A Dynamic Adaptive Acknowledgment Strategy for TCP over Multihop Wireless Networks

Ruy de Oliveira

Torsten Braun

Technical Report
IAM-04-005
July 2004

A Dynamic Adaptive Acknowledgment Strategy for TCP over Multihop Wireless Networks*

Ruy de Oliveira and Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Bern
Neubrückstrasse 10, CH-3012, Bern, Switzerland
oliveira,braun@iam.unibe.ch

Abstract

Multihop wireless networks based on the IEEE 802.11 MAC protocol are promising for ad hoc networks in small scale today. The 802.11 protocol minimizes the well-known hidden node problem but does not eliminate it completely. Consequently, the end-to-end bandwidth utilization may be quite poor if the involved protocols do not interact smoothly. In particular, the TCP protocol does not manage to obtain efficient bandwidth utilization because its congestion control mechanism is not tailored to such a complex environment. The main problems with TCP in such networks are the excessive amount of both spurious retransmissions and contentions between data and acknowledgement (ACK) packets for the transmission medium. In this paper, we propose a dynamic adaptive strategy for minimizing the number of ACK packets in transit and mitigating spurious retransmissions. Using this strategy, the receiver adjusts itself to the wireless channel condition by delaying more packets when the channel is in good condition and less otherwise. Our technique not only improves bandwidth utilization but also reduces power consumption by retransmitting much less than a regular TCP does. Extensive simulation evaluations show that our scheme provides very good enhancements in a variety of scenarios.

CR Categories and Subject Descriptors: C2.1 **[Computer-Communication Networks]** Network Architecture and Design; C2.2 **Computer-Communication Networks]**: Network Protocols; C2.1 **[Computer-communication Networks]**: Internetworking.

General Terms: Algorithms, Design, Performance.

Additioanl Keywords: Wireless Networks, Transmission Control Protol, Congestion Control, Packet Loss, Energy Efficiency, Shared Medium.

* The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

I. Introduction

The Transport Control Protocol (TCP) is the most widely deployed transport protocol in the Internet today. TCP has been successful due to its robustness in dealing with dynamic changes in the network traffic conditions and providing reliability on an end-to-end basis. This wide acceptance has driven the development of a great deal of applications for TCP, which motivates the extension of this protocol for wireless networks. These networks pose some critical challenges to TCP since it was not originally designed to work in such complex environments, where the level of bit error rate (BER) due to the physical medium is not negligible. High levels of mobility may further degrade the end-to-end performance because TCP reduces its sending rate whenever it perceives a dropped packet.

We do not address mobility related issues in this paper, since it has been investigated in many past work [1]–[5], and also because we target scenarios in which the level of mobility is relatively low (pedestrian movement). The disturbance of such movements should not be too much disruptive to the transport protocol. Nonetheless, our technique is expected to work satisfactorily under a moderate level of mobility. In fact, this paper investigates TCP over multihop wireless media, which has not yet been thoroughly investigated in the research community.

IEEE 802.11 [6] protocol is the standard Medium Access Control (MAC) protocol for ad hoc networks, and it consists of both link and physical layer specifications. As we explained in more detail in [7], The 802.11 implements a robust link layer retransmissions strategy along with the RTS/CTS (request-to-send/clear-to-send) control frames for recovering locally (link level) most of the potentially lost frames. There is also a virtual carrier sense mechanism that used by every node to announce to all other nodes within a given area when the medium is busy, which aims at preventing the well-known hidden node problem. This protocol works efficiently for topologies of at most 3 hops between sender and receiver, which is commonly referred to as a 3-hop scenario.

For larger scenarios in terms of number of hops, the hidden node problem still persists due to spatial reuse property inherent in the propagation model of such wireless networks. Basically, the spatial reuse imposes that within a certain geographical area, only one node can transmit at a time, as studied in [5], [8], [9]. This characteristic of such networks causes adverse impact on the traditional TCP protocol since it is always probing the network for bandwidth by increasing its transmission rate until a lost packet is detected. Hence, unless an efficient coordination between MAC and transport protocols is in place, the end-to-end performance can be severely impaired.

There has been a belief in the research community that TCP can improve its performance by simply ignoring medium induced losses and not slowing down when reacting to those. However, recent research developments on this subject [5], [8], [9] have indicated that this procedure may not be really effective. Rather, this aggressive behavior can adversely degrade the protocol performance by inducing more losses. Actually, the main problem of TCP over 802 MAC protocol is the excessive number of medium accesses carried out by the protocol. This is caused not only by the ACK packets that compete with the DATA packets for the medium, but also by the retransmissions performed by TCP when reacting to losses.

This paper presents a dynamic adaptive strategy for decreasing the number of medium contentions as much as possible. Specifically, our technique generalizes the concept of delayed acknowledgement recommended in RFC 1122 in which the receiver should only send ACK packets for every other DATA packet received. In our proposal, the receiver may delay up to four packets when the wireless channel is in good condition, and less when the channel is facing losses. By doing so, our protocol outperforms a regular TCP whenever the channel is able to provide higher bandwidth and should perform as effectively as a regular TCP does when the channel is heavily constrained.

These simple changes reduce considerably the number of transmissions and retransmissions over the wireless medium. As a result, the overall power consumption is greatly reduced, which is a key issue for the battery-powered devices in place. Besides, the proposed protocol keeps the TCP-friendly behavior, which is neglected by many existing proposals. We focus our discussions on short chain of nodes of at most 7 hops, because this is a reasonable limit for today's networks. Nevertheless, the concepts here presented are general and expected to be effective in larger scenarios as well.

The remainder of this paper is organized as follows. The next section describes the main related work on TCP over wireless networks. Section III provides an overview on spatial reuse impacts on TCP. In section IV, we introduce our proposal, where the design decisions are explained and the main features are discussed in detail. Section V presents and discusses the simulation results. Section VI concludes the paper pointing out the main open issues and potential future work.

II. Related work

TCP performance in wireless environments has been investigated in several studies such as [10]–[14]. Most of them are meant for cellular networks where only the last hop communicates through wireless link. In this section, we focus on multihop networks in which long chains of wireless links may exist. For the sake of clarity, we classify the main related work into three groups as follows. In the first group, we describe the techniques that deal with packet loss discrimination in the wireless environment to prevent the TCP sender from slowing down in the face of wireless losses. In the second group, we address the schemes that either propose changes to the MAC layer or simply propose to adjust TCP parameters pursuing better TCP and MAC interaction. In the third group, we join the proposals to improve bandwidth utilization by decreasing the number of medium access requests since that is costly in a shared medium.

