TCP with Delayed Ack for Wireless Networks

Jiwei Chen, Yeng Zhong Lee, Mario Gerla, M.Y. Sanadidi
University of California, Los Angeles, CA 90095
Email:cjw@ee.ucla.edu, {yenglee,gerla,medy}@cs.ucla.edu

Abstract— **This paper studies the TCP performance with delayed ack in wireless networks (including ad hoc and WLANs) which use IEEE 802.11 MAC protocol as the underlying medium access control. Our analysis and simulations show that TCP throughput does not always benefit from an unrestricted delay policy. In fact, for a given topology and flow pattern, there exists an optimal delay window size at the receiver that produces best TCP throughput. If the window is set too small, the receiver generates too many acks and causes channel contention; on the other hand, if set the window too high, the bursty transmission at the sender triggered by large cumulative acks will induce interference and packet losses, thus degrading the throughout. In wireless networks, packet losses are also related to the length of TCP path; when traveling through a longer path, a packet is more likely to suffer interference. Therefore, path length is an important factor to consider when choosing appropriate delay window sizes. In this paper, we first propose an adaptive delayed ack mechanism which is suitable for ad hoc networks, then we propose a more general adaptive delayed ack scheme for ad hoc and hybrid networks. The simulated results show that our schemes can effectively improve TCP throughput by up to 30% in static networks, and provide more significant gain in mobile networks. The proposed schemes are simple and easy to deploy.**

I. Introduction

The Transport Control Protocol (TCP) is the most widely used reliable transport protocol over the Internet. TCP was originally designed for wired links where buffer overflows account for most packet losses. However, in multihop ad hoc wireless networks, several other inherent factors attribute to the TCP performance deterioration, including unpredictable channel errors, medium access contention complicated by hidden/exposed terminal problems, and frequent route breakages caused by node mobility. All these factors pose great challenges to the design of TCP protocols to provide efficient and reliable end-to-end communications. Many researchers have made valiant effort to propose various methods to make TCP survive in such challenging environments. In this paper, we focus on studying the effect of delayed acks on TCP performance. Then based on our analysis, we propose adaptive schemes that address the aforementioned factors effectively.

In standard TCP the receiver generates one ack for each data packet or two in-order data packets with the standard delayed ack option. This mechanism works well in wired networks. In multihop wireless networks, however, this mechanism can be further improved due to:

- Since the data and ack packets usually take the same route (or spatially close routes), they interfere with each other. The interference increases with the number of acks generated.
- Generating acks wastes scarce wireless resources. Though acks are essential to provide reliability, generating more acks than necessary is not desirable in wireless networks. Ideally, the receiver should generate minimal number of acks required for reliable data recovery.

Recently, the delayed ack strategy has been studied to improve TCP performance [1], [2]. However, this field is not fully exploited and many issues remain unsolved. Some important questions include how delayed acks effect TCP performance, and how to choose the optimal delay window (the number of in-order data packets to be waited for before generating an ack) in multihop wireless networks. In this paper we carry out a systematic study to understand the effect of delayed acks on TCP over wireless links. We investigate TCP performance and delayed ack related packet loss characteristics, under various wireless network scenarios and flow patterns.

Our objective is to clearly identify the relationship between TCP throughput and delayed ack over multihop wireless links. We reveal several interesting findings, which are tremendously beneficial to deeply understand TCP behavior in wireless multihop networks, and to design enhanced TCP protocols. First, we found that TCP does not always benefit from arbitrary delaying of acks. In fact, for a given network topology and flow pattern, there exists an optimal delay window size at the receiver, at which TCP achieves maximum throughput. Second, since 802.11 does not guarantee collision free packet transmission, a packet is more likely to be interfered with when going through a long path. If a large delay window is used over a long path, it causes large bursts of packets in transit, and this in turn causes severe packet interference with each other. When packet loss rate becomes high, the benefit of delaying acks, via reducing ack packets, disappears. Consequently, TCP performance degrades after reaching the peak at the optimal delay window size.

Armed with the deep understanding of delayed ack impact on TCP performance over multihop wireless links, we propose an adaptive scheme based on the hop count of a TCP path. The basic idea behind this scheme is, for a short path where interference problem is minimal, we delay the ack as much as possible to maximally improve TCP throughput; while for a long path, we apply an appropriate delay window size restriction to avoid high packet loss to achieve optimal TCP performance. Furthermore, we propose an end-to-end delay based scheme tailored for hybrid wired/wireless networks. These two schemes can be deployed separately, but are also compatible with each other to provide better performance in heterogenous networks. It is also worth noting that our

1-4244-0425-8/06/$20.00 ©2006 IEEE

proposed schemes only reside in transport layers at the end hosts, thus no modification on intermediate nodes is needed.

While our work shares the common concept with previous research in that the TCP receiver delays ack up to a certain number of data packets, we take a unique, systematic and optimized approach to understand the delayed ack impact and propose effective solutions. Moreover, to the best of our knowledge, our proposed schemes are the only existing mechanisms designed for both wireless and hybrid networks.

In ad hoc networks, delayed acks may potentially improve TCP throughput regardless of underlying routing protocols. Different routing mechanisms, however, have great impact on TCP performance [3], and to what degree delayed acks can benefit TCP performance. In this paper, we present the performance of TCP-DCA (Delayed Cumulative Ack) over popular ad hoc routing protocols, namely, Ad-Hoc On-Demand Distance Vector Routing (AODV) [4] and Greedy Perimeter Stateless Routing (GPSR) [5]. The reason we include GPSR in our study is that Geo-routing is a recently developed routing scheme promising to be scalable, and robust to node mobility. The significant TCP-DCA performance improvement is shown over the above mentioned ad hoc routing schemes, in both static and mobile ad hoc networks.

The remainder of this paper is organized as follows. Background work is provided in Section II. A thorough study of delayed ack impact on TCP performance is presented in Section III under various network topologies and traffic patterns. Section V evaluates TCP-DCA on static and mobile ad hoc networks, and hybrid wired/wireless networks as well. Section VI discusses a few issues to further improve TCP with delayed ack. We conclude the paper in Section VII.

II. Related Work

The standard TCP assumes that a packet loss is invariably due to buffer overflow and reduces the congestion window by half when packet loss happens. However, in ad hoc network, packet loss caused by buffer overflow is rare. Instead, it is more likely due to medium contention as shown in [6]. The fundamental problem resides in the limitations of IEEE 802.11 MAC. Since the interference range is usually longer than the transmission range, Request-To-Send (RTS) and Clear-To-Send (CTS) in 802.11 MAC cannot ensure collision free transfer of packets. This causes hidden/exposed terminals leading to packet loss in ad hoc multihop wireless paths [6]. Many research efforts have been made to adapt TCP to the unique characteristic of wireless ad hoc network, e.g. Transport layer "Fixed RTO" in [3], delayed ack in [1], [2], network layer support [7], [8], [9] and even lower-layer assistance, for example, MAC support in [6].

