Low Delay Marking for TCP in Wireless Ad Hoc Networks

Choong-Soo Lee, Mingzhe Li, Emmanuel Agu, Mark Claypool, Robert Kinicki
{clee01,lmz,emmanuel,claypool,rek}@cs.wpi.edu
Computer Science Department at Worcester Polytechnic Institute
Worcester, MA, 01609, USA

Abstract— **End-hosts on wireless ad hoc networks typically use TCP as their transport layer protocol. Being designed for wired networks, TCP can perform poorly over wireless networks. Work that has proposed ways to improve TCP performance over wireless networks has concentrated primarily on improving TCP throughput only. Emerging applications, such as interactive multimedia and network games, require reduced delay at least as much as increased throughput. In this paper, we propose LDM[1], an IP layer queue marking mechanism that estimates the number of hops and flows at each wireless node to computes the optimal marking probability. We present simulation results and analysis that demonstrate that LDM greatly reduces the round-trip time of TCP connections while improving throughput under many configurations.**

Keywords— **TCP, Performance, Delay, ECN, Marking, IEEE 802.11, MAC, Ad Hoc, Multihop, Wireless**

I. Introduction

Wireless ad hoc networks currently carry traffic using the Transmission Control Protocol (TCP), the de facto standard for most applications. However, TCP was designed for wired networks and thus can perform poorly in ad hoc wireless environments including IEEE 802.11 [1] networks, as noted in many research papers [2], [3], [4], [5], [6], [7], [8].

The Media Access Control (MAC) layer of IEEE 802.11 wireless ad hoc networks uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with a Request-to-Send/Clear-to-Send (RTS/CTS) mechanism to avoid data packet collisions. The RTS/CTS pre-exchange greatly reduces data packet collisions due to the hidden terminal problem but also causes some side effects when the MAC layer becomes over-saturated. The primary reasons for TCP performance degradation are the contention delays and contention drops that the RTS/CTS mechanism causes, which have been identified as RTS/CTS jamming [9] and RTS/CTS-induced congestion [10].

Previous research on the improvement of TCP performance over wireless ad hoc networks includes the investigation of link breakage and routing failure related problems such as in [2], [4], [6], link layer solutions, such as in [3], [8], MAC layer solutions, such as in [5], and TCP protocol modifications, such as in [7]. A few recent papers present techniques to improve TCP throughput by controlling the total number of packets in flight. Fu *et al.* [8] present a link layer approach, Link-RED (LRED), that limits the TCP sending window to reduce MAC layer collisions, and Adaptive Pacing (AP), which adds a random delay when sending packets to reduce the probability of MAC layer collisions. Chen *et al.* [5] attempt a similar improvement by limiting TCP's window size directly.

Most proposed improvements to TCP are link layer optimizations which are difficult to deploy since they are tied to network card-specific device drivers rather than the operating system. Furthermore, improved throughput has been the most common metric, especially as traditional applications such as File Transfer (FTP) and electronic mail demand maximum throughput. However, emerging applications such as streaming multimedia and network games, demand lower round-trip times. Moreover, we project with the steady increase in maximum wireless network bandwidth (currently up to 54 Mbps for the 802.11g standard), end-to-end delays will become increasingly important relative to throughput.

We propose a technique we call Low Delay Marking (LDM) which modifies the packet queue manager at the IP layer to improve TCP performance. The goal is to improve round-trip times, loss rates and collisions with minimal degradation (and perhaps even some improvement) to TCP throughput. Our choice of an IP layer modification is to facilitate easier deployment since operating system upgrades and patches can be used independently of a hardware change in the wireless network devices.

The rest of this paper is organized as follows: Section II reviews background literature such as the hidden terminal problem, LRED and AP; Section III focuses on the LDM mechanisms; Section IV describes the simulation setup and analyzes the simulation results and compares them to AP; and Section V summarizes our findings and mentions some possible future work.

[1] LDM stands Low Delay Marking

II. BACKGROUND

This section briefly introduces background relevant to our research, including the hidden terminal problem, TCP with Explicit Congestion Notification (ECN), and the Link RED and Adaptive Pacing algorithms for dealing with wireless MAC layer retransmissions.

A. *The Hidden Terminal Problem*

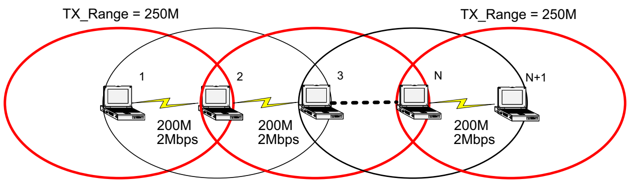

Fig. 1. Simulation Topology

Figure 1 illustrates the hidden terminal problem in IEEE 802.11 wireless Local Area Networks. Node 1 and node 3 are within the transmission (or power) range of node 2, but are out of range of each other. Hence, while they can both receive transmissions from node 2, they cannot receive each other's transmissions. If node 1 and node 3 simultaneously start transmission to node 2, their transmissions collide.

To mitigate the hidden terminal effect, IEEE 802.11 [1] mandates an RTS-CTS pre-exchange before any data packets can be sent. In the above scenario, if node 1 senses an idle channel and sends an RTS to node 2, its intended destination node, all nodes within node 1's range hear the RTS and backoff. When node 2 responds with a CTS message, all nodes within nodes 2 range, including node 3, become aware of the imminent data transmission and also backoff, thus solving the hidden terminal problem. RTS and CTS frames also contain duration information, called a Net Allocation Vector (NAV), on how long the data exchange will take. This allows other nodes that hear either the RTS or CTS frames to determine how long the channel will be busy, and hence backoff accordingly.

