) and $\frac{h}{4}$ match reasonably well, particularly for longer chains with $h > 20$. 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. However, as the chain becomes longer, the uneven packet distribution, or equivalently the queue size deviation at each node, tends to become smaller, as shown in Table I. Therefore, best spacial reuse is more likely to achieve at $\frac{h}{4}$ for longer chains.

[2]It is similar to enforcing TCP flow control. Note that MaxMin only *bounds*, but not necessarily represents the actual number of packets in flight.

Chain Len. (hops)	4	7	10	16	48
Buffer Deviation	1.45	1.31	1.23	1.10	1.05

TABLE I

Buffer deviation among nodes in chain topologies.

$MaxWin =$ in # hops	1	2	4	8	16	32
Avg. TCP wnd.	1	2	3.9	7.1	9.2	9.6

TABLE II

TCP average window w.r.t. $MaxWin$ in 7-hop chain.

Our analysis applies with different packet sizes, as long as successive packets of the TCP flow are of identical size. The results shown in figure 2 (left) confirm that the best TCP window W^* is invariant to different packet sizes of 576B, 1KB, and 1460B. However, the throughput achieved by TCP is different due to different per-packet overhead.

1) Suboptimal TCP Throughput: If we let the TCP window size grow arbitrarily (i.e., leave $MaxWin$ unbounded) – the typical case in reality, we observe that TCP experiences throughput reduction. Figure 2 shows that the throughput reduction is about 4% from the highest achievable throughput in a 7-hop chain. As the chain becomes longer, the observed throughput reduction can be as high as 10%. Also, Table II shows that for the case of unbounded $MaxWin$, the average TCP sender window size eventually stabilizes at 10 packets in a 7-hop chain. This steady-state window size is about 4 times as large as its $W^* \approx 2$. Since more packets are in flight, most of these packets will be backlogged at intermediate nodes, more or less evenly distributed across all nodes. These buffered packets at each node will increase contention loss due to the hidden terminal effect, thus leading to throughput reduction. In the next section, we will see this throughput reduction due to excessive TCP window size is much more significant in more complex topologies, resulting in as high as 15% to 21% throughput reduction in grid or random topologies.

2) Verification using Real Hardware: As a rough check on the above simulations, Figure 2 (middle) shows results measured on real hardware over a 7-hop chain. The hardware was configured to mimic the simulation parameters used in Figure 2 (Left) as closely as possible. The radios are Lucent ORiNOCO wireless cards, operating in the ad hoc mode at 2Mbps. During the experiment, 8 notebooks form a 7-hop chain and only neighboring nodes are within the transmission range. Manual routing is used for packet forwarding. As we can see from Figure 2 (Middle), two curves of the measured TCP throughput and the simulated value match fairly well. It demonstrates that our *ns-2* simulations are accurate enough to model the reality. Specifically, the average difference between

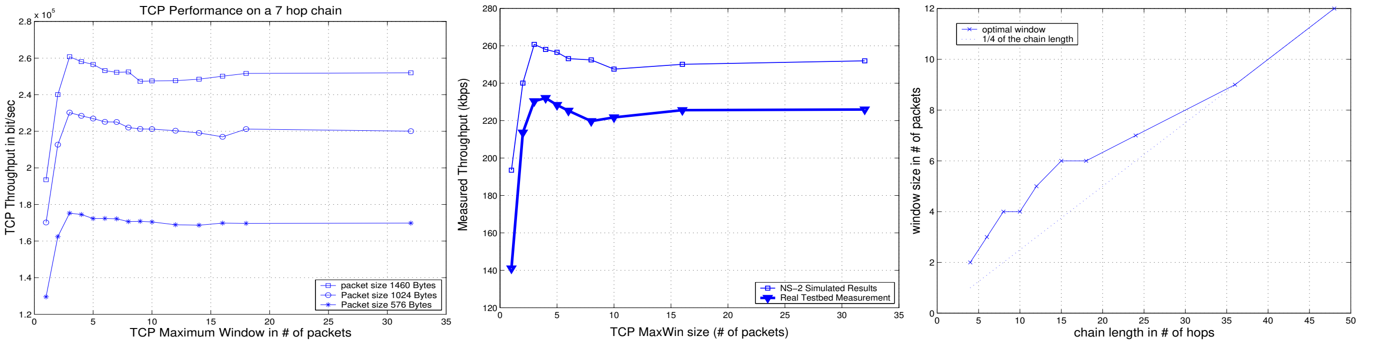

Fig. 2. *Best TCP Throughput is achieved when window size is about 3 in an 7-hop Chain*. Left: Single TCP throughput for different packet sizes Middle: Real Testbed experiment measurement on single TCP throughput for different MaxWin with packet size 1460 bytes. Right: TCP optimal window size v.s. chain length.

the measured TCP throughput and the simulated results is less than 10%, and the difference between the curves (with respect to $MaxMin$) is almost negligible.