Although a TCP ack packet is small, typically 40 bytes, the transmission of TCP ack packets may require the same overhead as that of data packets in 802.11 MAC depending on the RTS threshold. If interference from TCP acks could be reduced, data packets would suffer less collisions resulting in higher throughput. Several approaches to delay acks have been proposed [1], [2], [10]. TCP-ADA (Adaptive Delayed Ack) [10] proposed to decrease the number of acks to improve TCP performance. They claimed that maximum TCP throughput is achieved when one ack acknowledges the full congestion window. However, the scheme did not address several important issues, such as packet loss and out-of-order packet reception. In fact, as we will show in this paper, TCP does not always benefit from delaying ack as much as possible.

Altman [1] presented a basic delayed ack scheme which was further improved in [2] where TCP-DAA (Dynamic Adaptive Acknowledgement) was proposed. In TCP-DAA, the receiver adjusts the delay window according to the channel condition (packet loss event). A TCP-DAA receiver delays acking until it receives a certain number of data packets, ranging from 2 to 4 packets. When there is no packet loss, the TCP-DAA receiver waits for more data packets (up to 4) before generating an ack, but reduces the number to 2 in case of out-of-order packet arrival. However, since a TCP sender automatically cuts the congestion window when packet loss occurs, i.e. automatically adapting to the channel state at the sender side, the receiver side adaptation provides little extra improvement. We will show that in our adaptive delayed ack scheme, the receiver does not respond dynamically to packet loss, yet achieves better performance.

In [2], the ack time is set to be one average packet inter-arrival time. That is, an ack is generated when no data packet arrives after one average packet inter-arrival time since last unacknowledged data packet. In wireless networks, the inter-arrival time is highly variable due to random MAC contention and back-off, so it is very difficult to get accurate enough statistics in a complex large system. Another impact of this timer implementation is that the receiver operates insensitively to the number of acks (2 to 4) to be delayed because any unexpected extra delayed data packet will trigger an ack.

An important aspect of TCP with delayed ack is the delay window size selection. In TCP-DAA [2], the receiver may delay up to four ack packets and this number is limited by the sender congestion window, which is fixed at four packets. The similar delay window size of 3-4 packets is also picked in [1] heuristically. There are issues in this scheme that desires further clarification and improvement. First, although a small congestion window limit helps TCP operation in wireless networks, it is not suitable for hybrid wired and wireless network where a high bandwidth delay product exists. Second, if the congestion window is not limited by four packets, the choice of a delay window size of 4 may no longer be suitable. In this paper, we address the above issues and provide more effective and general solutions that apply to various environments.

In this paper, we also study the impact of congestion window limit. A small sender congestion window can decrease interference and maximize pipeline effect. [6] revealed that there exists an optimal TCP congestion window size that maximizes spatial channel reuse. Further increasing the window size does not lead to better performance. On the contrary, it results in increased link layer contention and degraded TCP throughput. In [11], the optimal congestion

window limit is determined based on the hop count for maximum pipeline effect and the interference/transmission range ratio. In the paper, we will show that in our scheme TCP-DCA, the sender's congestion window is not limited, and yet, performs better than the case of the limited congestion window used in [2].

We also investigate the impact of delayed acks at a TCP-DCA sender. Since the receiver does not ack as frequently as in the standard TCP, the congestion window increase rate for delayed ack is slowed down. Such slower probing rate improves TCP performance in ad hoc networks, as reported in [12]. The conservative window increase, however, hinders efficient transmission in wired network where delay bandwidth product may be large. In this paper, we solve this problem and allow TCP-DCA to cope with ad hoc and hybrid wired/wireless networks via adaptive schemes.

III. TCP Throughput vs. Delay Window

In this section, we investigate the impact of the receiver ack delay window size on TCP performance. The conclusion we draw is that, when the path length is short, TCP achieves better performance with a delay window as large as the entire sender congestion window. When the path length increases, however, a large delay window triggers bursty transmissions, which results in mutual data packet contention and higher losses. In fact, for long paths, there exists an optimal delay window size that achieves maximum TCP throughput. We verify the delay ack impact using various network topologies and flow patterns.

In the following, we first study a basic scheme for TCP-DCA with fixed delay window limit and evaluate the impact of the delay window size on TCP performance. We use static routing to investigate delayed ack performance here ignoring any routing impact.

A. TCP with Receiver Delay Window Limit

A TCP-DCA receiver maintains a variable *delay window* "w" representing the number of data packet arrivals before acking at the receiver. This delay window w is bounded by a delay window limit, and also limited by the congestion window size to prevent the delay window from exceeding the congestion window which results in delaying the ack, but for possibly non-existent packets. If data packets arrive in order, the receiver generates one cumulative ack for every w data packets. If an out of order packet is received, or a packet that fills a gap in the sequence space of packets in the receiver buffer (that is, recovery from earlier packet loss) the receiver acks immediately to inform the sender of the packet loss/recovery in a timely manner. The receiver also keeps a fall-back delay ack timer which estimates the expected time to receive w data packets. At the TCP sender, when it receives a cumulative ack acknowledging w data packets, it updates the congestion window and sends out a burst of at least w packets, provided such packets are ready for transmission in the sender buffer.

To get the delay timer period, the receiver monitors the data packet inter-arrival interval and computes a smoothed interval through a low-pass filter. The average inter-arrival interval is used to set the ack delay timer based on the current delay window size. In TCP-DCA, an accurate delay timer is not needed and the timer is solely for fall back purpose. In fact, our inter-arrival time computation is rather loose: the receiver samples the inter-arrival time of *any* consecutively arrived data packets, not necessarily in-order packets. Therefore, the inter-arrival sample is an *inflated* value compared with inter-arrival between in-order packets. A TCP-DCA receiver smoothes such inflated inter-arrival samples through a low-pass filter:

$$I_{avg}^{i+1} = \beta I_{avg}^{i} + (1-\beta) I_s \tag{1}$$

where I_{avg} is the average of *inflated* packet inter-arrival time (I_s). I_{avg} is used to set delay ack timer (T_w) which defines the total time for receiving a whole delay window of in-order data packets:

$$T_w = \alpha w I_{avg} \tag{2}$$

where w is the delay window size, α is a parameter to tolerate high dynamic packet delay. Obviously, the delay ack timer is rather inflated and quite robust to high inter-arrival variation. In the paper, we choose β as 0.8 and α as 1.5. The receiver also generates an ack within 500 ms of the arrival of an unacknowledged data packet in accordance with RFC2581 [13].