An RTS sender may receive no CTS either because its RTS packet collided with another transmission at the receiver or because the receiver's NAV indicated that the network is not available. The sender of the RTS packet eventually times out and does an exponential backoff before re-sending the RTS, up to a limit of seven times, as prescribed by the IEEE 802.11 standard. Since RTS and CTS packets are small in comparison to data packets, the wasted bandwidth incurred when RTS and CTS packets collide is minimal. However, RTS collisions increase network load which ultimately results in larger contention delays due to repeated exponential backoffs and RTS contention drops when the number of retransmissions exceeds the specified threshold of seven.

Moreover, TCP, left unconstrained, can saturate the MAC layer and cause numerous RTS collisions and drops, producing less than optimal throughput and high round-trip times. To briefly illustrate that an unconstrained TCP produces less than optimal performance, and hence motivate our proposed enhancements, we ran NS-2 simulations that restricted the maximum window size of a single TCP sender in a 7-hop ad hoc network using IEEE 802.11. Figure 2 shows the effect of the maximum TCP window size on throughput and round-trip time. The default, unconstrained TCP flow would have a window size towards the far right of the graphs. However, by constraining the window size to a maximum window size of about 3, TCP achieves higher throughput. Equally important, we can see that the round-trip time also increases as the maximum TCP window size increases, a measure of performance that has not been examined by previous researchers [8], [5].

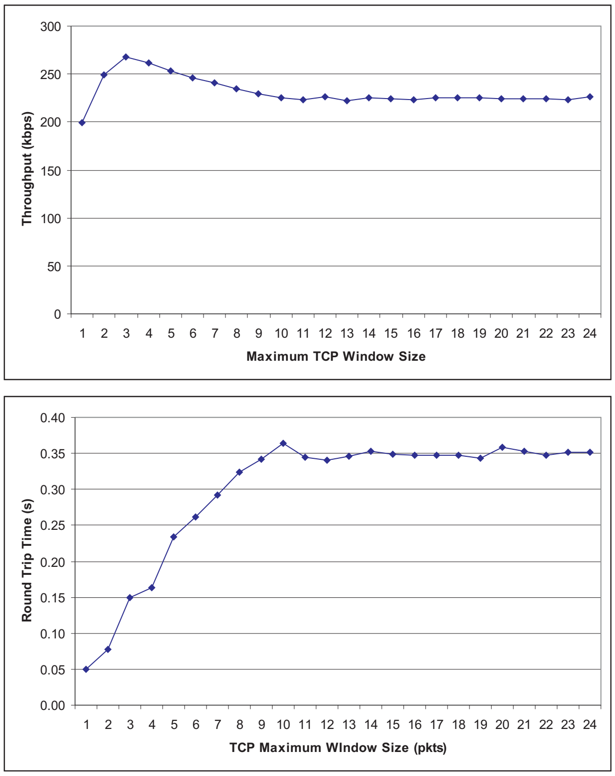

Fig. 2. Throughput and Round-Trip Time vs. Maximum TCP Window Size

B. *Explicit Congestion Notification*

Traditionally, TCP has relied on dropped packets at the router as an indication of network congestion. When a TCP sender receives three duplicate acknowledgments, it assumes a packet has been lost and it reduces its window size. However, when the congestion window of the TCP source is below four, it cannot receive three duplicate acknowledgments so a dropped packet triggers a retransmission timeout and a possible subsequent exponential backoff that causes significant throughput degradation.

[11] proposed Explicit Congestion Notification (ECN) as a TCP improvement in which packets are marked instead of dropped. Marking uses two bits in the IP header: one to indicate that then end-host is capable of detecting marks and the other by a congested IP router to signal congestion. If a router detects congestion, it marks packets that are ECN capable instead of dropping them. The TCP destination echos the mark (the ECN bit) back to the source which then takes the same set of actions it would take if the dropped packets had been detected. The key advantage of marking is that the TCP source receives the explicit congestion indicator much sooner than when packets are dropped.

The critical point for our work is that for ad hoc networks with a small diameter (about 20 hops or fewer), the window size of a TCP flow needs to be small for optimal performance, as show in Section II-A. With these small window sizes, an IP router that drops a packet from a TCP flow forces a timeout since the sender can not get three duplicate acknowledgments. With these same window sizes, an IP router that marks a packet from a TCP flow allows the TCP source to continue transmitting at a reduced rate since three duplicate acknowledgments are not required. We assume that all future TCP sources will be ECN enabled.

C. *Link RED and Adaptive Pacing*

Random Early Detection (RED) [12] is an Active Queue Management (AQM) scheme that uses the average queue length to determine the dropping or marking probability of packets in the queue. The probability is 0 if the average queue size is less than min_{th}, linearly increases from 0 to max_p from min_{th} to max_{th} and is 1 when the average queue size exceeds max_{th}.

LRED [8] is a data link layer strategy based on RED that keys on the average number of IEEE 802.11 retries instead of queue length. Analogous to RED, LRED does not drop any packets when the average link layer retries is less than min_{th} and linearly increases from 0 to 1 from min_{th} to max_{th}. LRED then drops packets with the minimum of the drop probability based on link layer retransmissions or the parameter max_p. LRED can achieve the optimal window size desired by TCP flows on wireless LANs for some configurations, but it shares RED's tuning weaknesses, noted in [13], [14], [15]. Moreover, the fact that LRED drops packets makes it difficult to configure when TCP windows are small, as described in Section II-B and if marking at the IP layer based on MAC layer data poses possible network layer violations.

Along with LRED, [8] presents Adaptive Pacing (AP) which is activated by LRED when the average number of retries is less than min_{th} and deactivated when the average number of retries exceeds min_{th}. AP increases MAC layer backoff intervals by the retransmission time of one data frame every time an ACK frame is received. Our analysis in [16] indicates that most of the throughput improvements from LRED coupled with AP are due to AP and not LRED. Unfortunately, the downside of AP is that the additional backoff time between transmissions increases round-trip times.