In the first group, Chandran et al. [1] proposed TCP-feedback, Holland and Vaidya [2] proposed a similar approach based on ELFN (Explicit Link Failure Notification) messages, and Liu and Singh [3] proposed ATCP protocol. These three proposals are quite similar regarding the way they detect and react to failures within the network. All of them attempt to distinguish the actual cause of packet losses inside the network and not to slow down if the loss is believed to be due to noise rather than congestion. A specific protocol is used to notify the sender when link interruptions, due to mobility, occur. Fu et al. [4] investigated TCP improvements by using multiple end-to-end metrics rather than a singe metric. They claim that a single metric may not provide accurate results in all conditions. They used four metrics: 1) Inter-packet delay difference (IDD) at the receiver, 2) Short-term Throughput (STT); 3) Packet out-of-order delivery ratio (POR); and 4) Packet Loss Ratio (PLR). These four metrics are cross checked for accurate detection of the network internal state. The evaluations focused on mobile scenarios, while channel errors were not appropriately investigated. Biaz and Vaidya [15] studied three schemes for predicting the cause of packet loss inside wireless networks, the key idea was based on RTT simple statistics on observed RTT and/or observed throughput of a TCP connection for deciding whether to increase or decrease TCP congestion window. The general results were discouraging in that none of the evaluated schemes performed really well. Liu et al. [16] proposed an end-to-end technique for distinguishing between packet loss due to congestion from packet loss by wireless channel. They designed a Hidden Markov Model (HMM) for making the mentioned discrimination based on Round-trip Time (RTT) measurements over the end-to-end channel. We have proposed a similar idea in [17] to discriminate packet losses due to congestion from packet losses by wireless in short chains of ad hoc networks. The discrimination is performed by a fuzzy logic engine to make the inference robust against the uncertainties inherent in RTT measurements.

In the second group, Li et al. [8] conducted an extensive analysis on spatial reuse properties of 802.11 MAC protocol in multihop networks. Their main conclusion is that the ideal capacity of a long chain of nodes is 1/4 of the raw channel bandwidth obtained from the radio. If greedy senders are in place, then the capacity degrades to up to 1/7 because the nodes closer to the source starve the last nodes. Fu et al. [9] investigated the best value for the maximum sender transmission window, and their results confirm those found in [8] in that TCP's best throughput is achieved when its window size limit is $h/4$ over a h-hop chain (chain with h hops). They also proposed a Link RED (LRED) algorithm for improving TCP performance. The LRED monitors the number of retransmissions at the 802.11 MAC protocol to compute the link drop probability which is used as the RED threshold rather than the conventional congestion threshold. The rationale for this mechanism is based on the observation that link losses are much more significant than congestion in such networks. More recently, Chen et al. [5] proposed an adaptive Congestion Window Limit (CWL) which is given by 1/5 of the Rount-Trip Hop-Count (RTHC) for the 802.11. They obtain that by considering the bandwidth-delay product of the path in multihop networks.

In the third group, Yuki et al. [18] proposed a technique for combining TCP DATA and ACK packets into a single packet for mitigating the interference problems found in multihop networks. Their evaluations are performed over ideal channels without typical losses. Altman and Jimenez [19] investigated the impact of delaying more two ACKs packets on TCP performance in multihop wireless networks. They concluded that in a chain topology of nodes, substantial improvement may be achieved by delaying 3-4 ACK packets. In their approach the sender always delays 4 packets (except at the startup) or less if the sender timeout expires. The sender uses a fixed interval of 100ms and does not react to packets that are out-of-order or filling in a gap in the receiver buffer, as opposed to the recommendation of RFC 1122. Their results are positive but obtained for a single flow only. Kherani and Shorey [20] proposed a simple analytical model for TCP over 802.11 based wireless ad hoc networks. In their model, the congestion window limit size defines the number of ACKs to be delayed. They assume no timeout for the delayed ACKs, which is opposite to the standard delayed ACK algorithm (RFC 1122) in which the delay cannot be higher than a given interval (typically 100ms). Although their model is very simplified the results are claimed to be quite accurate in predicting TCP throughput. Finally, Allman [21] conducted an extensive evaluation on Delayed Acknowledgment (DA) strategies, and they presented a variety of mechanisms to improve TCP performance in presence of side-effect of delayed ACKs. Their results are interesting but focused on wired networks only.

III. Background

A. Spatial Reuse in multihop wireless networks

In this section, we introduce the spatial reuse property of IEEE 802.11, which has been investigated in detail in [8] and revisited in [5], [9] under TCP perspective. The propagation model of 802.11 defines an interference range that is slightly higher than twice its transmission range. This means that a given node can only communicate with nodes placed at a maximum distance defined by the transmission range but can interfere with other nodes located at distances up to the interference range.

To prevent the classical hidden node problem, 802.11 includes a four-way RTS/CTS/DATA/ACK exchange, which may be retransmitted up to 7 times in order to recover as much as possible potentially lost packets due to mainly wireless errors. Additionally, 802.11 encompasses a virtual carrier sense mechanism in which each node overhears other nodes' transmission and backs off for a random interval to avoid collisions. For more detail refer to [7]. These mechanisms are effective in scenarios where no more than 3 hops are in place, as explained in the following.

Fig. 1 [8] depicts a chain topology of 6 nodes. The transmission range is represented by the full line and the interference range is shown as dashed line. According to the explanation above, each node can only transmit to its immediate neighbors within its transmission range but can prevent transmission of the other nodes placed up to 2 hops away from it. From Fig. 1, it is clear that the transmission of nodes 2 and 3 will prevent node 1 from transmitting since they are within transmission and interference ranges of node 1, respectively.

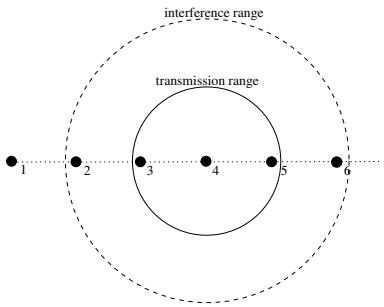


Fig. 1. Spatial reuse in multihop networks

Furthermore, the transmission of node 4 will prevent node 2 from receiving the request (RTS) from node 1 or sending a response (CTS) to it, thereby impacting node 1 transmission as well. Thus, node 1 will only succeed in its transmission when node 4 is done with its transmission to node 5. In other words, in a group of 4 subsequent hops, only one can be active at a time, which imposes a maximum channel utilization of 1/4 in comparison to the 1-hop bandwidth utilization.

