Alleviating Effects of Mobility on TCP Performance in Ad Hoc Networks using Signal Strength based Link Management

Fabius Klemm[1χ], Srikanth V. Krishnamurthy[2], and Satish K. Tripathi[2]

[1] Department of Computer Science,
EPFL, Lasaunne, Switzerland
Fabius.Klemm@epfl.ch
[2] Department of Computer Science and Engineering,
University of California, Riverside, CA, 92521
{krish,tripathi }@cs.ucr.edu

Abstract. Mobility in ad hoc networks causes link failures, which in turn result in packet losses. TCP attributes these losses to congestion. This results in frequent TCP retransmission timeouts and degradation in TCP performance even at light loads. We propose mechanisms that are based on signal strength measurements to alleviate such packet losses due to mobility at *light loads*. Our key ideas are (a) if the signal strength measurements indicate that a link failure is most likely due to a neighbor moving out of range, *in reaction*, facilitate the use of temporary high power to re-establish the link and, (b) if the signal strength measurements indicate that a link is likely to fail, initiate a route re-discovery *proactively* before the link actually fails. We make changes at the MAC and the routing layers to predict link failures and estimate if a link failure is due to mobility. We also propose a simple mechanism that can help alleviate *false link failures* that occur due to congestion when the IEEE 802.11 MAC protocol is used. We compare the above proactive and reactive schemes and also demonstrate the benefits of using them together. We show that, in high mobility, the performance of a TCP session can increase by as much as 45 percent when our methods are incorporated.

Key Words: Power Management, Ad Hoc Networks, TCP, Signal Strength, IEEE 802.11

1. Introduction

TCP performs poorly in wireless ad hoc networks as demonstrated in [8, 12]. The main reason for this poor performance is a high level of packet losses and a resulting

This work was partially funded by the DARPA Fault Tolerant Networks (FTN) contract no: F30602-01-2-0535.

χ This work was done when the author was with University of California, Riverside.

high number of TCP retransmission timeouts. First, a node drops a packet if it cannot forward the packet to the next hop of the route as the next hop node has moved out of transmission range. A second reason for packet loss is congestion in the shared medium. In this case, a node cannot reach the next hop node because there are too many nodes trying to access the channel at the same time. This might even result in a node *capturing* the medium of access if the IEEE 802.11 MAC protocol is used [12]. While congestion can degrade the observed performance of TCP even in wire-line networks, mobility causes a degradation of performance of TCP in ad hoc networks even at very light loads. Our objective in this paper is to mainly stem the degradation of TCP performance due to mobility.

Towards this goal, we propose mechanisms to reduce the number of packet losses. These mechanisms are based on signal strength measurements at the physical layer. Based on these signal strength measurements, when a node *fails* to communicate with a neighbor, the MAC layer at the node guesstimates if the failure is due to congestion or due to the neighbor moving out of range. If the MAC layer predicts that the neighbor has just moved out of range, then it stimulates the physical layer to increase its transmission power and attempts to re-establish the link to the neighbor temporarily. It also prompts the routing layer to search for a new route. The signal strength measurements can also be used to predict possible link failures to a neighbor that is about to move out of range. Thus, if the measurements indicate that the signal strength is going down and the link is likely to break, a search for a new route can be proactively initiated before the link actually fails. While searching for the new route, the routing layer should take care to avoid either the temporary high power link or the weak link (as the case may be). We have made modifications to the ad hoc on-demand distance vector (AODV) routing protocol [11] such that it precludes the use of such links during the computation of a new route. In order to cope with failures that are not due to mobility, we have included a simple mechanism by which, the MAC layer, upon guesstimating that the neighbor is within range, *persists* in its attempt to reach that neighbor for a longer period of time. We re-iterate that our goals are mainly to cope with the effects of mobility on TCP. At high loads, it is more likely that congestion dominates packet losses. In the simulation experiments that we perform to evaluate our schemes, we therefore restrict ourselves to conditions of light load (one or two TCP connections). In such scenarios we show that the performance of a TCP session can improve by as much as 45%.