III. Proposed Mechanism

This section presents the Low Delay Marking (LDM) algorithm which is run at each node along a TCP flow on a multihop ad hoc wireless network as illustrated in Figure 1 Each node counts the number of flows traveling through it, as explained in Section III-C, and maintains per-flow state information on the number of hops per flow, as described in Section III-B. For each arriving packet, the node computes the optimal window size for the flow, as described in Section III-A, and marks the packet with the marking probability required to meet this window size, as described in Section III-D. Figure 3 summarizes the LDM algorithm. In the algorithm, f_i is the i-th flow; h_i is the number of wireless hops f_i makes in going from source to destination; p_{mark} is the marking probability calculated by the IP packet queue management; n is the total number of flows going through the node; w_{opt} is the optimal window size for f_i; and p is the packet that arrived at the node.

```
at each node, on receiving packet p
    identify flow f_i to which p belongs
    estimate h_i for f_i
    estimate n
    calculate w_opt
    calculate p_mark
    mark p with probability p_mark
```

Fig. 3. The LDM Algorithm

A. *Optimal Window Size of a TCP Flow*

[7] and [8] derive expressions for the optimal TCP window size as a function of the number of hops between the source and destination nodes in a multihop wireless network. Summarizing these results, a TCP flow achieves maximum throughput when its window size is about one-

fourth of the number of hops in a wireless network chain. This restricted window size limits the number of packets in the network, thereby reducing MAC layer congestion (RTS/CTS collisions). However, in determining this optimal TCP window size, neither [7] nor [8] take into account the number of flows. Intuitively, the *aggregate* window size among all flows should be one-fourth of the number of hops (h). Thus, each flow should have a window size of one-fourth of the number of hops divided by the number of flows (n):

$$w_{opt} = \frac{\frac{h}{4}}{n} \tag{1}$$

B. Number of Hops for a Flow

To estimate the number of hops from the source to a destination for a flow, each node keeps per-flow state information, where a flow is identified by an IP source-destination pair. For each active flow, a node records the average time-to-live (TTL) values in the data packets it routes. It also observes destination-source acknowledgment packets for the same flow and records their average TTL value. Since the default TTL values set by modern operating system are typically 128 or 256, each node can compute the number of hops from the node to the source and the number of hops from the node to the destination, thus determining the total number of hops for each flow from source to destination. For example, if a node observes a data packet with a TTL value of 250 and then a corresponding acknowledgment packet with a TTL value of 251, it can compute the number of hops for that flow (h_i) as $(256 - 250) + (256 - 251) = 11$.

C. Number of Flows at a Node

Based on Morris' calculations[17], the number of flows at a node can be counted using a fixed-length bit vector v. When a packet arrives, it is hashed based on source-destination address and port number and the corresponding bit in v is set. The count of bits in v is an approximation of the number of active flows. The bits in v are cleared at a rate so as to reset every bit in v every few seconds. When a bit is cleared, the corresponding per-flow state information kept (for example, number of hops for the flow) is also cleared. This method of tracking flows is very accurate when the number of bits in v is significantly larger than the number of flows and it does not require any explicit modification of TCP.

D. Marking Probability

TCP performance models under congestion marking come from work in [18] and [19], with more detailed performance models in [20] and [21]. Based on results from pilot studies (see [16] for full details), we use the relationship between marking rate (p) and window size (w) derived in [17]:

$$p = \frac{0.76}{w^2} \tag{2}$$

From algorithms described in the previous sections and the state information kept on each active TCP flow, an LDM node calculates the optimal window size for each TCP flow and, using Equation 2, the appropriate marking probability to achieve that window size as:

$$p_{mark} = \frac{0.76}{\left(\frac{h}{4 \times n}\right)^2} = \frac{12.16 \times n^2}{h^2} \tag{3}$$

However, a w_{opt} of 1 results in a marking marking probability of 0.76 which, even with packet marking, causes timeouts. Therefore, if w_{opt} is calculated to be 1 or less, an optimal window size of 2 is used for w_{opt} instead.

Equation 3 represents the overall marking probability that needs to be applied to each flow. We propose that each ad hoc node contributes to this total equally, although alternate policies where the first node in a route applies the full marking probability are also possible. Since a packet has to go through $h - 1$ nodes from source to destination, LDM distributes the probability evenly over $h - 1$ nodes. Let p_{node} be the per-node marking probability. We can relate p_{node} to p_{mark} by:

$$p_{mark} = 1 - (1 - p_{node})^{(h-1)}$$

$$p_{node} = (1 - p_{mark})^{\frac{1}{h-1}}$$

$$p_{node} = \left(1 - \frac{12.16 \times n^2}{h^2}\right)^{\frac{1}{h-1}} \tag{4}$$

Thus, the overall marking probability, p_{mark} is the same as the probability of the packet not being marked through all $h-1$ nodes with probability of p_{node}. Using Equation 4, each node calculates the per-node marking probability for all incoming packets.

For evaluation purposes, the mechanisms described in Section III-B and Section III-C have been hard-coded into the simulation code used to evaluate LDM, with implementation and evaluation of the per-flow record keeping being future work.

IV. EVALUATION

This section discusses our simulation setup and analyzes the experimental results. Experiments presented include default TCP performance, TCP performance with window restrictions, TCP performance with adaptive pacing, and TCP performance with the LDM algorithm.