B. *More complex topology and flow patterns*

The results described above show that there exists a value for the TCP window size in the chain topology, so that channel reuse is maximized. It results in best channel utilization and consequently highest TCP throughput. In order to gain a general understanding of the TCP throughput, we expand our study to scenarios of more complex topologies and TCP flow patterns, including cross, grid and random topologies, and up to 12 flows. We keep the simulation parameters the same as in Section III.A, unless explicitly specified.

In all cases, we observe that there exists a TCP window size that achieves the best throughput, and TCP generally experiences 15% to 21% throughput degradation from its highest achievable value. The following provides a short summary:

a) Cross topology: In the cross topology shown in Figure 3, we run two TCP flows from node 0 to node 6 and from node 7 to node 12 respectively. The simulation results in Table III show that, the best window W^* for each flow is 2. In contrast, our measured aggregate TCP window is 12 packets at steady state, and 20% throughput reduction is observed.

b) Grid topology: On the 13 X 13 grid topology shown in Figure 3, we run 4, 8 and 12 TCP flows respectively. In each of these three cases, flows are spaced evenly in each direction. The results are summarized in Table III. In all cases, measured TCP windows are significantly larger than the size for highest achievable throughput, and throughput is consequently up to 21% lower as shown in the 12-flow case.

c) Random topology: We also run extensive simulations with random topologies, in which 200 nodes are placed within a rectangular area of size 1000×2500. 20 TCP flows are placed in the network, with their sources and destinations randomly selected. We still observe the existence of the best TCP window when the highest aggregate throughput is achieved, given each specific topology and flow settings. And the regular

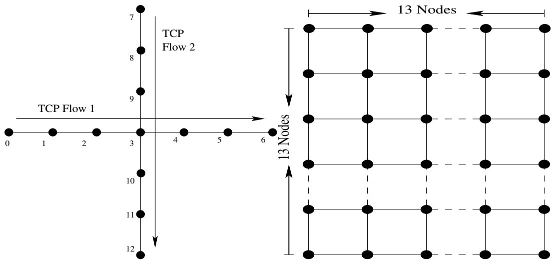

Fig. 3. *More complex simulated topologies*. Left: cross topology with 13 nodes. 200 meter distance between two adjacent nodes. 2 TCP flows in each direction. Right: 13x13 grid topology, 200 meter distance between horizontal or vertical adjacent nodes .

TCP throughput suffers about 15% reduction compared with the maximum, as shown in Table III.

C. *Summary*

All simulations and analysis confirm that for a given topology and traffic pattern, there exists a window size W^*, at which TCP achieves the best possible spatial channel reuse, and consequently increases the channel utilization. W^* is in general a function of the node topology and flow patterns, though we are able to derive it only for simple configurations assuming perfect coordinations among nodes.

However, if we let TCP $MaxWin$ go unbounded (i.e., the normal TCP behavior), a common observation for all topologies and flow patterns examined before is that TCP experiences a 4% to 21% decrease in its throughput. For example, The TCP throughput difference between the highest and steady-state in random topology, given in table III, is about 15%. This shows that TCP normally operates suboptimally, and there is clearly room for improvement. The simulations also show that this is mainly due to the fact that the average TCP window size is much larger than the best W^*, thus causing more packet drops and reduced throughput.

In order to understand why TCP's stead-state average window size is much larger than W^*, we examine TCP loss behavior in the multihop wireless networks.

Topology	Number of flows	Optimal Throughput (Kbps)	Measured Throughput (Kbps)	Optimal Window	Average measured Window
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 III

Aggregate TCP throughput and window size for all flows in more complex topologies. In all cases, TCP stable performance is suboptimal.