One apparent drawback to this approach is that the TCP-DCA receiver needs to be informed of sender's congestion window size to prevent delay window larger than congestion window leading to unnecessary delaying at the receiver. In TCP-DCA the sender reuses the advertised window field in data packet header for "advertising" its congestion window size to the receiver. This design is not unrealistic due to the following fact that current TCP connection is mostly used for one direction, i.e. there is only data packets from the sender to the receiver and no backward data, and the advertised window field from the sender is usually wasted.

B. Chain Topology

We used the chain topology in Ns2 [14] simulation and this topology is general since TCP typically uses single path. An example is shown in Fig.1. The TCP connection is sourced at the first node (node 1) and packets travel over the chain to the end node (node 6). The interference range is 550m and transmission range is 250m. We place the nodes at 200m intervals, so nodes that are 4 hops away can transmit simultaneously without interference. Notice that a node that is 3 hops away is a "hidden node". IEEE 802.11 is the underlying MAC. Data rate on the wireless channel is 2Mbps and one simulation run lasts 300 seconds. Each data point represents an averaged of 5 simulation runs with different random seeds. The packet size is 1000 bytes and TCP-NewReno is used.

1) Single TCP Flow: TCP performance over wireless multihop inevitably depends on the path length (hop count). The longer the path is, the lower the throughput would be. We present TCP throughput as a function of delay window

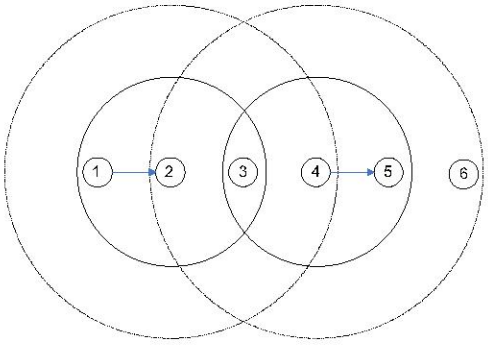
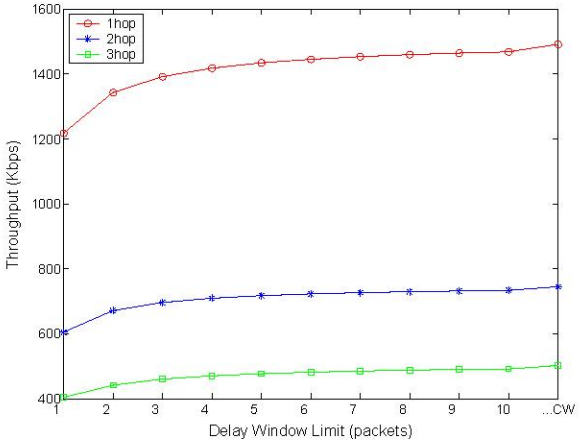


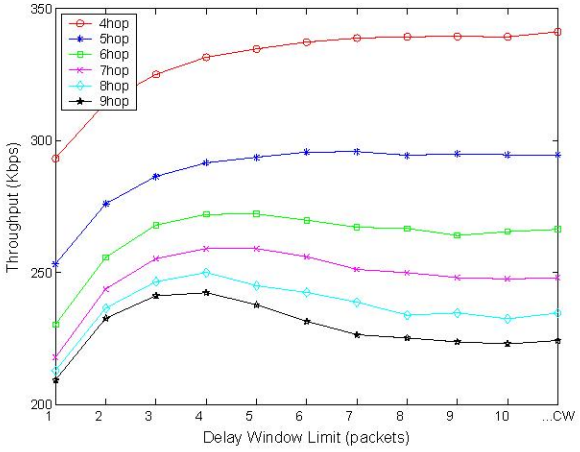
Fig. 1. Chain Topology. The solid-line circle denotes a node's valid transmission range. The dotted-line circle denotes a node's interference range. Node 4's transmission will interfere with node 1's transmissions to node 2.

limits on chain topologies with variable lengths in Fig.2. Fig.2(a) shows that when a sender and a receiver are within 3 hops, TCP-DCA gets steady performance gain by increasing the delay window size up to the entire congestion window size. For the one hop case, compared with the throughput of standard TCP (TCP-NewReno with delay window at 1), the fastest throughput increase can be seen when the delay window is small, say less than 5. The increasing trend becomes slower for delay windows larger than 5 and the throughput approaches the limit when the delay window reaches the congestion window size (CW). The same trend is also observed when the path length is 2 and 3. The reason for this performance gain is that 802.11 MAC provides collision free packet transmissions for short paths. Since the interference range is larger than 2 times the transmission range, when the sender and receiver are within 2 hops, every node along the path can sense all other nodes transmissions. In this case, no hidden nodes exist and thus no packet loss occurs. When the hop length is 3, the TCP receiver is the only node hidden from the TCP sender. If the receiver acks only after receiving all packets in a congestion window, no data packet can be interfered. Therefore, there is no data packet loss due to interference on a path not more than 3 hops. When the maximal performance gain is achieved when the receiver acks after receiving all packets in a congestion window, we observe that TCP-DCA attains up to 25% throughput gain relative to the standard TCP.

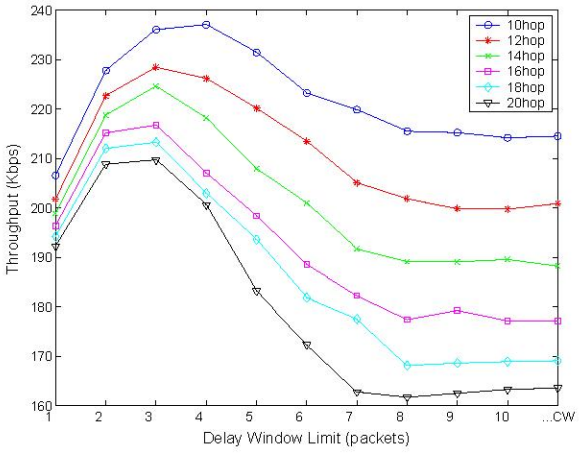
Since the interference range is larger than the transmission range, RTS/CTS cannot completely solve the hidden/exposed terminal problem. As the chain becomes longer than 3 hops, packet collision is unavoidable. For example, node 1 and node 4 are interfering nodes in Fig.1 and simultaneous packet transmissions on them will be interfered. The hidden terminal results in packet loss and this problem becomes more severe for longer paths because a packet has more chances to be interfered with. We observe this problem in TCP-DCA and this effect will be analyzed in detail later. Another problem is that when the sender receives a cumulative ack, it immediately sends a burst of packets. The larger the delay window, the larger the burst and the more packet loss due to increased interfering. Note that the packet burst size is directly related with the receiver delay window since since it is at least equal



(a) Path Length $h \leq 3$



(b) Path Length $10 > h > 3$



(c) Path Length $20 \geq h \geq 10$

Fig. 2. TCP Throughput vs. Delay Window on Chain Topology

to the size of the delay window.