A. Simulation Setup

To evaluate the effectiveness of LDM, the NS-2 simulator [22] was enhanced to include code for the LDM algorithm as described in Section III. Due to the unavailability of Adaptive Pacing code from [8], we also had to implement Adaptive Pacing in NS-2 so as to be able to compare it with LDM. The simulated wireless ad-hoc network topology used in our investigations is shown in Figure 1. In general, there are $h+1$ wireless nodes, N_0 through N_h, connected over an IEEE 802.11 chain topology. Default IEEE 802.11 layer settings are used with a wireless capacity of 2 Mbps and AODV routing. All flows use TCP-NewReno with maximum window size of 32, except in the window constrained case.

The experiments reported in this paper include: regular TCP, which represents current practice in ad hoc network performance; TCP with a manually constrained window size which represents the optimal performance by manually constraining each TCP flow with full network knowledge; Adaptive Pacing where all MAC frames are delayed by an additional amount, as described in Section II-C, and LDM, the marking mechanism presented in Section III. Each of these cases was simulated with 7 hop, 15 hop and 24 hop ad hoc chain topologies where all nodes are immobile. Each simulation was run five times, with the graphs depicting the averages and minimum and maximum values shown with error bars. While the graphs report performance in absolute terms for round-trip time, loss rate, and total number of RTS collisions, throughput has been normalized to that of regular TCP case to help clarify the performance differences.

B. Single Flow

The first experiment has a single TCP NewReno flow going through a multihop chain wireless network. Figure 4 presents the normalized throughput, the loss rate, the round-trip time and the number of RTS collisions.

Over the 7-hop chain, regular TCP flow achieves 193 Kbps throughput, TCP with a restrained window size of three averages 260 Kbps (+29.5%), Adaptive Pacing improves throughput to 234 Kbps (+17.1%), and LDM achieves a throughput of 232 Kbps (+11.2%). The number of RTS collisions stay nearly the same with restrained TCP, but Adaptive pacing causes a 48.6% increase and LDM reduces RTS collisions by 9.9%. Restrained TCP reduces the round-trip time from 323 ms to 148 ms (−54.1%), Adaptive Pacing increases the round-trip time to 489 ms (+51.4%) and LDM reduces the round-trip time to 162ms (−52.0%), close to the restrained TCP level. The restrained TCP has the lowest loss rate and the original TCP the highest loss rate. Both Adaptive Pacing and LDM have slightly higher loss rates compared to the restrained TCP but LDM offers a lower loss rate compared to Adaptive Pacing.

With the 15 hop chain topology, regular TCP achieves 184 Kbps, restrained TCP with window size of 5 achieves 209 Kbps (+15.2%), Adaptive Pacing improves throughput to 213 Kbps (+19.8%) and LDM achieves 185 Kbps (+4.5%). Restrained TCP reduces the number of RTS collisions by 28.7%, Adaptive Pacing has about the same number of RTS collisions while LDM reduces RTS collisions by 17.9%. Restrained TCP reduces the round-trip time from 491 ms to 265 ms (−46.1%), Adaptive Pacing increases the round-trip time to 692 ms (+41.0%), LDM reduces the round-trip time to 327 ms (−33.3%), close to the restrained TCP. Restrained TCP has the lowest loss rate and regular TCP the highest. Both Adaptive Pacing and LDM have slightly higher loss rates compared to restrained TCP but LDM offers a lower loss rate compared to Adaptive Pacing.

Over the 24 hop chain, regular TCP achieves 176 Kbps, restrained TCP with a window size of 7 achieves 201 Kbps (+13.6%), Adaptive Pacing improves throughput to 228 Kbps (+28.4%), and LDM achieves 173 Kbps (+1.8%). Restrained TCP reduces the number of RTS collisions by 25.8% over that of regular TCP, Adaptive Pacing has about the same number of RTS collisions, and LDM reduces RTS collisions by 15.1%. Restrained TCP reduces the round-trip time from 626 ms to 394 ms (−37.0%), Adaptive Pacing increases the round-trip time to 887 ms (+41.8%), and LDM reduces the round-trip time to 459 ms (−26.6%), much closer to that of restrained TCP. Restrained TCP has the lowest loss rate and original TCP the highest. Both Adaptive Pacing and LDM have slightly higher loss rates compared to restrained TCP but LDM offers a lower loss rate compared to Adaptive Pacing.

C. Multiple Flows

This experiment involves three TCP flows going through a multihop wireless network. Figure 5 depicts the total throughput normalized to that of regular TCP, the total loss rate, the total number of RTS collisions and the round-trip time of one of the flows.