This analysis considers an ideal MAC protocol which can schedule the nodes transmission evenly. Nevertheless, 802.11 does not work so smoothly, and because of that if the sender at the beginning of the chain is not able to pace its transmission rate to match the channel capacity, the channel utilization will be decreased. In other words, the factor of 1/4 can only be achieved if the sender injects into the network the exact amount of data that the channel is able to forward.

B. The Optimal limit for TCP Congestion Window Size

The spatial reuse discussed above indicates that a TCP sender should limit the size of its congestion window ($cwnd$) in order to achieve better performance. The $cwnd$ defines the maximum number of packets that a TCP sender may inject into the network at once without waiting for an ACK packet from the receiver. Thus, according to the discussion above, a limit of $h/4$ for $cwnd$ is presumably an optimal setting, where h is the number of hops in the chain topology. This means that long chains with many groups of consecutive 4 hops will have higher limit for $cwnd$ than short chains do. Interestingly, a 4-hop chain will have an ideal limit of 1 packet for the TCP cwnd.

In [9] the authors found that a limit of 3 packets size provided best channel utilization in a 7-hop chain topology, which is slightly higher than the theoretical limit explained above. Their simulation results showed that better match to the $h/4$ is achieved for long chains due to better distribution of the packets in flight among the nodes. A more extensive investigation into this problem has been performed by [5], where the authors propose limiting the $cwnd$ acoording to the round number of hops in place.

By either investigation above, a short chain of nodes of up to 10 nodes, should have a limit of 3 or 4 packets. We have confirmed such results by simulation (with slight variations) which is depicted in Fig. 2. These simulations were performed using ns2 simulator with default settings apart from the packet size that were set to 1460 bytes.

From Fig. 2, we can draw very important conclusions. First, for short chains of nodes of up to 3 hops, even a very small $cwnd$ of 1 or 2 packets size is sufficient to guaranttee maximum throughput. For scenarios of up to 10 hops, which is likely to be enough for a large number of applications as we will elaborate on below, a limit of 3 packets size suffices.

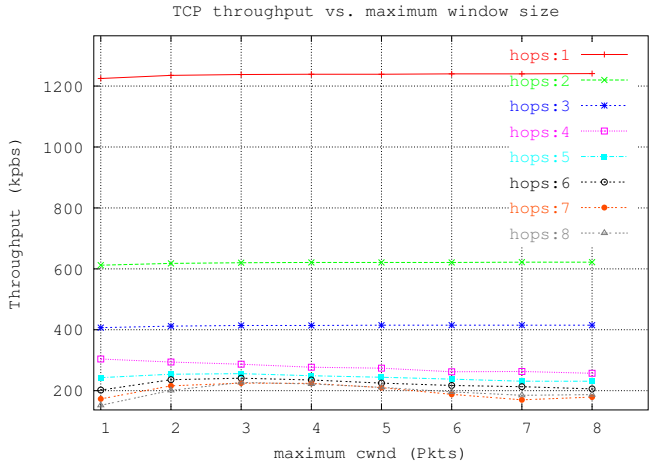


Fig. 2. The optimal limit for the sender congestion window size

As opposed to some related work, such results indicate that, at least for short chains of nodes, it does not help much to avoid decrease in the $cwnd$ size. The problem is that the $cwnd$ should be set to very

small sizes anyway, and yet the difference in throughput for different limits of $cwnd$ size is not that much, as shown in Fig. 2.

IV. Dynamic Adaptive Acknowledgment

In this section, we present the main features of our proposed approach, which we call TCP-DAA. As earlier mentioned, TCP-DAA is meant for feasible scenarios, in which the IEEE 802.11 MAC protocol may provide acceptable performance. Recent investigations on 802.11 have shown that such a protocol is effective in recovering most of the wireless losses in typical scenarios, but it does not scale as the number of wireless hops increase, due to the spatial reuse property discussed in the last section. There are a number of important applications in which the number of hops involved will be far below 10 hops, and the number of nodes will normally not exceed 100 nodes. For instance, typical classrooms, meeting and workshop events, working offices, wifi in home buildings and so on.

In particular, TCP-DAA takes advantage of the fact that the 802.11 protocol schedules the contending transmission requests in such a way that normally every node cannot transmit more than one packet at a time, but must contend for the medium again in order to prevent capture of the medium by a single node. This feature allows a TCP source to send more packets at once without receiving ACKs. However, if the receiver acknowledges every incoming DATA packet, then the probability of collisions between DATA and ACK packets increases considerably. Additionally, since the receiver must also contend for the medium by using the CTS/RTS control frames, the overall overhead at the MAC layer, for transmitting ACKs, is not negligible.

These problems may be mitigated if the sender merges several acknowledgements in just a single ACK, what is possible due to the cumulative ACK scheme used in the TCP congestion control. Figs. 3(a), 3(b) and 3(c) show the transmission delay experienced by both DATA and ACK packets in a chain topology of 3 hops for a simulation run that lasts 10 seconds. It is easy to see that in a configuration without delayed acknowledgments (Fig. 3(a)), the delays experienced by ACK packets are quite significant in comparison to the DATA packets delay.

Using the standard DA, such a delay decreases substantially, and with TCP-DAA the improvement is even higher. Taking the average delay for each run, from top to bottom in Fig. 3, the ACK delay accounted for 40%, 21% and 9,6% of the total transmission time, respectively. Note that in this simulation, the ACK packets with sequence number in the neighborhood of 105 and 360 experienced much higher delay than the average. Such a phenomenon is triggered by spurious timeut at the receiver. We will discuss this issue further below.

The fewer amount of ACKs for the sender might lead TCP to low performance in typical wired scenarios where the congestion window ($cwnd$) limit is usually high. This might happen because a TCP sender may only increase its $cwnd$ size, toward the limit, upon receipt of ACKs. This problem is not so critical, however, in our technique as the $cwnd$ limit in place (4 packets) is rather low. This means that after a reduction of $cwnd$ due to a packet loss, it will quickly reach the limit again upon receiving a few ACKs. Nevertheless, some improvement seems to be possible if the sender increases its $cwnd$ in such a way that compensates the fewer number of ACKs received, as investigated in [21]. This is left for future work.