When the packet loss becomes high, the TCP throughput gain from delaying ack is lost due to the increased packet losses. Fig.2(b) shows this tradeoff of TCP-DCA performance gain with delay window size for paths larger than 3 hops. When the hop count is 4 or 5, we do observe unsuccessful packet transmissions caused by interference, however, since a TCP sender is able to recover packet loss rather rapidly due to the small round trip time (RTT), TCP-DCA maintains performance gain by delaying ack for more data packets. For paths longer than 5 hops, TCP achieves throughput gain when the delay window size is small, but for large delay window size, delaying ack cannot maintain throughput gain because of excessive data packet losses. Further, now that RTT is larger, TCP spends more time detecting packet loss and recovering lost packets by entering fast retransmit/recovery in which only one packet is recovered per RTT, and thus more TCP throughput degradation. Therefore, for long paths, large delay window is not preferred.

We also show TCP-DCA performance over a very long path $h \geq 10$ in Fig.2(c). TCP only gets performance gain for small delay window size. For large delay window size, TCP even gets lower throughput than the standard TCP.

C. More Complex Topologies and Flow Patterns

We expand our study to scenarios of more complex topologies and flow patterns, including cross and grid topologies. For all cases, we observe the similar results to what is described above, i.e., when the path length is short, TCP achieves optimal throughput by delaying acks as much as possible, up to entire sender congestion window. On the other hand, when the path becomes longer, a large delay window hinders effective transmission and deteriorates TCP throughput gradually. We show that there exists a certain delay window size, at which TCP achieves optimal throughput performance. The following provides a short summary of results with more flows and various topologies. TCP-DCA performance over random topologies with multiple flows are also shown in Section V-B.

1) Multiple TCP Flows: Fig.3 exhibits result for 5 concurrent TCP flows on a chain topology with hop count varying from 3 to 7. It shows similar results to Fig.2.

2) Cross and Grid Topology: Fig.5 shows examples of cross and grid topologies with variable path lengths. Similar trends to the results in chain topologies are observed. We also ran extensive simulations on cross and grid topologies with multiple overlapped flows instead of one flow. The results, not included here due to space constraints, confirm the same trends discussed above.

IV. TCP with Adaptive Delayed Cumulative Ack (TCP-DCA)

In this section we further analyze the packet loss associated with bursty traffic triggered by delayed ack to investigate the reason for the above phenomena from the angle of MAC layer. Then we propose TCP-DCA with adaptive delayed window according to the underlying path information to optimize TCP performance.

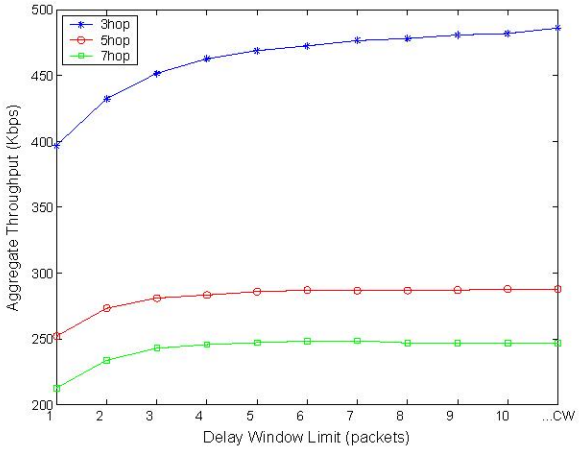
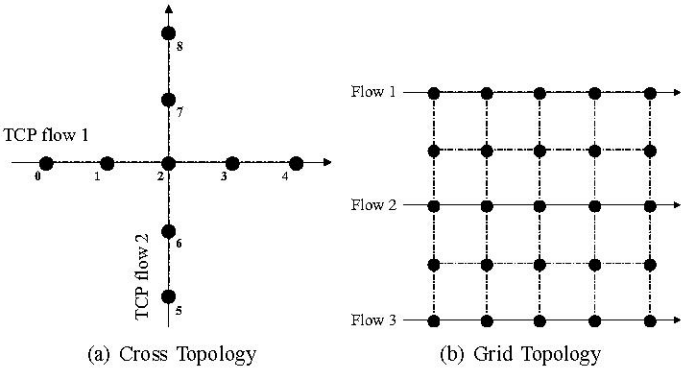


Fig. 3. TCP Throughput vs. Delay Window on Chain Topology (5 flows)



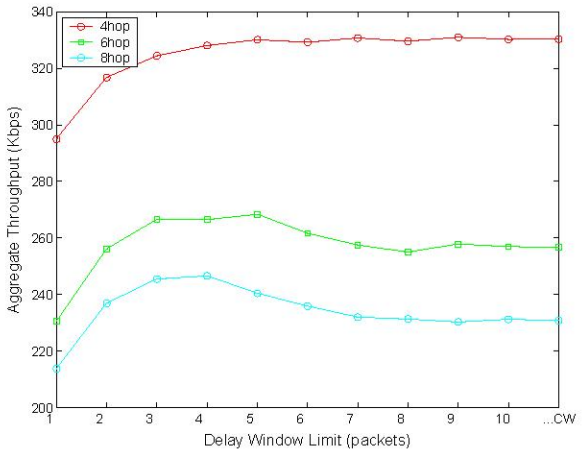
(a) Cross Topology (b) Grid Topology

Fig. 4. More Complex Topologies. Left: cross topology with 4 hop path on each direction. 200 meter distance between two adjacent nodes. Right: 5x5 grid topology, 200 meter distance between horizontal and vertical adjacent nodes.

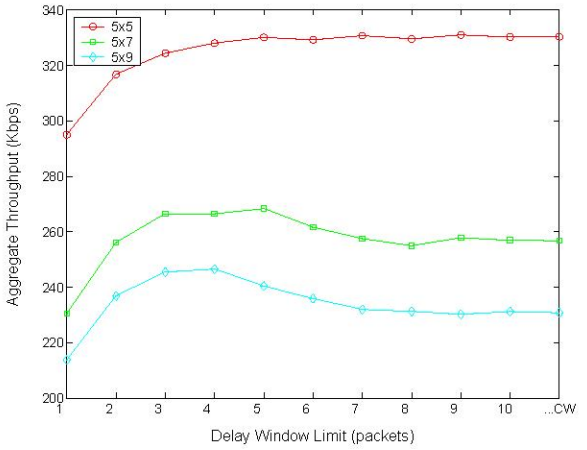
A. Packet Burst Transportation over 802.11 MAC

In what follows we study the effectiveness of burst transportation over 802.11 MAC. In TCP-DCA, the sender emits a burst of packets after receiving a cumulative ack, therefore, the efficiency of such bursts transport has direct impact on TCP performance. For short paths, since the 802.11 MAC can guarantee packet transmission without collision, no packet loss occurs no matter what the burst size is. Therefore, TCP throughput reaches the peak when delay window is at maximum size (congestion window size). However, when paths are longer, collisions occur.