Over the 7 hop chain, regular TCP achieves 179 Kbps,

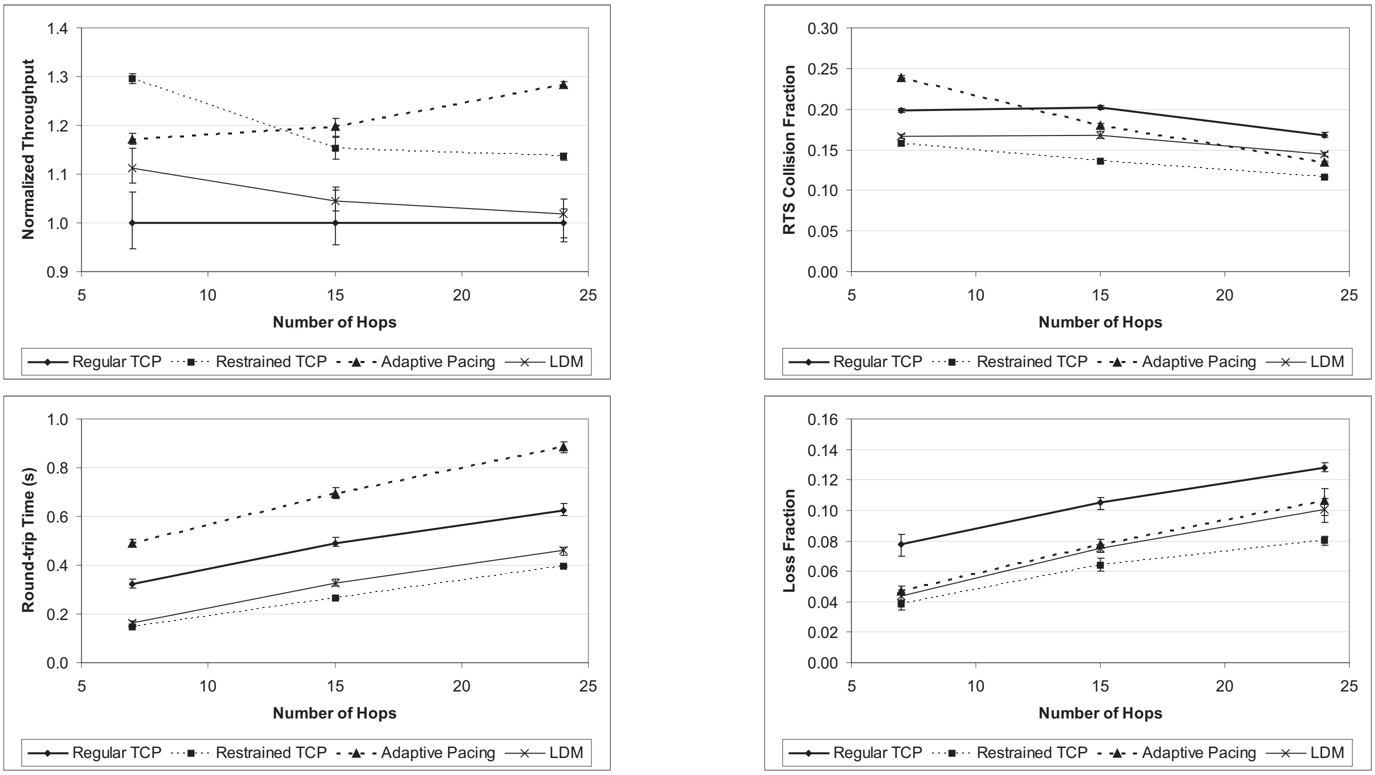

Fig. 4. Single Flow over Multihop Chain Topology

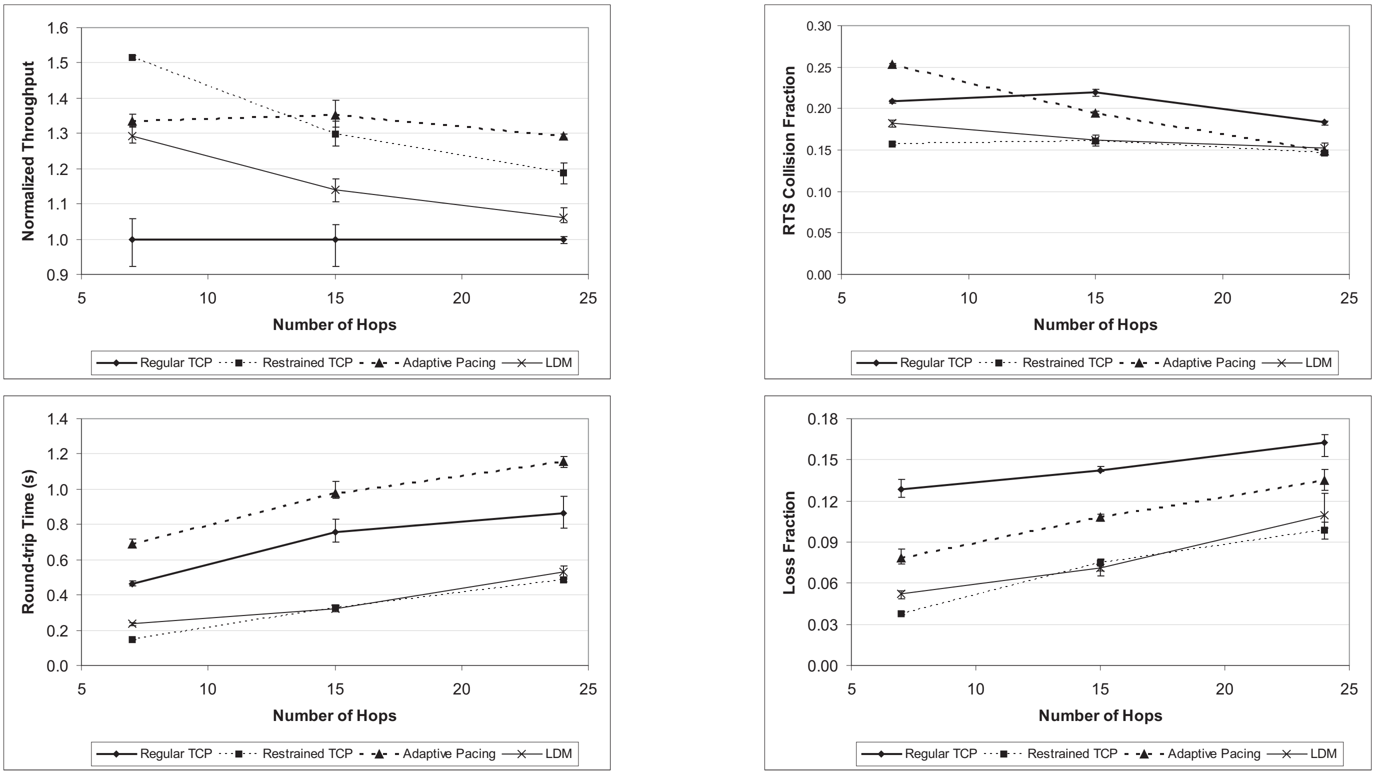

Fig. 5. 3 Flows over Multihop Chain Topology

restrained TCP with a window size of 1 achieve 262 Kbps (+51.6%), Adaptive Pacing improves throughput to 231 Kbps (+33.4%), and LDM achieves 220 Kbps (+29.22%). The number of RTS collisions increases by 7.2% for restrained TCP, Adaptive Pacing increases RTS collisions by 72.6%, and LDM stays around the same as restrained TCP