IV. TCP LOSS BEHAVIOR

In this section, we investigate TCP's throughput reduction from the packet loss's perspective. It turns out that, unlike the wired Internet where all incoming packets will be dropped once buffer overflows, the drop probability in a multihop wireless network is not significant, when the TCP window reaches and exceeds W^*. The low drop probability, after the maximal link capacity is reached, results in an average TCP window size much larger than W^*. This in turn causes excessive packet drops due to hidden terminal effects, contributing to throughput reduction of TCP.

A. Link-layer contention v.s. buffer overflow

In the wired Internet, packet losses are mainly due to buffer overflows at the bottleneck router. In multihop wireless networks, if we tentatively ignore the loss due to interferences (since the link-layer retransmissions can shield almost all such losses from TCP), packet drops may be possibly caused by buffer overflows or contentions due to hidden terminals.

The following analysis shows that most packet drops in the multihop wireless network are due to link-layer contentions and buffer overflows are rare. For the simple chain topology of Figure 1, consider the case where each node has exactly one packet in its buffer – a common case as the measurement in Table IV shows. Although buffer occupancy is low, packets can still be dropped due to link contention. Assume that node D first starts transmitting its buffered packet to E after a random backoff period, by initiating the RTS-CTS-DATA-ACK handshake with node E. After hearing from node D, node B will defer until the packet transmission is completed (recall that D is within the carrier-sensing range of B). During this period, if node A which is unable to detect the ongoing transmission between D and E, initiates an RTS message to B, the latter will not be able to reply. If A receives no response from B after seven retransmissions of the RTS, it drops its head-of-line packet, according to the IEEE 802.11 MAC. What we just described is essentially a case of the Hidden Terminal, and it results in packet drops even at low buffer occupancy (only one packet). In contrast, wireline networks do not experience link drops, but suffer from buffer overflows. In the same chain topology, the wired counterpart will experience buffer overflow drops at the first node of the chain.

Node ID	#1	#2	#3	#4	#5	#6	#7	#8	#9
Max. Buf	9	11	13	14	16	15	12	10	6
Avg. Buf	0.4	0.8	1	1.9	1.9	1	0.9	0.7	0.3

TABLE IV

Buffer occupancy in packets. No packet drop due to buffer overflow.

The simulations for an 8-hop chain confirm the above analysis. In the 300-second simulation run, all 165 TCP drops out of 12349 transmissions are due to link drops, and none is caused by buffer overflows. The average buffer occupancy at each node, shown in Table IV, is only about 1$\sim$2 packets.

We also conduct extensive simulations using more complex configurations and flow patterns including grid, cross, and random topologies. The simulation results show that, in general, buffer overflows are rare, and most packet drops experienced by TCP are due to link contentions. However, in the corresponding wired setting, all drops happen due to buffer overflows.

B. Load-sensitive Drop Probability

In a multihop wireless network, it is the link-layer contention induced packet loss that offers the first sign of network overload. Therefore, the next question is how this loss occurs as the network overload builds up. The drop probability will decide the average TCP window size at which TCP stabilizes eventually.

"Network overload" actually has different implications in the multihop wireless context. In the wireline counterpart, a busy link or a built-up queue at the bottleneck signifies a condition of network overload. In a multihop wireless network, network overload is *no longer* a bottleneck link property, but a *shared feature* of multiple links. It cannot be detected within a single hop alone. Informally, we can define it as the state

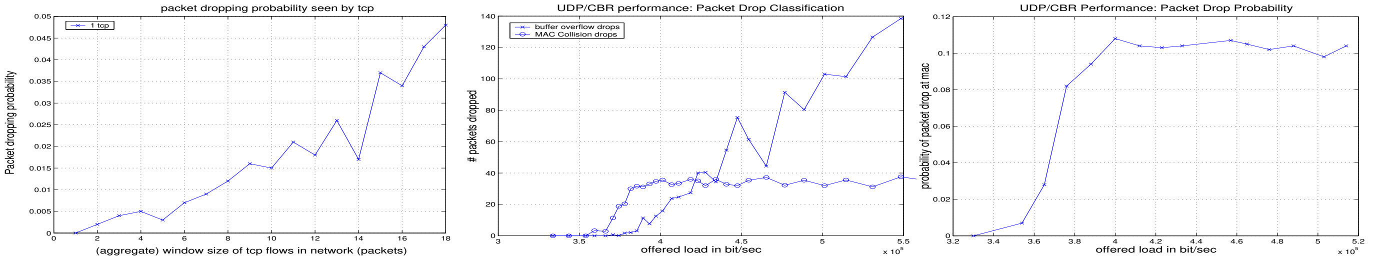

Fig. 4. *RED-like Link Drop Behavior. Contention loss happens well before buffer overflows.*. Left: Contention loss v.s. buffer-overflow loss; Middle: Contention loss as a function of offered load. Right: Throughput variation as contention builds up.

of the network where the incoming traffic is more than the network can handle, while operating at the point of maximum utilization; this is met when the optimal spatial channel reuse is achieved.