Considering packet losses caused by interference among data packets of a burst, in Fig.1, if node 1 is transmitting to node 2 and node 4 is transmitting to node 5 simultaneously, the packet from node 4 is rarely interfered by a transmission from node 1, while the packet from node 1 has high probability of being interfered with. The reason is that a data packet is usually much larger than RTS/CTS/ACK in the MAC layer, the packet interference probability at node 2 is significant since it is interfered by data transmission on node 4. On the contrary, the interference at node 4 can happen only when node 4 is receiving CTS/ACK and node 2 is transmitting CTS/ACK, since RTS/CTS/ACK packets are



(a) Cross Topology



(b) Grid Topology

Fig. 5. TCP Throughput vs. Delay Window on Complex Topologies

small (at most 20 bytes, i.e. less than 2% of data size), the packet loss probability at node 4 is negligible. Even with MAC retransmissions of collided packets, successful packet transmission probability from node 1 to node 2 depends on the link-layer queue size at node 4. Because node 4 has little interference, the transmissions from node 4 usually capture the channel while node 1 keeps in back-off phase induced by heavy interference. From this example, we know that the packet loss can happen for a long path.

Therefore in a long path, the packets sent at the beginning of the burst will potentially interfere with the packets at the rear of the burst. And the packet losses increase with the increase of the burst size. Moreover, the longer the path, the more the interfering nodes, thus the more packet losses since the packet has more chances to be interfered. For a given long path, if the burst size is small, TCP can get performance gain. However, when burst size becomes large and the packet loss is so high that TCP cannot efficiently recover from it, TCP performance starts to deteriorate.

B. TCP-DCA with Adaptive Delayed Ack

Up to now we have presented the relationship between packet burst size and delay window size, and the tradeoff between TCP throughput and delay window size on different path lengths. Inspired by this observation, we dynamically determine the delay window size in TCP-DCA based on the hop count of a TCP connection. For a short path ($h \leq 3$), TCP-DCA could delay ack for a whole congestion window to get best performance. For longer paths, delay window should not be too large. We did extensive simulations and give proper delay window sizes applied in TCP-DCA according to the path length. These values are listed in Table I. For path length information, TCP-DCA receiver can get it from the TTL field in TCP packet header or from the routing layer at the receiver node.

TABLE I
DELAY WINDOW AT TCP-DCA RECEIVER

Path Length (h)	Delay Window Limit
$h \leq 3$	Congestion Window
$3 < h \leq 9$	5
$h \geq 10$	3

TABLE II
DESIGN DIFFERENCE

	TCP-DAA	TCP-DCA
Sender	Duplicate acks threshold to trigger retransmission is 2; CW upper limit is 4; Retransmission (RTO) timer is increased fivefold	Puts CW into advertised window field in packet header
Receiver	Delay window is adaptive from 2 and 4 based on loss event	Delay window is adaptive on the path length

V. PERFORMANCE EVALUATION

This section presents the TCP-DCA performance over wireless and hybrid networks. First, we show TCP-DCA performance on static multihop wireless networks and compare it with TCP-DAA [2]. Second, we demonstrate TCP-DCA performance on mobile ad hoc network, and show TCP-DCA performance over several ad hoc routings. Last, we extend the hop count based approach to an end-to-end delay based scheme to further improve TCP-DCA performance in hybrid wired/wireless networks.

A. Static Ad Hoc Network

In this section, we show TCP-DCA performance in static ad hoc networks. Meanwhile, we compare TCP-DCA with TCP-DAA and the standard TCP. TCP-DAA is an interesting extension of [1] and has shown good performance in static ad hoc networks [2].

We list the major differences between TCP-DAA and TCP-DCA in Table II. From these differences, TCP-DCA is clearly much simpler than TCP-DAA. For fair comparison, most simulations scenarios presented in this subsection were shown in [2]. The delay window in TCP-DCA is configured as in Table I. Each result is the average of 5 simulation runs.

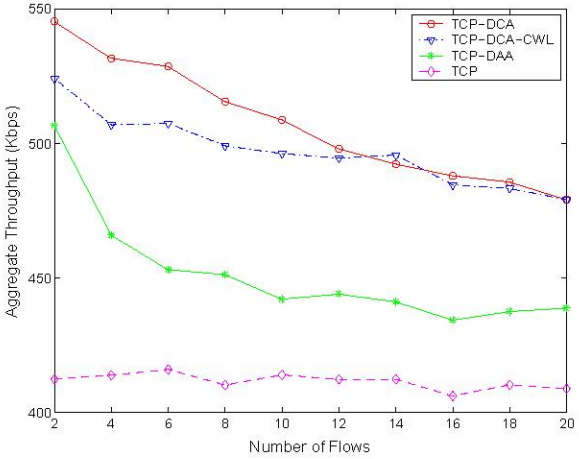
In Fig.6 we compare TCP-DAA to TCP-DCA in the chain topology with different hop count and number of concurrent flows. Although TCP-DCA is designed without congestion

window limit, we show TCP-DCA with congestion window limit at 4, named TCP-DCA-CWL, to study the impact of congestion window limit on TCP-DCA. Note that in TCP-DAA, the congestion window limit is set to 4. Fig.6(a) shows the performance of TCP-DCA over 3 hop chain. The aggregate throughput of TCP-DCA decreases when the number of flows increases. When there are few flows, since each TCP-DCA generates as few acks as possible, the advantage of TCP-DCA is obvious. When more flows coexist, the congestion window of each flow becomes smaller since they compete for the bandwidth, thus the benefit of delayed ack reduces because more acks are generated. If the congestion window is below the congestion window limit in TCP-DCA-CWL, they have similar performance. TCP-DCA and TCP-DCA-CWL provides better performance than TCP-DCA and the standard TCP. TCP-DCA achieves best performance, up to 15% improvement over TCP-DAA, and 30% over the standard TCP respectively.