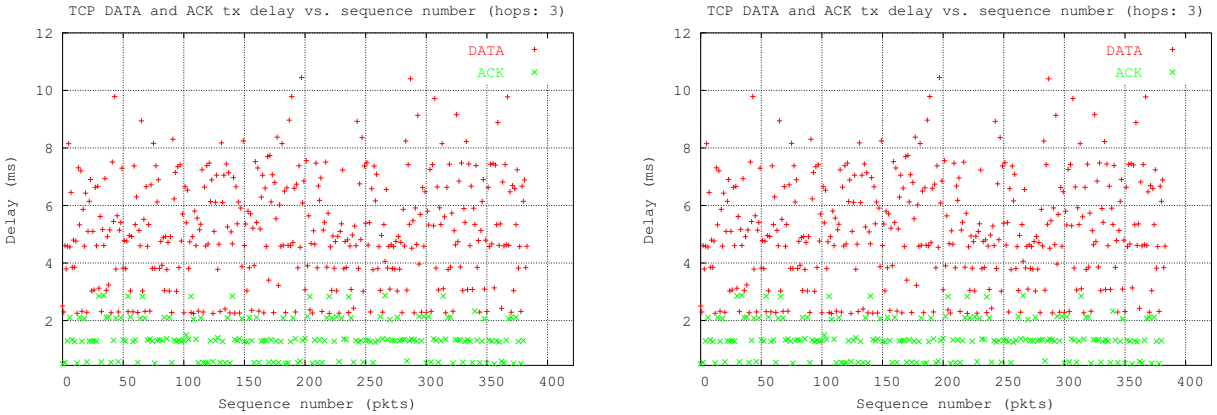
By delaying the acknowledgment notification to the sender, the receiver may trigger a retransmission by timeout at the sender if the receiver delays excessively. Thus, the receiver has to be well adjusted in order to avoid such spurious retransmissions. The standard delayed acknowledgement (DA) proposed in RFC 1122, recommends that a receiver should send one ACK for every other packet received, and should not delay an ACK when either an out-of-order packet or a packet filling in a gap in the receiver buffer is received. Besides, the maximum delay should not exceed a given time interval (typically 100 ms).

Such concepts behind the standard DA work reasonably well in wireless environments, as shown in [22], but higher enhancements are possible by delaying up to four ACK packets rather than only two, as shown in [19]. In the latter, the authors concluded that in a chain topology of nodes, substantial improvement may be achieved by delaying 3-4 ACKs. In their approach they simply delay 4 packets (except at the

startup) or wait for a timeout expiration, which has also a fixed interval, for sending an ACK to the sender. They also considered a large limit for the sender $cwnd$ of 10 packets in order to keep several packets in transit. We will hereafter call this approach LDA (Large Delayed Acknowledgment), because we will compare our results with the results from this technique.

The main problem with both the standard DA and LDA schemes is the fixed timeout interval (100 ms), since the packet inter-arrival at the receiver will change not only with the channel data rate, but also with the intensity of traffic going through the network.

TCP-DAA combines the idea of higher number of delayed ACKs with the dynamic reaction proposed in RFC 1122 (reaction to packets that are either out-of-order or filling in a gap). Furthermore, our protocol adjusts itself to the channel conditions, in that it computes adaptively the timeout interval for the receiver on the basis of the packet inter-arrival interval at the receiver. In this way, the receiver delays just enough to avoid spurious retransmissions at the sender and is able to adapt itself to different levels of delays imposed by the wireless channel, thereby being independent of both channel data rate and number of concurrent flows crossing the network. As we will show in section V, TCP-DAA outperforms the two schemes above (standard DA and LDA) in several scenarios.



(a) No delayed ACK

(b) standard DA

(c) TCP-DAA

Fig. 3. DATA and ACK delay in a typical wireless channel

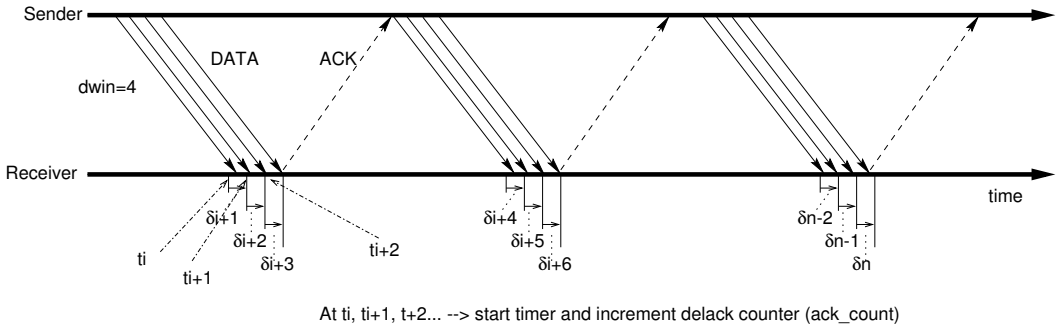


Fig. 4. Timing diagram of TCP-DAA

A. Algorithm

As mentioned above, a smooth interaction between sender and receiver may provide optimal performance in our approach. We are currently investigating the sender side in terms of response to losses and timeout interval computation since both are affected by the delayed ACKs. The technique we used for minimizing unnecessary retransmissions by timeout consists of two adjustments: the number of duplicate ACKs for triggering a retransmission by fast retransmit mechanism is decreased from 3 to 2 packets, and the timeout interval is increased fivefold. These are the only two changes performed on the regular TCP sender, which proved to be effective in most of our evaluations.

Fig. 4 depicts the timing diagram of TCP-DAA under normal conditions, i.e., after startup and without any loss. For every four DATA packets transmitted, the sender responses with an ACK. Whenever a given ACK $_{i,i+1,i+2...}$ is to be delayed, an associated timer is started (t_i) or restarted (t_{i+1}, t_{i+2}) if there is one already running. The receiver also measures the DATA packet inter-arrival gap between the packets for which the ACK is to be delayed ($\delta_{i,i+1,i+2...}$).

Note that the receiver keeps track of the number of ACKs delayed by maintaining an ack_count variable which increases from 1 to the current value of its delaying window ($dwin$) value. By checking the value of ack_count, the receiver is able to determine if the received packet is the first one from the group that is going to have their acknowledgment delayed.

In case such a packet is the first one, then the inter-arrival interval between the last received packet and the current one is not used. This is needed to avoid that unnecessary intervals such as the one between δ_{i+3} and δ_{i+4} in Fig. 4 is improperly taken in the timeout interval computation. By using this strategy, we assure that in normal conditions, the inter-arrival measurements will reflect very closely the gap between the received DATA packets triggering delayed ACKs. Note that under packet loss, the receiver will not need such measurements as it will not delay out-of-order packets. Rather, it will await until it receives in-order packets again.