Our simulations show that link drop exhibits graceful drop feature as network load increases. This is clearly different from the tail-drop gateway of the Internet (in which the drop probability becomes one once the buffer is full and the load further increases).

Figure 4 (left) plots the link drop probability as a function of the TCP window size. We use the TCP window size to characterize the current network load. The graph shows that contention drop probability gradually increases as more and more packets are injected into the network. The figure clearly presents graceful link drops, with the increase of network load. To better understand the underlying mechanics of this link drop behavior, we also experimented with a CBR flow using UDP. From the results of contention loss probability shown in Figure 4 (right), a similar gradual link drop behavior is exhibited as the network load increases. The figure also depicts the saturation of the contention drop probability, which happens when the load is beyond a certain threshold.

We further ran extensive simulations with more complex topologies including cross, grid, and random layout and more complex flow patterns. The detailed results, not included here due to space constraints, provide findings similar to the above.

To summarize, simulations show that multihop wireless links exhibit graceful drop behavior in the presence of network overload: When the TCP window size is smaller than W^* (which allows for best spatial reuse and channel utilization), the drop probability is negligible. When the window size is bigger than W^*, drop probability starts to increase. When the window size reaches another threshold $\bar{W}$, the drop probability saturates. From Figure 1, we can also explain why the drop probability eventually saturates as the load increases. As far as contention drop probability is concerned, the *number of non-empty buffers* is the key, while the absolute number of packets in the buffer does not matter much. When all the buffers are non-empty, the network reaches highest-level contention. The value of link drop probability is governed by the 802.11 MAC protocol, and not by the number of packets in each buffer – the latter aspect is only related to TCP congestion control and decides on how long the TCP flow will stay in the link loss phase.

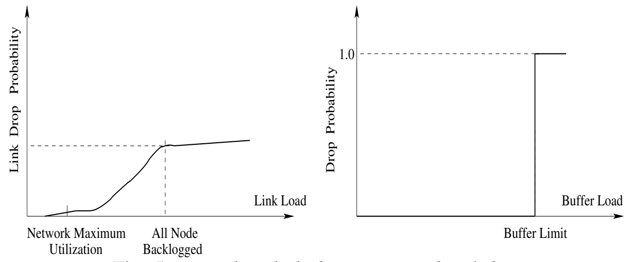

Fig. 5. *Wireless link drop v.s. wired tail drop*

C. An Analysis of Link Drop Probability in a Random Topology

The following analysis shows why multihop wireless links exhibit graceful drop as the load increases. Consider an N-node random topology. For simplicity, assume that each node has the same probability to become backlogged (i.e., has packets to deliver). We denote the total number of *backlogged* nodes in the network as m and all of them are ready for transmission. We will show that as m increases, the link drop probability will increase but saturate at a certain network load.

The m backlogged nodes in the network will use the RTS-CTS-DATA-ACK handshake of 802.11 protocol for data transmission. If m is very large, not each of the m nodes is able to successfully transmit its data packet, due to collisions. Among m nodes, let us denote that $c(m)$ nodes are able to successfully initiate RTS request. This happens only if each of these $c(m)$ nodes detects clear channel through its carrier sensing. Note that $c(m) \leq m$ for large m. Due to hidden terminal problem [8], [9], not all of these $c(m)$ nodes may successfully transmit their DATA packets. Let us denote the number of nodes that are able to successfully transmit the DATA packets as $b(m)$. It is easy to see that $b(m) \leq c(m)$.

Using Markov chain models, we can derive each node's packet loss probability P_l as follows (The proof is given in the technical report [13]):

$$P_l(m) = \frac{\frac{b(m)}{m}}{1-\left(1-\frac{b(m)}{c(m)}\right)^{r+1}} \cdot \left(1-\frac{b(m)}{c(m)}\right)^r \quad (1)$$

where $r = 7$ is the maximum retry count for RTS in 802.11 MAC.