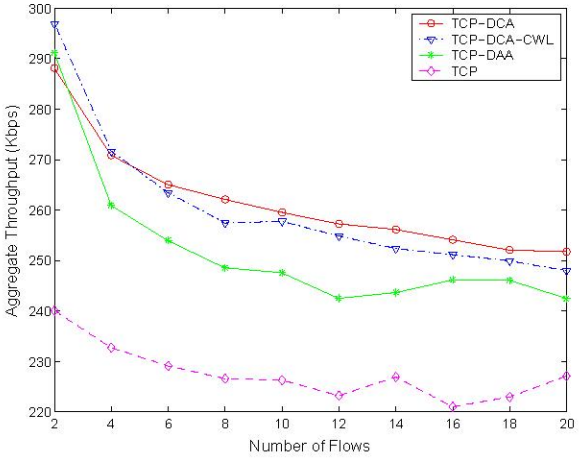
Fig.6(b)-Fig.6(c) show the performance of TCP-DCA on a 5 and 7 hop path. TCP-DCA outperforms all the other algorithms in most situations. Only TCP-DAA outperforms slightly our mechanism in the scenario with 7 hops and 2 concurrent flows. As the number of flows increases, the performance of both strategies degrades. However TCP-DAA degrades significantly, while it is not the case in our scheme. Based on the simulation results, our scheme is superior to TCP-DAA. Although not shown, we also find that our scheme generally recovers faster when sender's timeout occurs. In TCP-DCA, the sender's RTO is increased by fivefold, thus possibly has long idle time with severe performance hit.

Note that the congestion window limit on TCP-DCA does not bring considerable benefit. For a short path in Fig.6(a), the congestion window limit could considerably restrict the TCP performance as the number of flows is small. For a long path in Fig.6(b)-Fig.6(c), the congestion window limit only provides slightly more performance gain for 2 flows, and this advantage disappears for more flows. From Fig.6, the performance of TCP-DCA without congestion window limit is better than that of TCP-DCA-CWL in most situations. It indicates that TCP-DCA can achieve the distinguished performance without setting congestion window limit. This property makes TCP-DCA distinct since setting congestion window itself is non-trivial. It requires the knowledge of transmission/interference ratio [11] at the physical layer which is usually hard to obtain.

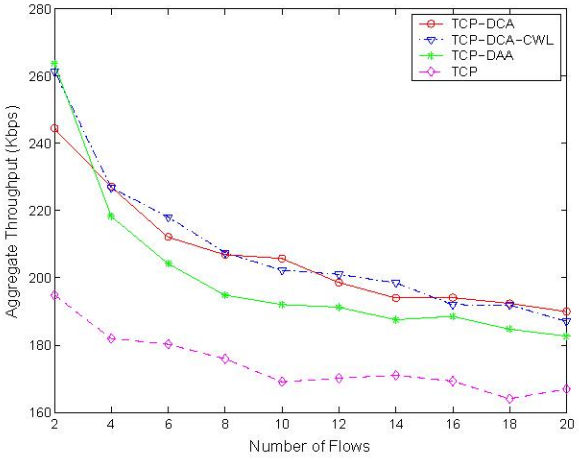
We also give results for the grid topology. This grid topology is shown in Fig.7(a) and 6 flows running on the grid. The performance of TCP-DCA and TCP-DAA is presented in Fig.7(b). We compare TCP-DAA, TCP-DCA and the standard TCP with/without congestion window limit. When the congestion window size is limited by 4, the performance of TCP-DCA is similar to that of TCP-DAA, and neither scheme provides significant improvement over the standard TCP. However, when the congestion window is not limited, TCP-DCA achieves better performance than TCP-DAA and the standard TCP. Note that TCP-DCA without the congestion window limit still gives comparable performance with TCP-



(a) 3 Hop Path

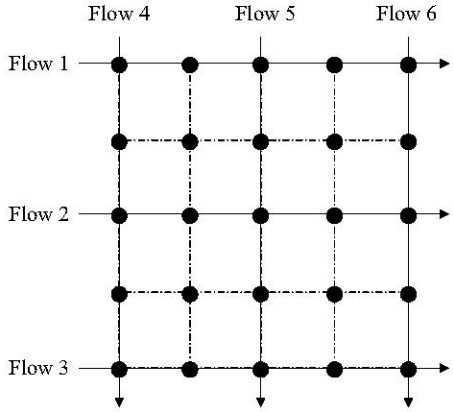


(b) 5 Hop Path

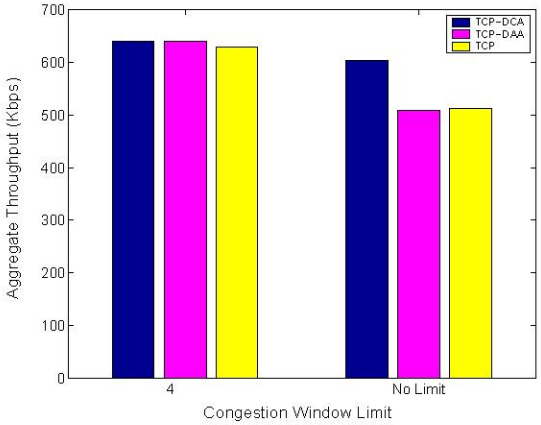


(c) 7 Hop Path

Fig. 6. Aggregate Throughput for Chain Topology

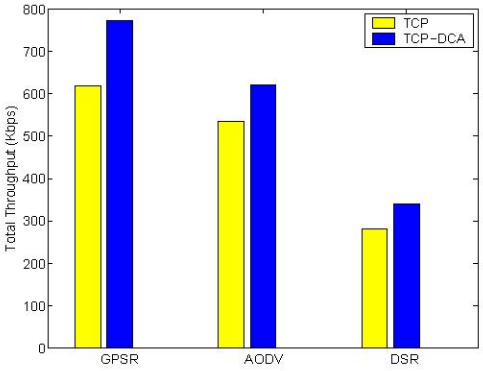


(a) Grid Topology

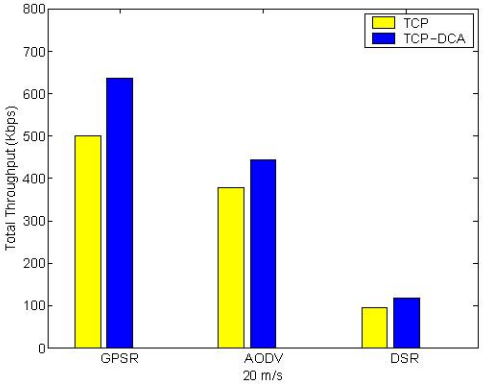


(b) Aggregate Throughput for Grid Topology

Fig. 7. Performance Comparison on Grid Topology



(a) 0 m/s



(b) 20 m/s

Fig. 8. TCP Performance over Mobile Network

DAA with the congestion window limit.

B. Mobile Ad Hoc Network

We have demonstrated that applying delayed ack scheme improves TCP performance in a static network, now we show that it can improve TCP performance in a mobile network as well.