The use of signal strength and a count of the transmitted packets in the local neighborhood (nodes can overhear other packet transmissions) can provide an estimate of whether or not there is congestion in the local vicinity of a node. A node should only increase its transmission power if the network is lightly loaded. If there is congestion, temporary increases in power levels can actually increase the number of collisions and increase the congestion. This could degrade the performance further. However, congestion estimation mechanisms are a focus of further study and are beyond the scope of this paper.

Several researchers have been working on improving TCP for wireless ad hoc networks. Holland and Vaidya [8] demonstrated that node mobility causes TCP throughput to drop considerably due to "TCP's inability to recognize the difference between link failure and congestion". They introduced the *Explicit Link Failure Notifications (ELFN)* to allow TCP to react appropriately to link failures. Similar failure notification schemes are presented in [4] and [5]. In [13], the authors propose to split long TCP sessions into multiple segments. By doing so, they argue that, even if a link failure occurs on one of these segments, data flow can be sustained on other segments. The previous schemes however, are unable to salvage TCP packets that are in transit. Our efforts are to salvage packets in transit if a link failure is due to mobility as opposed to congestion.

To the best of our knowledge, ours is the first work to use lower physical layer features such as signal strength and adaptive transmission power levels to improve TCP performance in ad hoc networks. The methods that we propose can be used with the User Datagram protocol (UDP) as well. However, since the effects are unlikely to be as profound we do not consider UDP in this paper.

The rest of this paper is organized as follows: In section 2, we explain the causes of high packet loss. Section 3 will introduce methods to decrease packet loss in ad hoc networks. Section 4 will present the simulation setup and the simulation results. We shall present our conclusions in section 5.

2. Packet Losses in Ad Hoc Networks

In wireless ad hoc networks, TCP performance is affected by packet losses. *Node mobility* and *link layer congestion* are the main reasons for packet loss. A link failure on an active path due to mobility causes the MAC protocol to report a link failure to the routing layer. The routing layer will then have to re-compute routes to the appropriate destinations. When the IEEE MAC protocol [9], which is popular for ad hoc networks, is used, when congestion occurs *false link failures* may be induced. Since our methods should be invoked only when there is a true link failure due to mobility, it is important to correctly identify such failures.

A false link failure occurs when the MAC protocol at a node, say, N_0 declares that the link to a neighbor N_1 is broken, even though N_1 is within its transmission range [1]. The MAC protocol at N_0 fails to establish an RTS-CTS handshake because N_1 cannot respond to its RTS message because it senses another transmission in its vicinity. This is a direct result of the following: In models used, it is often assumed that each node has a transmission range of 250 meters and an interference range of 550 meters[1]. Nodes within the *transmission range* of a node N_0 can receive packets from N_0. Nodes that are not within the transmission range but are within *interference range*

[1] The network simulator ns-2 uses these values for the transmission and interference range.

can sense a transmission from N_0 but cannot successfully receive packets from it. These nodes are also precluded from performing transmissions if N_0 is in the process of transmitting. Thus, upon receiving RTS messages, they will have to ignore these messages.

At the network layer, the routing protocol has to react appropriately to route failures. We discuss how AODV behaves in light of a link failure report from the MAC layer as we use AODV [11] in our simulations later. AODV would simply drop the packets that are to be routed on the failed link. It brings down the routes to the destinations using the failed link and generates *route error* messages and sends such a message to the source of each connection that uses that link.

3. Reducing Link Failures to Improve TCP Performance

We propose mechanisms that help alleviate packet losses due to link failures due to mobility. Our mechanisms are based on measuring the signal strength at the physical layer. As pointed out in Section 2, it is important to first estimate if those failures that are due to mobility. False link failures, discussed earlier, cannot be overcome by tuning power levels. We propose a simple way to identify and cope with false link failures. The methods we propose however only work at light loads and will have to be complemented by other techniques that can estimate the congestion/load in the network.

3.1 Reducing False Link Failures

The IEEE 802.11 MAC reports a link failure if it cannot establish an RTS-CTS handshake with a neighbor within seven RTS retransmissions [9]. Our idea is to send out more than eight RTSs if the probability that the neighbor is still within transmission range is high. We call our version of the MAC protocol the *Persistent MAC*.