The following three corollaries characterize the graceful drop behavior of multihop wireless links, 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:

Corollary 4.1: 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 $P_l \approx 0$.

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 (1), the drop probability over each link is $P_l \approx 0$. 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:

Corollary 4.2: When the network is overloaded (i.e., the number of backlogged nodes m is greater than B^*), the link drop probability P_l increases as m increases.

We still use (1) 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 $\frac{dP_l(m)}{dm} > 0$. 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:

Corollary 4.3: Once network is heavily loaded in the sense that $m > C^*$, then the link drop probability P_l 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 $P_l(m)$ remains statistically flat according to (1).

Our simulation results show a good match with the above simplified analysis.

D. Why TCP suffers from throughput reduction?

Now we use the graceful link drop behavior to explain why standard TCP suffers from throughput reduction as described in section III. TCP achieves highest throughput at the window size W^* that maximizes spatial reuse. The analysis and simulations of Sections IV-A and IV-B indicate that, the packet drop probability is close to zero at window size W^*. When the TCP window size exceeds W^*, the link drop probability starts to increase. However, due to the graceful drop feature, it is not sufficient to keep TCP from maintaining its average window size around W^*. Instead, the average window size W_{avg} is much larger than W^*. Since the W_{avg} inflight packets are statistically distributed among intermediate nodes evenly, link drop probability is non-zero. The link contention in turn reduces the chance of maximal spatial reuse. Consequently, TCP suffers from throughput degradation from the highest achievable throughput (i.e., when best spatial reuse is obtained).

V. Improving TCP Performance

This section describes two techniques to improve TCP performance over multihop wireless networks. The link RED technique seeks to react earlier to link overload. The adaptive pacing technique seeks to improve spatial reuse. The combination of these two techniques is able to improve TCP throughput by as much as 30%.

A. Distributed Link RED (LRED)

Our Link RED (LRED) algorithm is based on the observation that TCP can potentially benefit from the built-in dropping mechanism of the 802.11 MAC. The main idea is to further tune up wireless link's drop probability, based on the perceived link drops. While the wired RED provides a linearly increasing drop curve as the queue exceeds a minimum value min_th, LRED does so as the link drop probability exceeds a minimum threshold.

In LRED, the link layer maintains the average number of the retries for recent packet transmissions. The head-of-line packet is dropped/marked from the buffer with a probability based on this average number. At each node, if the average number of retries is small, say less than min_th, which means that the node is rarely hidden, packets in the buffer are not dropped/marked. When it gets larger, the dropping/marking probability is computed, and the minimum value of the computed drop probability and a maximum bound max_P is used.

A feature of this algorithm is that it can integrate 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 further improved, by paying the moderate cost of a slightly more complex link-layer design.

To summarize, LRED is a simple mechanism that, by monitoring a single parameter –the average number of retries in the packet transmissions at the link-layer, accomplishes three goals: a) It helps to improve TCP throughput, b) It provides TCP an early sign of network overload, and c) It helps to improve inter-flow fairness, as shown in the simulations of Section V-C.

Algorithm 1 L-RED: LinkLayerSend(Packet p)

Require: avg_retry is the average MAC retries for each packet
if $avg\_retry < min\_th$ **then**
  $mark\_prob \leftarrow 0$
  $pacing \leftarrow ON$
else
  $mark\_prob = min\{\frac{avg\_retry-min\_th}{max\_th-min\_th}, max\_P\}$
  set $pacing$ OFF
end if
mark p with $mark\_prob$
MacLayerSend(p, $pacing$)
$retry$ = GetMacRetries()
$avg\_retry = \frac{7}{8}avg\_retry + \frac{1}{8}retry$

B. Adaptive Pacing

Our second technique seeks to take an adaptive pacing approach at the link-layer. The goal is to improve spatial channel reuse, by distributing traffic among intermediate nodes in a more balanced way, while enhancing the coordination of forwarding nodes along the data path. This design works in concert with the 802.11 MAC.

In the current 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 receiver problem [8] persists due to lack of coordination between nodes that are *two* hops away from each other. Adaptive pacing solves this problem, without requiring nontrivial modifications to the 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 receivers, and extends the range of the link-layer coordination from one hop to two hops, along the packet forwarding path.