In Fig.8 we show the performance of the standard TCP and TCP-DCA running over three routing schemes: GPSR, AODV and DSR. The experiment consists of 40 nodes randomly moving within a 1000m × 1000m area. Each node moves with the same speed without pause and the speed ranges from 0m/s to 20m/s to study low to high mobility. The Random Waypoint mobility model [15] is used. Note that setting the speed to 0m/s also represents a random static topology. All results displayed are calculated as the average of five runs with the same traffic pattern, but with different randomly generated mobility trace files. Traffic pattern includes single or multiple TCP flow(s).

We only illustrate results on speed 0 m/s and 20 m/s in Fig.8 with single TCP flow, for other speeds and multiple flows, the results are similar and we omit them due to the space limit. From Fig.8 we observe that the delayed cumulative ack contributes to the remarkable performance gain compared with the standard TCP. TCP-DCA produces at least 20% performance gain regardless of routing protocols and mobility speed. One interesting result is that TCP over GPSR performs generally better than TCP over AODV and DSR in mobile cases. GPSR has advantages in mobile network because nodes only keep Geo-locations for their neighbors, and supports end-to-end communication pattern without explicit route establishment. Therefore TCP performance on GPSR is less affected by mobility, for more details, please refer to [5].

C. Hybrid (Wired and Wireless) Network

In this section we study TCP-DCA in the wired and wireless ad hoc environment. Since wired network has abundant bandwidth, it demands a much larger congestion window than the pure ad hoc network for effective TCP performance. In current TCP implementation, TCP sender increases the congestion window only based on the number of acks received while not taking into account the acknowledged data covered in the acks. Using delayed ack could potentially cause slow congestion window increase and thus hurt TCP performance. Therefore, the pure adaptive delay window based on the hop count appears not adequate for this scenario. For example, in WLAN where 1 hop wireless client can delay the cumulative ack for the whole congestion window, the congestion window at the sender could increase slowly, particularly if a large propagation delay is encountered in the wired part. Although slow increase for congestion window is

helpful for improving TCP performance in ad hoc networks as shown in our previous results, it is not desirable for the hybrid network. In RFC3465 [16] Mark proposed to increase the congestion window based on the number of bytes acknowledged by the arriving acks rather than based on the number of acknowledgments that arrive at the sender in RFC2581 [13]. However, this approach potentially causes large bursts if the delay window is large. An appropriate delay window limit needs to be investigated in this scenario.

To combat the inefficiency of TCP-DCA in this scenario, we note that if minimum RTT is small, the congestion window can increase rapidly no matter what cumulative ack is delayed, otherwise the receiver could limit the delay window to a small value for faster acking. Recalling that a large end-to-end delay also prevents TCP from efficient packet loss detection/recovery, therefore this scheme also helps in TCP recovery from packet loss. Note that this end-to-end delay based scheme is compatible with the previous hop count based scheme. The reason is that since the propagation delay over wireless link is negligible, the minimum RTT on a multihop wireless path is basically the aggregate of transmission time for a data packet from source to the destination and an ack packet from destination back to source. Thus the minimum RTT has relation with hop count of the path, i.e. it increases linearly with respect to the hop count. For instance, with wireless bandwidth of 2Mbps, data packet size of 1000 bytes and ack size of 40 bytes, the minimum RTT for a 1 hop path is about 6ms considering various overhead, such as header overhead at TCP (20 bytes), IP (20 bytes) and MAC (34 bytes) and MAC layer fixed overhead shown in Table III (for details, please refer to [17]).

Since current wireless networks usually have link capacity no less than 2Mbps, the TCP receiver could anticipate the maximum value of the minimum RTT by assuming the wireless capacity as 2Mbps. It can do this because the TCP receiver has the information of data packet size and the path hop count (in hybrid network, hop count to the access point can be provided by the routing stack at the TCP receiver node). Thus, to be consistent with the hop-count based approach in multihop wireless networks (see Table I), the delay window chosen at the receiver is illustrated in Table IV by considering end-to-end delay. Note that this delay-based approach is more general since it can be applied in both pure ad hoc and hybrid networks. By estimating the minimum RTT at the receiver, the wireless receiver can select a proper delay window. For example, in hybrid networks where a large propagation delay resides in the wired link, the receiver still intelligently chooses a small delay window according to Table IV.

A challenge here is to estimate the minimum RTT at the receiver. If TCP includes two-way data, the receiver can get the accurate RTT measurement since the sender acks the data immediately. If TCP only has one-way data, RTT measurement at the receiver may not be straightforward. Since TCP timestamp is already used in many TCP implementations and it is a standard option, TCP receiver can use this option for RTT measurement, though such an estimate may be

TABLE III

IEEE 802.11B MAC OVERHEAD (WITH LONG PREAMBLE), L IS PACKET SIZE, R IS CAPACITY

Data Frame	$192\mu s + 8L/R$
SIFS	$10\mu s$
DIFS	$50\mu s$
RTS	$192\mu s + 160/R$
CTS	$192\mu s + 112/R$
ACK	$192\mu s + 112/R$

TABLE IV

DELAY WINDOW AT TCP-DCA RECEIVER (DELAY-BASED, WIRELESS BANDWIDTH IS AT LEAST 2MBPS AND TCP DATA PACKET SIZE IS 1000 BYTES)

Minimum RTT (ms)	Delay Window Limit
$RTT_{min} < 18$	Congestion Window
$18 \leq RTT_{min} \leq 60$	5
$RTT_{min} > 60$	3

inflated if the sender does not send data packets immediately after receiving an ack [18]. However, this would not cause much problem since only the minimum RTT is needed. In addition, the ack delay timer is only a coarse timer for predicting when to ack, and a possible inflated minimum RTT could be tolerated.

In Fig.9 we show a TCP connection from a wired network to a wireless network with up to 2 wireless hops. The one-way propagation delay on the wired link is 50ms. The performance of TCP-DCA, TCP-DAA and the standard TCP are shown in Fig.10. Since TCP-DAA limits the congestion window, the performance of TCP-DAA is much inferior to that of the standard TCP or TCP-DCA. TCP-DCA provides best performance, about 20% throughput gain over the standard TCP in both cases of 1 and 2 hop counts. Therefore, TCP-DCA is extensible to hybrid networks instead of working well only in an ad hoc network. It is an attractive point of TCP-DCA as it becomes common for wireless networks to communicate with the outer world. For example, in the real life, one needs to connect a mobile device (e.g. laptop) to the Internet to download files from a remote server, and may also need to upload files from mobiles to the Internet.