In order to determine if a node is still within range, a node keeps a record of the received signal strengths of neighboring nodes. Received signal strength measurements are taken at the physical layer. When a node receives a packet from a neighbor, it measures the received signal strength P_r. The node then observes how P_r changes over time. This would provide an indication as to whether the node is still within range or has probably moved out of range.

For our implementation with the network simulator ns-2, we used the received signal strength P_r to calculate the distance to the transmitter of the packet. Ns-2 uses the *free-space propagation model* as described in [7]. Using this propagation model, the distance d to the transmitter of a packet can be calculated as follows:

$$d = \sqrt[4]{\frac{P_t \cdot G_t \cdot G_r \cdot h_t^2 \cdot h_r^2}{P_r \cdot L}}, \qquad \textbf{(1)}$$

where, P_t is the default transmission power and P_r the received signal power; G_t and G_r are the antenna gains of the transmitter and the receiver respectively; h_t and h_r are the heights of the antennas, and L is the system loss, which is set to 1 by default. We assumed that the network was homogenous, i.e. all nodes use the same parameters P_t, G_t, G_r, h_t, h_r, and L. If a node transmits with a different signal power P_t, it must include the value of P_t in the options field of the MAC protocol header.

The MAC protocol keeps a record of the distances to neighboring nodes in a *neighbor table.* A table entry consists of five fields: a neighbor ID, a distance d_1 to the neighbor (estimated using **(1)**), the time at which this distance was estimated t_1, and a distance d_2 to the same neighbor estimated at time t_2. When a node receives a packet from a neighbor N_Y, it replaces the older entries of the table, corresponding to that neighbor, with the more recent ones. For simplicity, in our models, we have used only two timestamps and have assumed linear node movement. Thus, at any given time t, we estimate the current distance, d_{est} , as follows:

$$d_{est} = d_2 + \frac{d_2 - d_1}{t_2 - t_1} \cdot (t - t_2), \quad \text{for } t_1 < t_2 < t. \qquad \textbf{(2)}$$

If a node N_X cannot establish an RTS-CTS handshake with a neighbor N_Y, it uses the neighbor table to estimate the current distance to N_Y. If d_{est} is smaller than the transmission range of N_X, Persistent MAC will send out up to eight[2] additional RTSs to establish a handshake with N_Y. If d_{est} is greater than the transmission range, or if the information in the neighbor table about N_Y is too old, the Persistent MAC will report a link failure to the routing protocol. The persistent MAC will also report a link failure if the additional attempts to establish a handshake with N_Y were to fail. Note here that the increase in RTS is not likely to increase the actual congestion since the RTS messages are sent only if the channel is sensed idle. Furthermore, for each RTS failure, the node still continues its back-off process, which in turn, would give ample time for the congestion to abate.

We re-iterate that this is a simple methodology to reduce the number of false link failures. During periods of high load, the link failures may mainly be due to congestion. In such cases, a more sophisticated methodology may be required to estimate congestion. Our main objective is to use power management to improve TCP performance in MANETs when the network is lightly loaded. Towards this goal, the above mechanism is useful in differentiating between true and false link failures.

We also note that the linear model to estimate distances is a simple method used to evaluate our mechanisms. We expect that the absolute value and the gradient of the

[2] This number will typically be a system parameter. The choice will depend on the density of the network and the congestion levels.

received signal strength might be indicative of whether or not a node is moving out of range and may even be more realistic in practice. However, one might expect similar results with such methods.

3.2 Signal Strength based link management methods

We propose two mechanisms for alleviating the effects of mobility on TCP. We call these the *Proactive* and the *Reactive Link Management (LM)* schemes. These schemes are implemented at the MAC layer. We also provide a *modification of AODV* at the network layer that can exploit the presence of the link management schemes. Proactive LM tries to predict link breakages, whereas Reactive LM temporarily reestablishes a broken link with higher transmission power to salvage packets. The modified AODV allows the forwarding of packets in transit on a route that is going down while simultaneously initiating a search for a new route.