Similarly to what TCP sender does, the receiver computes a smoothed packet inter-arrival ($\overline{\delta}_i$) at the arrival of a given DATA packet p_i as indicated in (1), where $\overline{\delta}_{i-1}$ refers to the last calculated value, δ_i is the packet inter-arrival sampled, and α is the inter-arrival smoothing factor.

$$\overline{\delta}_i = \alpha * \overline{\delta}_{i-1} + (1 - \alpha) * \delta_i \tag{1}$$

The value computed from (1) is used to set the timeout interval at the receiver. In our design, we established that after the receipt of a DATA packet that causes an ACK to be delayed, it is reasonable to wait for at least the time the second next packet is expected. The rationale here is that the delay variations are relatively high in such environments and in case of a single dropped packet, the next DATA packet will arrive out-of-order, which will trigger immediate transmission of an ACK, as recommended in RFC 1122.

On the other hand, if it was only a delay variation, and the DATA packet arrives before the expected time for the subsequent packet, no timeout will be triggered and the receiver will save an extra and

unnecessary ACK packet from being sent into the network. We then use a timeout interval t_i as shown in (2). Note that the factor 2 in (2) refers to the estimated time for the second expected DATA packet to arrive. This equation also includes a timeout tolerance factor κ which defines how tolerant the receiver may be in deferring its transmission beyond the second expected DATA packet.

$$t_i = (2 + \kappa) * \overline{\delta}_{i-1} \tag{2}$$

Fig. 5 illustrates how the receiver dynamic window ($dwin$) changes. Under normal conditions, it is maintained at maximum size of four packets. Whenever the receiver gets a packet which is either out-of-order or filling in a gap in the receiver buffer, or when its timer expires, then it immediately sends an ACK to the sender and halves $dwin$ to two packet size. Another design option here would be to restart the $dwin$ from size one. We chose to halve it to make our technique to work, just after a response to a lost packet, exactly as the regular DA does.

In our algorithm, subsequent DATA packets will trigger $dwin$ growth toward the maximum size again, provided they are not in one of the three conditions above. Using this dynamic behavior, the receiver prevents the sender from missing ACKs when packet losses occur. As mentioned above, the LDA proposal [19] works with a fixed $dwin$ of size 4 (except at startup), and uses a large $cwnd$ limit at the sender to keep the channel full of DATA packets in flight. While this procedure may prevent a lack of ACKs at the sender, it may also induce excessive number of retransmissions at the sender, as shown below.

The $dwin$ growth is governed by (3) which shows that such an increase may be fixed by one packet size or variable according to the startup speed factor μ. The reason for this factor is that during the startup phase, the sender starts with a window of two packets size and then increases it by one at every ACK received. Although $dwin$ is initialized with one packet size, if it started from the startup increasing at the rate of one packet size, then there would happen a shortage of ACKs at the sender which would not be able to send packets into the network before its retransmission timer had expired.

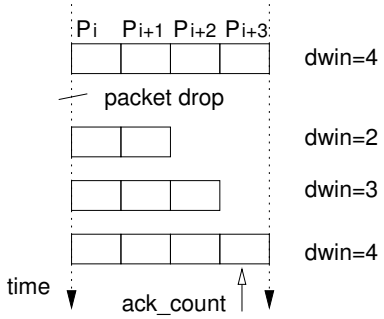
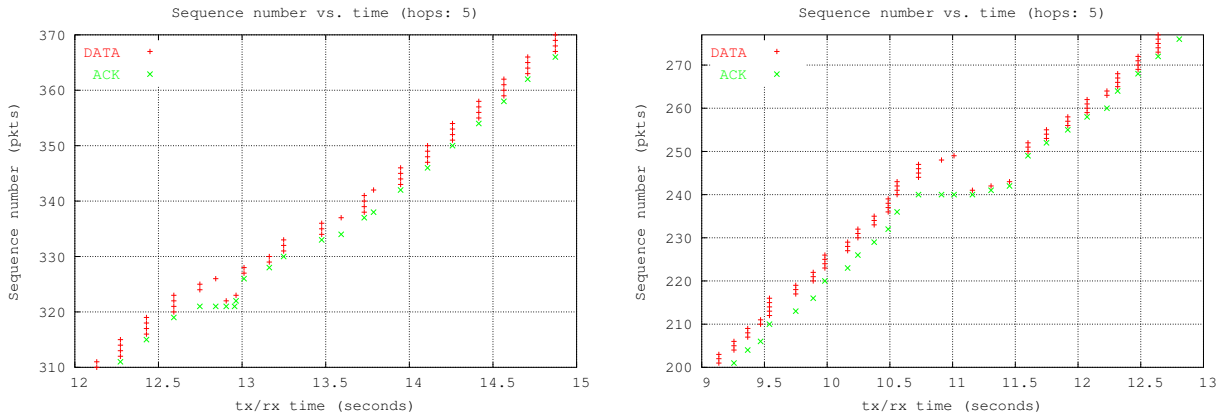


Fig. 5. Receiver dynamic window for delaying packets

Hence, the threshold $maxdwin$ is used to define the instant the startup phase is over, which occurs when $maxdwin$ first reaches its maximum value. From our evaluations, we noticed that by properly setting the μ parameter, our algorithm achieved better performance for short-term transmissions.

$$dwin = \begin{cases} dwin + \mu, & \text{if } maxdwin = false \\ dwin + 1, & \text{otherwise} \end{cases} \tag{3}$$

The mechanisms just explained make TCP-DAA effective because whenever it is possible it uses the scarce channel bandwidth efficiently, and when the channel is facing really poor conditions, it performs in general as effective as the other implementations. Using its dynamic adaptive window, TCP-DAA somehow probes the network for resource availability, since it will always delay more packets (up to 4) when the network so permits. Furthermore, our protocol is TCP friendly as it preserves the basics of TCP-friendly behavior that is to decrease transmission rate under loss detection and to increase it if the network so allows.



(a) TCP-DAA

(b) LDA

Fig. 6. Delayed acknowledgment strategies

B. Reaction to packet losses

In order to fix the concepts explained above, we show here a typical response of our mechanism when reacting to lost packets. We include the response of the LDA scheme to highlight the difference of our proposal to that one. Fig. 6 shows part of a simulation run in which both strategies faced a dropped packed in a chain topology of 5 hops.

Let $packetn$ be the DATA packet of sequence number (n). Fig. 6(a) shows that the sender transmits four packets (320-323) at 12.6 seconds. In this run, $packet322$ and $packet323$ were dropped, and so the receiver times out and acknowledges only two packets (320, 322) instead of four. The receiver also updates its $dwin$ to the size of two packets. Upon the receipt of the ACK for $packet321$, the sender sends two new packets (324, 325), because two sent packets were acknowledged. At this moment there are only 2 packets in flight (324, 325). Since $packet324$ and $packet325$ are detected by the receiver as out-of-order packets, they trigger immediate acknowledgments at the receiver. By receiving the first duplicate ACK, the sender transmits a new packet (326) which will also be out-of-order.