The algorithm works together with LRED as follows. Adaptive pacing is enabled by LRED. When a node finds its average number of retries to be less than min_th, it calculates its backoff time as usual. When the average number of retries goes beyond min_th, adaptive pacing is enabled and the backoff period is increased by an interval equal to the transmission time of the previous data packet. This way, a better coordination among nodes is achieved under different network load.

Algorithm 2 Adaptive Pacing

Require: *extra_Backoff* $= 0$
if received ACK **then**
  random_Backoff $\leftarrow$ *ran_backoff(cong_win)* {DATA transmission succeeded. Setup the backoff timer}
  if $pacing$ is ON **then**
    extra_Backoff = *TX_Time(DATA)* + *overhead*
  end if
  backoff $\leftarrow$ *random_Backoff* + *extra_Backoff*
  start *backoff_timer*
end if

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

TABLE V

Throughput and Fairness Comparison between NewReno and NewReno+LRED+PACING in Cross Topology.

C. Performance Evaluation

This section evaluates the performance of TCP NewReno over the 802.11 standard and our enhanced link layer, on chain, cross and grid topologies. We study the scenarios of single and multiple TCP flows.

Chain topology The results for chain topologies of various lengths, both for a single flow and six flows, are plotted in Figure 6. In all cases, we observe that our LRED + adaptive pacing enhanced link layer is able to boost TCP throughput up to 30%. Furthermore, our modifications force TCP to stabilize at a window size close to the best value. For chains longer than 15 hops, our techniques are again able to achieve a gain of 10%$\sim$30% in throughput; and as it appears, the longer the chain, the better the throughput improvement. This is because our pacing mechanism helps TCP to improve spatial channel reuse; the longer the chain, the more it benefits from the channel reuse.

Cross Topology Two flows over a 13-node cross topology were simulated in this experiment. Table V presents the throughput and fairness results 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]. 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 results in large unfairness; this is mainly due to the well-known MAC-layer capture effect of the IEEE 802.11 protocol [6].

Grid Topology Finally, for the grid case, we simulated 2, 4, 8 and 12 flows over a 13$\times$13 grid topology (Figure 3). Aggregate throughput and fairness results are summarized in Table VII, while more details for the specific case of 4 flows, 2

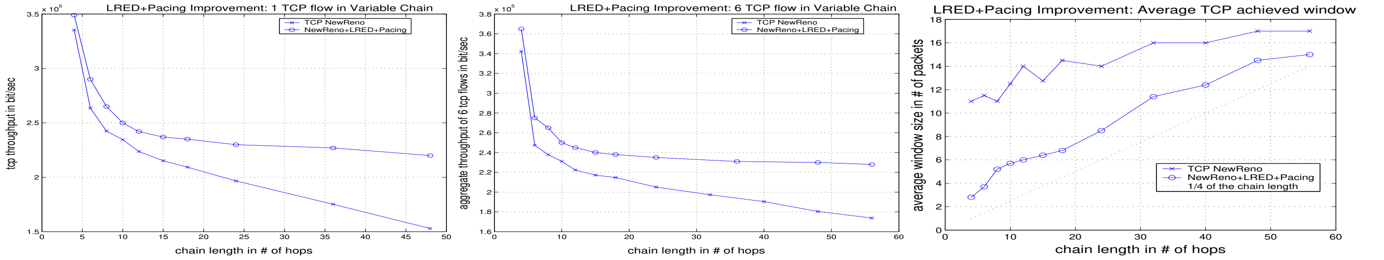

Fig. 6. *Throughput improvement over TCP NewReno in h-hop chain topologies ($h = 3, \ldots, 48$).* Left: Single flow throughput; Middle: 6-flow aggregate throughput; Right: Average window size.

	TCP NewReno w/standard 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 VI

Throughput and Fairness Comparisons between NewReno and NewReno+LRED+PACING. 4 flows in 13×13 Grid.

in each direction, are provided in Table VI. Again, in all cases, we are able to achieve about 5%∼10% throughput increase, while significantly improving the fairness index.

VI. Discussions

This section further discusses several issues.

Other TCP variants Based on the results shown in Sections III and IV, it seems that TCP Vegas, which gauges the throughput before increasing its congestion window size, may work better in the ad hoc settings. Our experiments show that TCP Vegas and TCP NewReno perform comparably in short hops (≤ 6). However, in longer hops (≥ 9), TCP Vegas performs 10%∼20% worse than NewReno. The main reason is that TCP Vegas keeps its window size very small (e.g., about 3 packets even in a 16-hop chain), and it cannot fully utilize the multihop wireless channel. TCP NewReno seems to be the best TCP variant that we could compare with.