(9.8%). Restrained TCP reduces the round-trip time from 464 ms to 148 ms (-68.1%), Adaptive Pacing increases the round-trip time to 489 ms (+51.4%), and LDM reduces the round-trip time 237 ms (-49.0%), closer to the restrained TCP. Restrained TCP has the lowest loss rate and regular TCP the highest. Both Adaptive Pacing and LDM have slightly higher loss rates compared to restrained TCP but LDM offers a lower loss rate compared to Adaptive Pacing.

Over the 15 hop chain, regular TCP achieve 148 Kbps, restrained TCP with window size of 2 achieve 213 Kbps (+29.9%), Adaptive Pacing improves throughput to 215 Kbps (+35.2%), and LDM achieves 188 Kbps (+14.0%). The number of RTS collisions decreases by 11.5% for restrained TCP, Adaptive Pacing increases the number of RTS collisions by 15.7%, and LDM reduces RTS collisions by 21.8%. Restrained TCP reduces the round-trip time from 755 ms to 330 ms (-56.31%), Adaptive Pacing increases the round-trip time to 976 ms (+29.3%), and LDM reduces the round-trip time by to 320 ms (-57.6%), close to restrained TCP. LDM has the lowest loss rate and original TCP the highest. Both Adaptive Pacing and restrained TCP have slightly higher loss rates compared to LDM, but Adaptive Pacing offers a higher loss rate compared to restrained TCP.

Over the 24 hop chain, regular TCP achieves 176 Kbps, restrained TCP with a window size of 2 achieves 202 Kbps (+18.9%), Adaptive Pacing improves throughput to 227 Kbps (+29.1%), and LDM achieves 186 Kbps (+6.2%). The number of RTS collisions is reduced by 8.9% for restrained TCP, Adaptive Pacing has about the same number of RTS collisions, and LDM reduces RTS collisions by 15.3%. Restrained TCP reduces the round-trip time from 866 ms to 486 ms (-43.9%), Adaptive Pacing increases the round-trip time to 1155 ms (+33.4%), LDM reduces the round-trip time to 529 ms (-38.9%), much closer to the restrained TCP. Restrained TCP has the lowest loss rate and original TCP the highest. Adaptive Pacing has higher loss rates compared to the restrained TCP while LDM offers a slightly higher loss rate compared to the restrained TCP, yet a lower loss rate compared to Adaptive Pacing.

D. Summary

We summarize the performance of LDM compared with regular TCP from Section IV-B and Section IV-C into a table in Figure 6. A '+' denotes cases where LDM's performance is better by more than 10%, a '0' where LDM's performance is with within 10%, and a '-' where LDM is worse by more than 10%. From the table, LDM provides about the same or better throughput compared to regular TCP but provides a much lower round-trip time, loss rate and number of RTS collisions.

Category	Single Flow			Multiple Flows		
Hops	7	15	24	7	15	24
Throughput	+	0	0	+	+	0
RTS Collisions	0	+	+	0	+	+
Round-Trip Time	+	+	+	+	+	+
Loss Rate	+	+	+	+	+	+

Fig. 6. Performance of LDM compared to Regular TCP

Category	Single Flow			Multiple Flows		
Hops	7	15	24	7	15	24
Throughput	0	–	–	0	–	–
RTS Collisions	+	+	+	+	+	+
Round-Trip Time	+	+	+	+	+	+
Loss Rate	0	0	0	+	+	+

Fig. 7. Performance of LDM compared to Adaptive Pacing

We summarize the performance of LDM compared with Adaptive Pacing in the table in Figure 7. LDM provides about the same or less throughput compared to adaptive pacing, but provides greatly reduced round-trip times, loss rates and RTS collisions. These results are especially significant for applications that are sensitive to high delays.

V. Conclusion

The RTS/CTS mechanism in IEEE 802.11 was designed to mitigate the hidden terminal problem in wireless networks. RTS/CTS can reduce packet loss due to collisions in the MAC layer and works well for infrastructure wireless networks. However, in wireless ad hoc networks, the side effects of RTS/CTS mechanism include congestion and jamming in the MAC layer, which are hidden from higher layer protocols such as TCP. Consequently, transport layer protocols which do not account for MAC layer delays, such as TCP, will overestimate the available capacity and use too large a window size. Subsequently, this will further congest the MAC layer, leading to an increase in packet loss and round-trip time and a decrease in throughput.

This paper presents Low Delay Marking (LDM), an IP layer approach to enhance TCP performance towards lower delay and loss rate without sacrificing throughput. Building on knowledge of the optimal TCP window size discussed in [8], LDM marks packets with the probability calculated with the estimated number of hops and the number of flows. This forces the TCP flows to reduce their window size closer to an optimal value, thus resulting in a less congestion at the MAC layer. Less MAC layer congestion leads to fewer collisions and therefore decreases round-trip times and loss rates for all flows in the network.

We simulated and evaluated LDM over multiple chain topologies with a single and multiple-flows. The results show that LDM provides significantly better round-trip times (up to a 57.6% reduction) and loss rates (up to a 59.5% reduction) while still providing the same or better throughput compared to regular TCP. LDM also provides much better round-trip times (up to a 67.2% reduction) and loss rates (up to a 33.8% reduction) compared to Adaptive Pacing.