3.2.1 Proactive Link Management

The idea of Proactive LM is to inform the routing protocol that a link is going to break *before* the link actually breaks. The link break prediction mechanism uses the information from the neighbor table, which we described in section 2.1. Proactive LM estimates the projected distance to a neighbor in the immediate future. For example, if the current time is t, the distance of a particular neighbor, $d_{0.1}$, at $(t + 0.1)$ seconds is:

$$d_{0.1} = d_2 + \frac{d_2 - d_1}{t_2 - t_1} \cdot \left(t + 0.1 - t_2\right), \text{ for } t_1 < t_2 < t. \qquad \textbf{(3)}$$

Proactive LM informs the routing protocol as soon as $d_{0.1}$ becomes greater than the transmission range. The routing protocol then informs the packet source, which stops sending and initiates a route discovery[3]. In this example, packets in transit have 0.1 seconds to traverse the *weak* link. If the warning comes too late, the weak link breaks before all packets in transit can be salvaged. On the other hand, the warning should not come too early, as we want to use the link as long as possible.

In [2], Goff, Abu-Ghazaleh, Kahvecioglu, and Phatak implemented a similar mechanism to predict link breakages. This mechanism also measures the signal strength of received packets. A node informs the packet source when the link to the next hop is close to breaking. The packet source initiates a route request but does not stop sending. If the route discovery is successful, the source switches to the new route.

3.2.2 Reactive Link Management

Reactive LM temporarily increases the transmission range of a node to *reestablish* a broken link. Packets in transit can then traverse the reestablished *high power link.* A

[3] We shall describe in detail how the modified AODV reacts to the notifications from Proactive LM in a later sub-section.

node N_X tries to set up a high power link if the RTS-CTS handshake (to a neighbor N_Y) with *default* transmission power is not successful. N_X therefore sends RTSs with high transmission power. N_Y must also switch to high transmission power to send the CTS. Otherwise, the CTS would not reach N_X. The RTS frame must therefore contain the value of the transmission power P_t. N_X and N_Y also send the DATA and the ACK packets with high power. When Reactive LM at N_X establishes the temporary high power link to N_Y, it stimulates the routing protocol to begin a new route discovery.

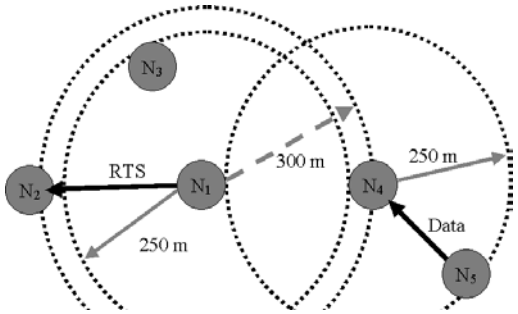

Fig. 1. Effects of Power Asymmetry

Reactive LM maintains a table to record the default and high power links. A node should be able to change its transmission power quickly, because it should not use high transmission power to communicate with neighbors that are within default transmission range. Furthermore, nodes must not broadcast *Route Request* messages from AODV with high transmission power since new routes should consist of default power links only. As Reactive LM does not use signal strength to estimate the distance to a neighbor, it also raises the transmission power in case of false link failures. However, it can be combined with Persistent MAC, which helps avoid this effect.

The IEEE 802.11 MAC Collision Avoidance (CA) mechanism does not work well if nodes have different transmission ranges [6]. Figure 1 shows an example, in which N_1 increases its transmission range from 250 to 300 meters to reestablish the link to N_2. N_1 does not know about the DATA transmission from N_5 to N_4, because it could not receive N_4's CTS. N_1 therefore disturbs the DATA transfer from N_5 to N_4. Due to this effect, as congestion or the load in the network increases, increasing transmit power can in fact degrade performance. Thus, this method should be incorporated only at light loads.