Variable packet size In most analysis and simulations presented in previous sections, we assume identical packet size. If the packet length varies, our study shows that these results still hold: there exist a TCP window size W^* that achieves highest throughput but TCP is unable to achieve it. However, the relationship between W^* and the network topology is more complicated.

Implications of our results The built-in load-sensitive link drop property can assist in controlling the network overload, and this is particularly true for long-hop flows. However, the drop behavior has a lot of randomness and is hard to control. In our design, we intend to reduce the randomness and fine-tune the drop behaviors through the adaptive pacing mechanism, and use LRED to limit contentions.

VII. Related Work

TCP over wireless cellular networks has been an active research topic. [1] presents a summary of TCP optimization over such networks, where the single wireless link is the first/last hop in the data path. The focus is to hide wireless channel errors from TCP. Due to the built-in link layer retrnasmission mechanism of the IEEE 802.11 protocol, most channel-errors can be made transparent to upper layers and channel error-induced packet losses are negligible. We study another problem, i.e., link-contention induced packet losses, which do not exist in the single-hop cellular wireless networks.

There are several recent studies on TCP performance over ad hoc networks. [2] investigates the effect of mobility-induced link breakage on TCP performance. It studies the impact of routing dynamics of DSR protocol and mobility pattern upon TCP protocol. The authors further proposed an *explicit link failure notification* (ELFN) technique to help TCP differentiate link failure-induced losses from congestion losses. A more recent work [3] examines various performance aspects of TCP-ELFN. We focus on TCP performance in static ad hoc networks instead, and study the impact of multihop, shared wireless medium upon TCP. In fact, we use pre-configured, manual routing in our study to minimize the impact of routing dynamics.

[6], [7] study the effect of TCP ACK traffic on TCP performance, where severe unfairness and capture effect caused by the MAC backoff mechanism are reported. The work is conducted in the context of two MAC protocols: CSMA and FAMA, and 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 work showed that even with link-layer ACKs in IEEE 802.11 MAC, TCP still suffers packet losses due to link-layer contentions. In fact, all packet drops are due to such contentions, and buffers never overflow in our simulated scenarios. We further show that TCP typically operates in a sub-optimal region that does not lead to best utilization of the shared wireless channel.

	NR Aggregate	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 VII

Aggregate throughput and fairness comparisons between NewReno and NewReno+LRED+PACING with 2, 4, 8 and 12 flows in grid topology.

Our study of the best TCP window size is also related to [12], where the capacity of an ad-hoc network is analyzed. It is also shown in [12] that there exists an optimal throughput for a UDP/CBR flow when varying its offered load. In particular, over an ad hoc chain, the optimal throughput achieved is approximately 1/4 to 1/7 of the *effective* wireless bandwidth that maximizes the spatial reuse along the chain. When the offered load exceeds this point, the throughput of the UDP flows sharply decreases. In this paper, we study the interaction between TCP and the underlying IEEE 802.11 DCF MAC. Based on our analysis of the link layer drop probability, we propose two mechanisms, i.e., LRED and Adaptive Pacing to fine-tune the dropping behaviors and improve the TCP throughput.

VIII. Conclusion

Ad hoc networks hold great promise in pervasive computing and wireless sensor networks. TCP is a natural choice for reliable data delivery in this scenario. This work systematically studies the impact of shared medium on TCP performance. Our results show that over multihop wireless network, the throughput of TCP improves significantly if it operates around certain window that achieves highest spacial channel reuse. However, TCP typically stabilizes around a much larger window, thus resulting in reduced throughput. To gain more insight, we further studied the packet losses under different load conditions, and found out that network nodes in a multihop wireless setting collectively exhibit a distributed, graceful drop feature. As the network load increases, the drop probability increases but eventually saturates. We also propose two link-layer techniques, LRED and Adaptive Pacing, which improve the throughput of standard TCP flows by as much as 30%.

References

[1] 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

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

[3] J. P. Monks, P. Sinha and V. Bharghavan, "Limitations of TCP-ELFN for Ad Hoc Networks," *MOMUC* 2000

[4] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing," *IEEE INFOCOM* 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," *Proceedings (MobiCom '01)* Rome, Italy, July 2001.

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