Currently, our evaluation is done over with the number of hops and number of flows known ahead of time by each router. Implementation of hop and flow counting techniques presented in Section III is our current ongoing investigation. Additionally, evaluations with more complex topologies such as crosses and grids is also under investigation.

Appendix

I. LRED Investigation

We implemented Link RED (LRED) and Adaptive Pacing from [8] in NS [22]. Unfortunately, we were not able to get the exact parameters used in the simulations from [8]. We tried various combinations of min_{th}, max_{th} and max_p without success at reproducing LRED results. In order to help finding the right parameters, we ran a single TCP flow simulation over a 7-hop chain topology. We graphed the cumulative distribution of the number of RTS retransmissions per frame in Figure 8. As the Figure shows, over 85% of the frames were successfully transmitted without any retransmissions. This results in a small range of min_{th} and max_{th} over which LRED can operate.

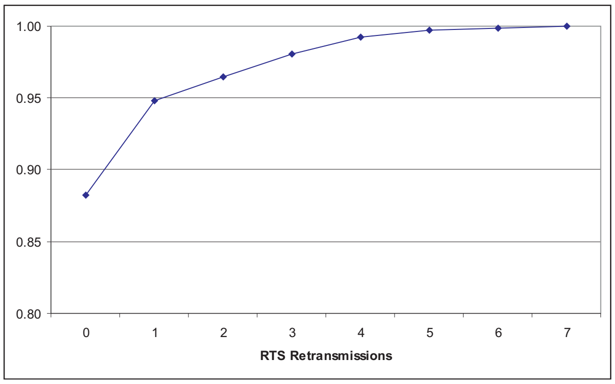

Fig. 8. CDF of RTS Retransmissions

Moreover, [8] tunes LRED based on a 7-hop chain wireless network topology. This requires a singe TCP flow to operate with a window size of 3 for optimal performance. However, packet loss for such a flow results in TCP timeouts which would decrease its throughput significantly. We suspect, but have not proven, that the benefits to TCP shown in [8] come from Adaptive Pacing, not LRED.

II. Cross Topology

In order to evaluate LDM over a more complex topology than a chain, we created a 6-hop cross topology as shown in Figure 9. This experiment involves two flows: one going the left to the right and the other going from the top to the bottom. Figure 10 depicts the total throughput normalized to that of regular TCP, the sum of roundtrip times, Jain's fariness [23] of throughputs and roundtrip times and the total number of RTS collisions.

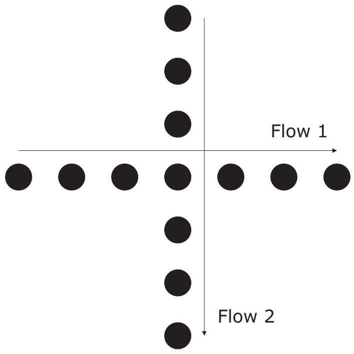

Fig. 9. Cross Simulation Topology

Regular TCP achieves 203 Kbps, restrained TCP with a window size of 2 achieves 261 Kbps (+28.7%), Adaptive Pacing improves throughput to 238 Kbps (+17.4%), and LDM achieves 198 Kbps (-2.3%). Restrained TCP reduces the number of RTS collisions by 15.2% over that of regular TCP, Adaptive Pacing increases the number of RTS collisions by 48.1%, and LDM reduces RTS collisions by 26.7%. Restrained TCP reduces the round-trip time from 604 ms to 121 ms (-79.9%), Adaptive Pacing increases the round-trip time to 912 ms (+51.1%), and LDM reduces the round-trip time to 315 ms (-68.5%), much closer to that of restrained TCP. Restrained TCP has the lowest loss rate and original TCP the highest. Regular TCP provides Jain's fairness index of throughputs of 0.873, restrained TCP produces 0.721 (-17.4%), Adaptive pacing 0.755 (-13.5%) and LDM 0.931 (+6.7%). Regular TCP provides Jain's fairness index of roundtrip times of 0.999, retrained TCP 0.900 (-9.9%), Adaptive Pacing 0.876 (-12.2%) and LDM 0.990 (-0.8%).

References

[1] IEEE Computer Society LAN MAN Standard Committee, "IEEE 802.11, 1999 Edition, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," .