When the sender receives the second duplicate ACK at 12.9 seconds, it retransmits the first lost packet (322), and halves its $cwnd$ to the size of 2 packets (fast retransmit/fast recovery). The $cwnd$ will be increased gradually after the sender leaves the fast recovery phase. When the sender receives the third duplicate ACK, at 12.96 seconds, it does nothing because it is in the fast recovery phase. At the instant 12.97 seconds, the sender gets the acknowledgment for $packet323$ and then it retransmits this packet and leaves the fast recovery procedure. $Packet323$ fills in the gap at the receiver buffer, which triggers the ACK of $packet326$ due to the cumulative property of TCP acknowledgment strategy.

At the instant 13.01 seconds, the sender receives the acknowledgment for $packet326$, and so transmits two new packets (327, 328). These two packets cause the receiver to send one ACK only as its $dwin$ is set to 2 packet at this point. After that, the $dwin$ increases and, as a consequence, the number of delayed ACK increases toward 4. Fig. 6(a) shows two spurious retransmissions by timeout at the receiver. $Packet339$ and $packet338$ are unnecessarily acknowledged at the instants 13.62 and 13.79, respectively. This means that the timeout interval computation may still be improved, but that is left for future work.

Fig. 6(b) shows the response of LDA to a packet loss. In this simulation run, the $packet241$ was lost at about 10.55 seconds. Differently from our technique in which the amount of packets in flight are limited to 4 packets, the proposed LDA works with a large limit for the $cwnd$ (10 packets), so it has more packets in flight than TCP-DAA does. One can notice in Fig. 6(b) that although only one packet was dropped, various acknowledgments triggered the transmission of less than the optimal 4 packets. This shows that the retransmission timer expired in several unnecessary situations.

Additionally, the sender waits for the default 3 duplicate ACKs for retransmission the lost packet, which inccurs in longer time for it to take action. In short, by comparing Fig. 6(a) with Fig. 6(b), one can clearly see that TCP-DAA provides more stability in terms of the number of delayed ACKs. As a result, less packet delay variation is perceived by the sender, which in turn tends to minimize the inaccuracy in the timeout interval computation at the sender.

V. Performance Evaluations

Here we evaluate and compare the performance of TCP-DAA with the other TCP flavors and with the LDA proposal presented in [19]. We compare our work with LDA because it also investigates delayed acknowledgments strategy for improving TCP performance in multihop networks. We also compare our results with other TCP flavors in their theoretical best conditions, so as to make sure that our proposal is indeed efficient among a wide range of options. Hence, we simulate the other TCP flavors including two potential improvements: the standard delayed acknowledgement (DA), and a low limit for their $cwnd$ (3 packets) for the sake of bandwidth utilization, as explained in section 1.

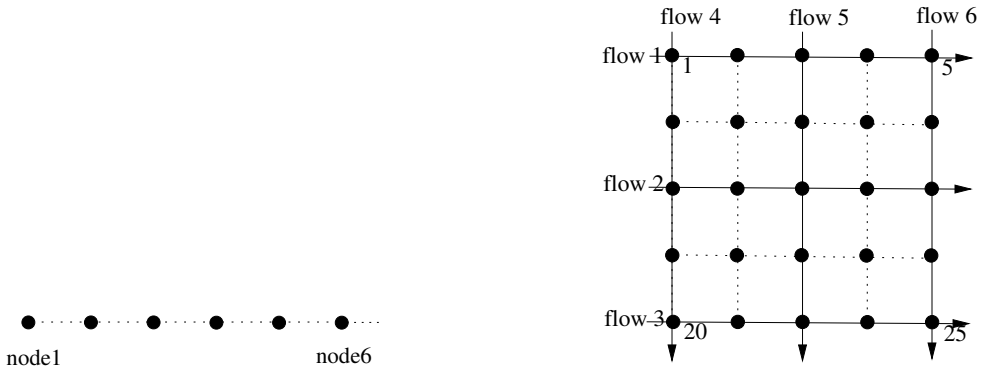
A. Simulation setup

We used ns2 simulator in our evaluations over the two scenarios depicted in Fig. 7 in which we have a single chain topology and a grid topology with 25 nodes. In both topologies, each node is 200 meters apart from its closest neighbors and the wireless data rate is 2 Mbps. In the simulator, the effective transmission range is 250 meters while the interference range is 550 meters in accordance with 802.11 MAC protocol. In all evaluations, the throughput bw is calculated as $bw = \frac{seq*8}{stime}$, where seq is the maximum sequence number (in bytes) transmitted and acknowledged and $stime$ is the simulated time.

The parameters settings are as follows. Packet size is 1460 bytes, the window limit (WL) for the other flavors (except for the regular TCP) is 3 packets, the regular TCP is the Newreno flavor; and for TCP-DAA, α is 0.75, κ is 0.2 and μ is 0.3. The other parameters were kept as the default of the simulator and all simulation runs lasted 300 seconds.

B. Retransmissions

Since TCP-DAA is more tolerant to packet delay variations by having the regular RTO (retransmission timeout) at the sender multiplied by a factor of 5 and a dynamic acknowledgment strategy at the receiver, it is expected that it minimizes spurious retransmissions by timeout.



(a) chain topology (b) grid topology

Fig. 7. Simulated scenarios

Fig. 8 shows the result of a simulation run in which 10 flows shared the medium in the chain topology of Fig. 7(a) under different number of hops. The figure exhibits the aggregate number of retransmissions including both retransmissions by timeout and by the fast retransmit mechanism.

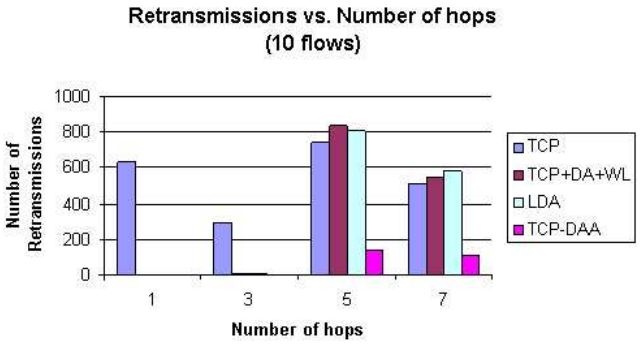


Fig. 8. Aggregate number of retransmissions

It can be seen that for 1 and 3 hops scenarios, our protocol did not retransmit any packet at all, while the regular TCP retransmitted quite a lot. For 5 and 7 hops, our mechanism provided significant enhancement by retransmitting much less than the other algorithms did. In these evaluations, TCP-DAA retransmitted from 32 to 100% less than the regular TCP did and 29 to 100% less than the proposed LDA did. This outcome is doubtlessly expressive in terms of power consumption benefits.