VI. FUTURE WORK AND DISCUSSION

In [19], ack congestion control is introduced to TCP to combat bandwidth asymmetry on forward/backward path. Ack packet should be reduced since backward path has limited bandwidth. However, the ack congestion control is implemented at the gateway with a RED queue. It is not an end-to-end approach since the receiver needs to get ECN

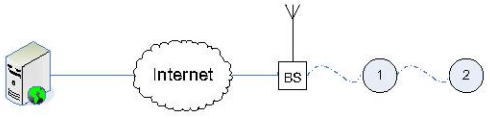


Fig. 9. Hybrid Wired/Wireless Topology

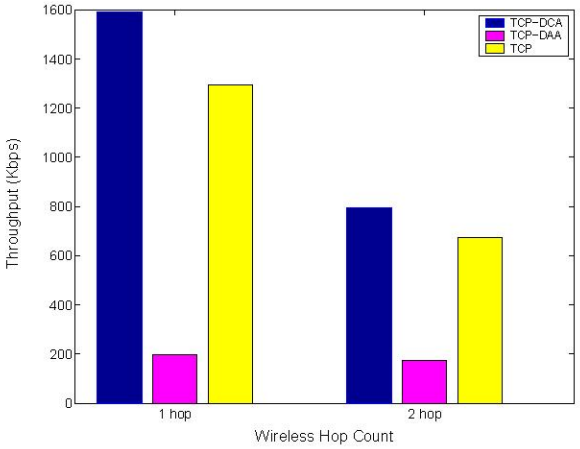


Fig. 10. Wired and Wireless Performance

signal from the gateway. In contrast, our scheme is end-to-end, and does not require any changes at intermediate nodes.

Delayed ack inevitably triggers burst transportation at the sender. The burstiness increases the packet loss and potentially hurts TCP performance. Since TCP-DCA does not use large delay window except for short paths, the burstiness is limited. It is possible to further reduce the burstiness. For example, in [20], congestion window increase is limited by at most 2 packets for each delayed ack, or ack reconstruction in [19] to rate control the sender. How to further decrease the burstiness would be an interesting future work.

Another future work includes an analytic model to study the impact of the proposed delayed ack scheme on TCP performance, taking into account of packet size, wireless medium contention and bandwidth and so on. The model will give more insight and justify the parameter selection for delay window size adaptation in TCP-DCA aside from simulation results alone.

In general, the advertised window field in TCP packet header is not used by the sender since TCP connections are predominantly half duplex. We propose in TCP-DCA to let the sender reuse the advertised window field for "advertising back" its congestion window size to the receiver. An alternative solution is to imbed the congestion window size in an option field of the packet header.

In this paper we have focused on how to reduce MAC layer interference between data and ack packets, via optimal delayed ack policy. We do not, however, address packet losses due to lossy physical channel. In the future, we plan to further investigate the lossy channel issue. We believe that TCP performance can be further improved by combining TCP-DCA with schemes that effectively combat packet losses due to error-prone physical channel, such as TCP-Westwood [21] and ELFN (Explicit Link Failure Notification) [7].

Furthermore, TCP-DCA is mainly a receiver-side modification, it can be combined with sender side modifications to achieve better performance, such as the interaction between sender's RTO calculation and receiver's delayed ack. It is also achievable to be integrated with other mechanisms [6], [7], [9] to improve TCP performance in wireless networks.

VII. Conclusion

TCP, a dominant reliable transport protocol in wired networks, is a highly competitive candidate for providing reliable data transport in wireless and wired/wireless networks. This paper systematically examines the relationship of TCP performance to delayed ack via extensive simulated results. We found that TCP does not always get throughput gain by delaying unlimited acks. The maximal TCP throughput is achieved at a certain delay window that balances decreasing ack flow and burst loss. Thus, we introduced two novel adaptive delayed ack mechanisms compatible with TCP called TCP-DCA, based on the path hop length and/or end-to-end delay, to maximize the TCP performance for wireless and hybrid networks. Our proposed schemes are shown to have superior performance, achieving up to 30% gain over the standard TCP in static networks (wireless or hybrid wired/wireless networks). We also demonstrated better TCP performance achieved by TCP-DCA over different routing protocols in mobile ad hoc networks.

References

[1] E. Altman and T. Jimenez, "Novel delayed ack techniques for improving tcp performance in multihop wireless networks," *Personal Wireless Communications*, September 2003.
[2] R. de Oliveira and T. Braun, "A dynamic adaptive acknowledgment strategy for tcp over multihop wireless networks," *Infocom*, 2005.
[3] T. D. Dyer and R. V. Boppana, "A comparison of tcp performance over three routing protocols for mobile ad hoc networks," *Mobihoc*, 2001.
[4] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," *Proceedings of IEEE WMCSA'99*, Feb. 1999.
[5] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," *Proceedings of ACM MobiCom'00*, Aug. 2000.
[6] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," *IEEE INFOCOM'03*, Mar. 2003.
[7] G. Holland and N. H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Proceedings of ACM MobiCom'99*, Aug. 1999.
[8] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing tcp fairness in ad hoc wireless networks using neighborhood red," *Mobicom*, 2003.
[9] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless neworks," *IEEE Personal Communications Magazine*, vol. 8, no. 1, 2001.
[10] A. K. Singh and K. Kankipati, "Tcp-ada : Tcp with adaptive delayed acknowledgement for mobile ad hoc networks," *WCNC*, 2004.
[11] K. Chen, Y. Xue, and K. Nahrstedt, "On setting tcp's congestion window limit in mobile ad hoc networks," *ICC*, 2003.
[12] K. Nahm, A. Helmy, and C. J. Kuo, "Tcp over multihop 802.11 networks: Issues and performance enhancement," *Mobihoc*, 2005.
[13] M. Altman, "Tcp congestion control," *RFC 2581*, 1999.
[14] "The network simulator - ns2," *Available at http://www.isi.edu/nsnam/ns/*.
[15] C. Bettstetter, H. Hartenstein, , and X. Prez-Costa, "Stochastic properties of the random waypoint mobility model," *ACM/Kluwer Wireless Networks: Special Issue on Modeling and Analysis of Mobile Networks*, September 2004.
[16] M. Allman, "Tcp congestion control with appropriate byte counting (abc)," *RFC 3465*, 2003.
[17] M. Kappes, "An experimental performance analysis of mac multicast in 802.11b networks for voip traffic," *SPECTS*, 2004.
[18] V. Jacobson, "Tcp extensions for high performance," *RFC 1323*, 1992.
[19] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The effects of asymmetry on tcp performance," *Mobicom*, 1997.
[20] M. Allman, "On the generation and use of tcp acknowledgments," *ACM Computer Communication Review*, 1998.
[21] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," *Proceedings of ACM MobiCom'01*, July 2001.