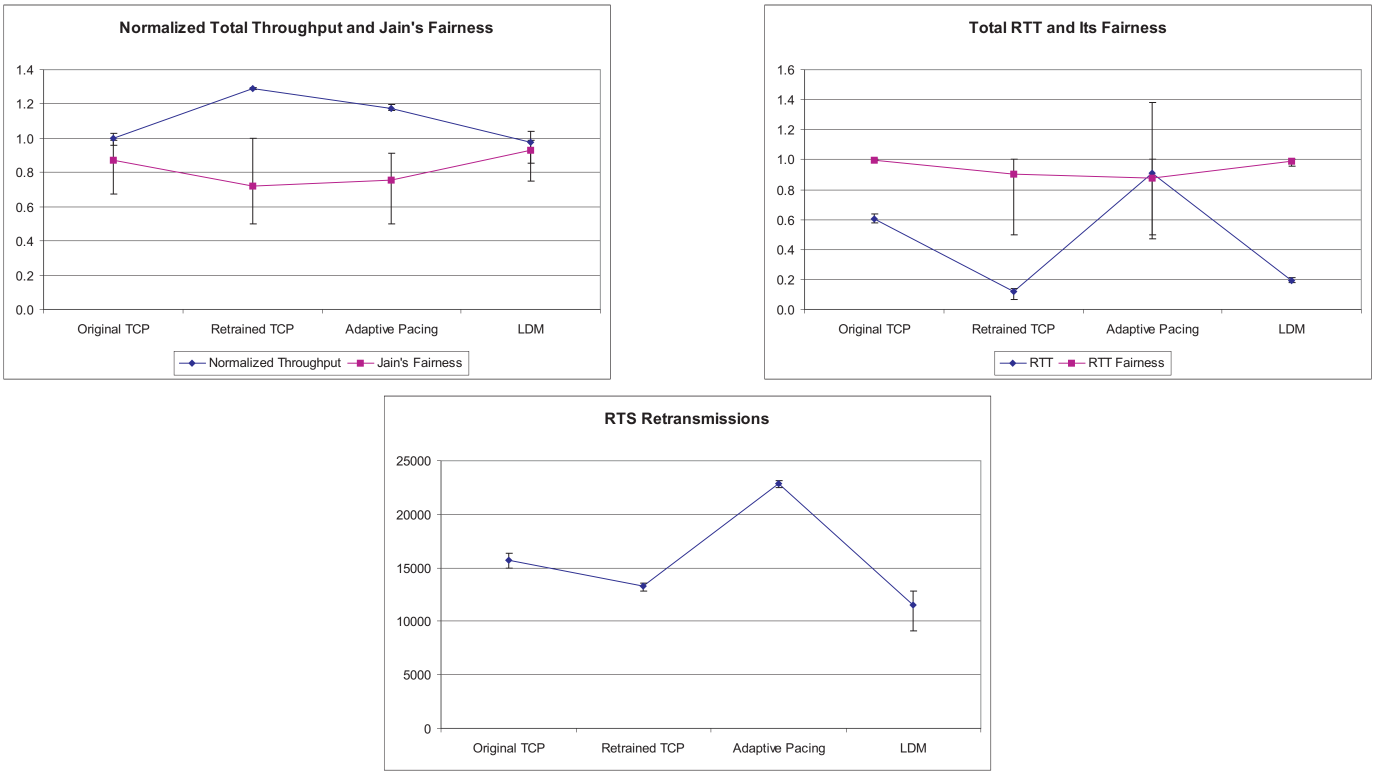

Fig. 10. Cross Simulation Results

[2] Kartik Chandran, Sudarshan Raghunathan, S. Venkatesan, and Ravi Prakash, "A feedback based scheme for improving TCP performance in ad-hoc wireless networks," in *Proceedings of ICDCS'98*, 1997, pp. 472–479.

[3] M. Gerla, K. Tang, and R. Bagrodia, "Tcp performance in wireless multihop networks," in *Proceedings of IEEE WMCSA'99*, Feb. 1999.

[4] Gavin Holland and Nitin H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *Proceedings of IEEE/ACM MOBICOM '99*, August 1999, pp. 219–230.

[5] Frederico Cali, Marco Conti, and Enrico Gregori, "Dynamic tuning of the ieee 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Trans. Netw.*, vol. 8, no. 6, pp. 785–799, 2000.

[6] Venkatesh Ramarathinam and Miguel A. Labrador, "Performance analysis of tcp over static ad hoc wireless networks," in *Proceedings of the ISCA 15th PDCS*, September 2002, pp. 410–415.

[7] K. Chen, Y. Xue, and K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks," in *ICC*, 2003.

[8] Zhenghua Fu, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang, and Mario Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss ," in *Proceedings of IEEE Infocom Conference*, Mar 2003.

[9] C.Ware, T.Wysocki, and J.F.Chicharo, "On the Hidden Terminal Jamming Problem in IEEE 802.11 Mobile Ad Hoc Networks," in *Proceedings of IEEE ICC'01*, 2001.

[10] Saikat Ray, Jeffrey Carruthers, and David Starobinski, "Rts/cts-induced congestion in ad-hoc wireless lans," in *Proceeding of IEEE WCNC 2003*, Mar. 2003, pp. 1516–1521.

[11] Sally Floyd, "TCP and Explicit Congestion Notification," *Computer Communication Review*, Oct. 1994.

[12] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, Aug. 1993.

[13] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, "Tuning RED for Web Traffic," in *Proceedings of ACM SIGCOMM Conference*, Aug. 2000.

[14] G. Iannaccone, M. May, and C. Diot, "Aggregate Traffic Performance with Active Queue Management and Drop from Tail," *ACM Computer Communication Review*, July 2001.

[15] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A Self-Configuring RED Gateway," in *Proceedings of IEEE Infocom*, Mar. 1999.

[16] Choong-Soo, Emmanuel Agu, Mark Claypool, and Robert Kinicki, "Low Delay Marking for TCP in Wireless Ad Hoc Networks," Tech. Rep. WPI-CS-TR-03-34, CS Department, Worcester Polytechnic Institute, Dec. 2003.

[17] Robert Morris, "Scalable TCP Congestion Control," in *Proceedings of IEEE Infocom*, Mar. 2000.

[18] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic," *ACM Computer Communication Reviews*, pp. 30 – 47, Oct. 1991.

[19] Matthew Mathis, Jeffrey Semke, Jamshid Madhavi, and Teunis Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM Computer Communication Review*, vol. 21, no. 5, Oct 1991.

[20] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation," in *Proceedings of ACM SIGCOMM*, 1998.

[21] Neal Cardwell, Stefan Savage, and Tom Anderson, "Modeling the Performance of SHort TCP Connections," Tech. Rep., University of Washington, 1989.

[22] Universiy of California Berkeley, "The Network Simulator - ns-2," Online at http://www.isi.edu/nsnam/ns/.

[23] R. Jain, *The Art of Computer Systems Performance Analysis:*

Techniques for Experimental Design, Measurement, Simulation, and Modeling, John Wiley and Sons, Inc., 1991.