C. Throughput in a chain topology

In this section, we investigate the bandwidth utilization over a wide range of situations. While most related work present results over either a single flow or a fixed number of hops, we show here our evaluation over not only different number of hops but also different number of concurrent flows in the chain topology of Fig. 7(a). In addition, we compare our results with all the other TCP flavors, which is usually also not done in most cases. It would be reasonable only to compare our algorithm with the other flavors, including regular TCP, with DA enabled. However, as the vast majority of related work present results over the regular TCP without DA, we also evaluate this configuration here for easing comparison with other related work.

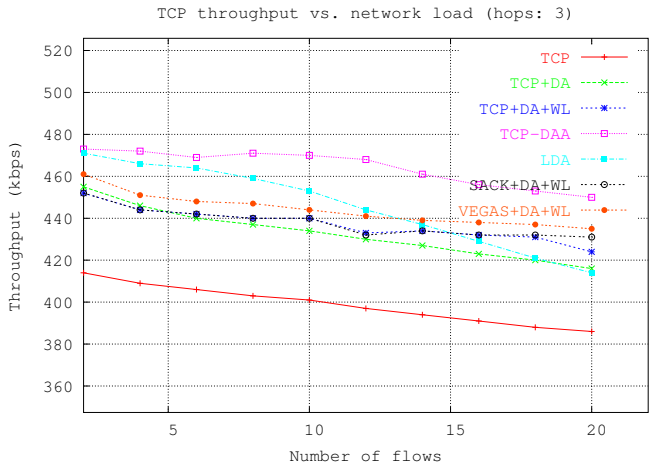
Fig. 9 exhibits a remarkable achievement of TCP-DAA. These results were obtained by taking the average of 5 runs. TCP-DAA outperforms all the other algorithms in most situations. Only TCP-Vegas and TCP, both with optimal setting (DA+WL), outperform slightly our protocol in the scenario with 7 hops and 2 concurrent flows. Nevertheless, as the number of flows increase, the performance of both strategies degrades significantly while in our scheme it does not happen. We believe that TCP-DAA will be improved if the default sender RTO calculation is fine tuned to this strategy.

It is interesting to note that, in general, the more flows the better the improvement of our algorithm over the other protocols. The reason for that is the high level of transmission delays due to the higher number of flows in the network. Under such high delays, the packet delay variance becomes less significant in the RTO computation, and so less interference of the delayed ACKs is perceived by the sender.

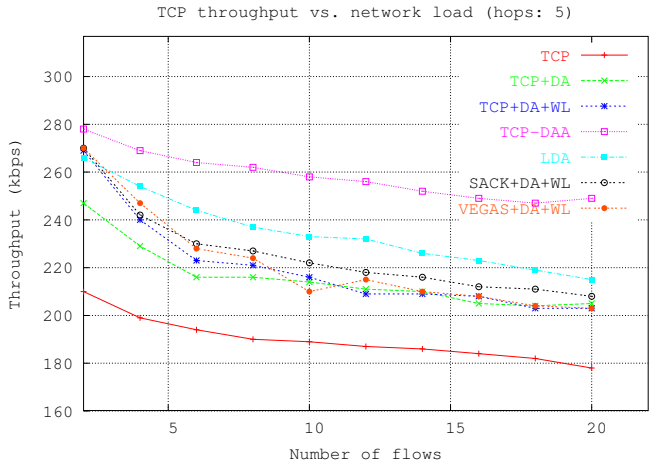
Overall, the improvements are up to about 50% over the regular TCP whether it was using DA or not. Over the LDA, up to 30% was obtained. We also performed simulations for 2, 4 and 6-hop scenarios and the results were similar, and in some cases less improvement were observed, but in most cases our algorithm performed better than all the others.

D. Throughput in a grid topology

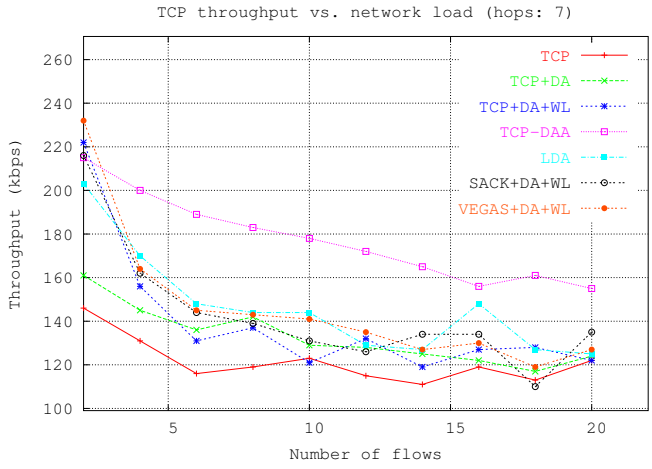
Here we describe the investigations carried out in a more complex scenario, the grid topology illustrated in Fig. 7(b). In these evaluations, we had first only 3 flows crossing the topology horizontally (flows 1,



(a) 3-hop



(b) 5-hop



(c) 7-hop

Fig. 9. Aggregate throughput in a chain topology

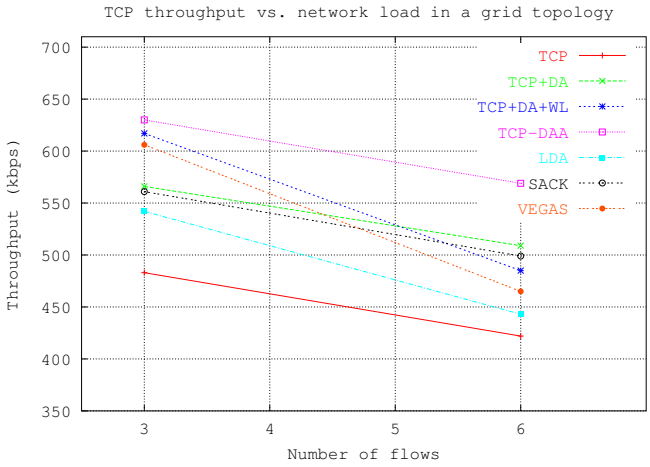


Fig. 10. Aggregated throughput in grid topology with cross traffic

2 and 3 in Fig. 7(b)). In the next step the 6 flows (3 horizontal and 3 vertical) were injected into the network concurrently, and the results are depicted in Fig. 10.

These results refer to the best outcome of TCP-DAA out of 5 simulation runs. The idea here was to emphasize that TCP-DAA may perform quite well in a more complex topology. However, due to the various interactions among the nodes sharing the medium, optimal improvement is not always possible because of the existing unfairness at the MAC layer. That is the inherent drawback of using 802.11 MAC protocol. The obtained results show that again the performance improvement of our algorithm was superior under larger number of flows. That is, better improvements were obtained for the condition in which the network had 6 flows.

In terms of throughput improvement, TCP-DAA provided a throughput increase of up to 13, 28 and 35% over regular TCP, regular TCP with DA and the proposed LDA, respectively. These results are also very important in that they show that our algorithm may interact effectively with the 802.11 MAC protocol, even when several nodes are competing for the medium. However, more evaluations are needed here in order to determine to what extent this smooth interaction may be sustained.

E. Discussions

In general the results were quite positive in the sense that TCP-DAA outperformed the other algorithms in most cases. Using our technique, a TCP connection may improve performance when the wireless channel is facing reasonable conditions and should perform as effective as a regular TCP does otherwise. The evaluation results went further to show that even under a certain level of loss rate, our mechanism can still provide enhancements. Furthermore, the bandwidth utilization is improved along with power consumption gain.

Nonetheless, there is surely room for improvements. First, the end-to-end throughput enhancements were in general not very significant for scenarios with low number of concurrent flows. This may be changed by fine tuning the algorithm used by sender and receiver for computing the timeout intervals involved. Second, the TCP-DAA parameters (α, κ and μ) values were determined via simulation using coarse granularity, which means that they may also eventually be improved. Third, the sender side may increase the $cwnd$ smartly to compensate the lack of ACKs due to the delayed strategy. Fourth, it seems that better results may be achieved for grid topologies if the MAC layer fairness is improved.

VI. Conclusions

We have introduced and evaluated our algorithm for improving TCP performance in multihop wireless networks. The key idea of our dynamic adaptive acknowledgement strategy is to provide capability to a

TCP receiver to adjust itself, according to the channel conditions, in terms of the ratio of DATA packet to ACK packets.

The outcome of the simulation evaluations showed that our algorithm outperforms not only the regular TCP and the main TCP flavors, but also similar techniques that have been proposed in the literature, in a variety of conditions. Our scheme improves throughput and power consumption, which are two key issues in such environments. Finally, it is easy to deploy since the changes are limited to the end nodes only.

We do not claim that our technique is the optimal acknowledgment strategy for a TCP implementation in multihop networks, but the achieved results are indeed encouraging, which justifies further investigation on this direction. Open issues for future work include: fine tuning of the timeout interval computation at both sender and receiver, optimal $cwnd$ increase method for the sender receiving fewer ACKs, and investigation on alternative MAC layer strategies.

REFERENCES

[1] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback based scheme for improving tcp performance in ad-hocwireless networks. Amsterdam, May 1998. 18th International Conference on Distributed Computing Systems (ICDCS).

[2] G. Holland and N. H. Vaidya. Analysis of tcp performance over mobile ad hoc networks. Seattle, August 1999. Annual International Conference on Mobile Computing and Networking (Mobicom'99).

[3] J. Liu and S. Singh. Atcp: Tcp for mobile ad hoc networks. volume 19, pages 1300–1315. IEEE Journal on Selected Areas in Communications, July 2001.

[4] Z. Fu, B. Greenstein, X. Meng, and S. Lu. Design and implementation of a tcp-friendly transport protocol for ad hoc wireless networks. 10th IEEE International Conference on Network Protocosls (ICNP'02), November 2002.

[5] K. Chen, Y. Xue, and K. Nahrstedt. On setting tcp's congestion window limit in mobile ad hoc networks. Anchorage, Alaska, May 2003. IEEE International Conference on Communications (ICC 2003).

[6] IEEE. Wireless lan medium access control (mac) and physical layer (phy) specifications - std 802.11. The Institute of Electrical and Electronics Engineers, 1999.

[7] R. Oliveira and T. Braun. Tcp in wireless mobile ad hoc networks. University of Bern, Technical Report IAM-02-003, July 2001.

[8] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless network. Rome, Italy, July 2001. ACM MOBICOM'01.

[9] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on tcp throughput and loss. San Francisco, USA, April 2003. Infocom'03.

[10] A. Bakre and B. Badrinath. I-tcp: Indirect tcp for mobile hosts. pages 136–143, Vancouver, Canada, May 1995. IEEE ICDCS'95.

[11] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving tcp/ip performance over wireless networks. Vancouver, Canada, November 1995. 1st ACM Mobicom.

[12] K. Brown and S. Singh. M-tcp: Tcp for mobile cellular networks. volume 27, pages 19–43. ACM Computer Communications Review, 1997.

[13] V. Tsaoussidis and H. Badr. Tcp-probing: Towards an error control schema with energy and throughput performance gains. Japan, November 2000. 8th IEEE Conference on Network Protocols.

[14] C. Zhang and V. Tsaoussidis. Tcp-probing: Towards an error control schema with energy and throughput performance gains. New York, June 2001. 11th IEEE/ACM NOSSDAV.

[15] S. Biaz and N. H. Vaidya. Distinguishing congestion losses from wireless transmission losses:a negative result. New Orleans, LA, USA, October 1998. IEEE 7th Int. Conf. on Computer Communications and Networks.

[16] J. Liu, I. Matta, and M. Crovella. End-to-end inference of loss nature in a hybrid wired/wireless environment. INRIA Sophia-Antipolis, France, March 2003. WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks.

[17] R. Oliveira and T. Braun. A delay-based approach using fuzzy logic to improve tcp error detection in ad hoc networks. Atlanta, USA, March 2004. IEEE WCNC 2004.

[18] T. Yuki, T. Yamamoto, M. Sugano, M. Murata, H. Miyahara, and T. Hatauchi. Performance improvement of tcp over an ad hoc network by combining of data and ack packets. to apper in IEICE Transactions on Communications, 2004.

[19] T. Jimenez E. Altman. Novel delayed ack techniques for improving tcp performance in multihop wireless networks. Venice, Italy, September 2003. Personal Wireless Communications (PWC'03).

[20] A. Kherani and R. Shorey. Throughput analysis of tcp in multi-hop wireless networks with ieee 802.11 mac. Atlanta, USA, March 2004. IEEE WCNC'04.

[21] M. Allman. On the generation and use of tcp acknowledgements. volume 28, pages 1114–1118. ACM Computer Communication Review, 1998.

[22] S. Xu and T. Saadawi. Does the ieee 802.11 mac protocol work well in multihop wireless ad hoc networks? volume 39, pages 130–137. IEEE Communications Magazine, june 2001.