3.2.3 Modifications to AODV

Proactive and Reactive LM inform the routing protocol of either weak or high power links. In this subsection, we explain how our modified version of *AODV* reacts to these MAC layer notifications. The routing protocol does not necessarily have to distinguish between *weak* and *high power links*. In both cases, the objective is to inform the packet source of the link failure, initiate a new route discovery and to salvage the packets in transit. In AODV, a route to a destination N_D in the routing

table of a node N_X can be in either of two states: The route can be UP, which means N_X forwards packets to N_D. If N_X receives a *Route Request (RREQ)* for N_D, it will respond with a *Route Reply (RREP)* because it knows a route to N_D. The second state is DOWN; in this state, N_X does not have a routing entry for N_D. If N_X wants to send packets to N_D, it will initiate a route discovery. If N_X receives an RREQ for N_D, it will broadcast the RREQ. If N_X receives a packet for N_D, it will drop the packet and respond with a *Route Error (RERR).*

We have added an *additional route state* to allow nodes to salvage packets in transit. This state has the following characteristics: If N_X receives a packet for N_D, it will *forward* the packet. If it receives a RREQ for N_D, it will not reply with a RREP but *broadcasts the RREQ*. If an application at N_X wants to send packets to N_D, the modified AODV will initiate a route discovery. We call this third route state the *Going Down (GDWN)* state. Figure 2 gives an example, in which the MAC protocol at N_2 informs the modified AODV that the link to N_3 is getting weak (or has become a temporary high power link). The modified AODV then, sends a GDWN packet to its active neighbor N_1, which also sends a GDWN packet to N_0. All three nodes N_0, N_1, and N_2 change the route state for destination N_4 from UP to GDWN. N_1 and N_2 keep forwarding TCP transit packets towards N_4, but N_0 stops sending packets and initiates a route discovery. The old route to N_D via N_1, N_2, and N_3 is then no longer used and finally times out, i.e. the route state is set to DOWN. If the MAC protocol reports a *link breakage,* the modified AODV behaves like the original AODV, i.e. it brings down the route to destination N_D, sends RERR messages to its active neighbors, and drops all packets in transit to N_D.

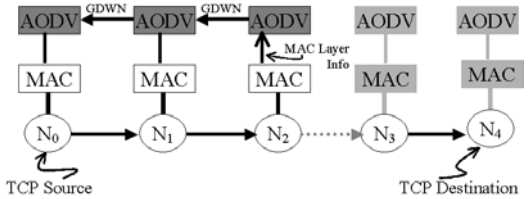

Fig. 2. Modifications to AODV

3.2.4 Transport Layer

The methods we presented in the previous subsections are aimed at reducing the number of packet drops. TCP Tahoe, Reno, and New Reno grow the congestion window until packets are dropped. In wireless ad hoc networks, congestion does not lead to buffer overflow as in wired networks, but rather to false link failures, which cause the routing protocol to bring down the route. Therefore, even in static networks, where we would expect stable routes, the excessive growth of the TCP congestion window and false link failures cause repeated route changes. This behavior was shown by Saadawi and Xu [12], who quantified the performance of several versions of TCP, in ad hoc networks. They showed that TCP Tahoe, Reno, and New Reno suffer from the "instability problem" due to the excessive growth of the congestion window. They suggested restricting the maximum window size. They also showed that *TCP Vegas* does not suffer from this instability problem, because it uses a more

conservative mechanism with *Round Trip Time (RTT)* estimations to control the size of the congestion window. TCP Vegas does not need packet losses to stop the growth of the congestion window. Since our goal is to study the effects of mobility as opposed to congestion, we used TCP Vegas for our simulations with ns-2. [3], [10].

3 Simulations and Discussion

The simulation scenario consists of 50 mobile nodes, which move in a rectangular area of 300 by 1500 meters according to the *random waypoint model.* The random waypoint model defines the node movement as follows: The start position of each node is assigned randomly. A node remains stationary for a specified period of time called the *pause time.* After the pause time, the node randomly chooses a location and a speed between zero and some *maximum speed* and travels to that location in a straight line. Once the node reaches its new location, it rests there for *pause time* seconds and then randomly chooses a new location and travel speed. With a pause time of zero seconds, all nodes are constantly in motion.

Figure 3 shows the simulation scenario for two setups that we consider. We call these setup I and setup II. In setup I, there is *one* TCP Vegas connection between two static nodes. Setup II has *two* crossing TCP Vegas connections between four static nodes. The static nodes are placed at the edges of the rectangular area. As mentioned earlier, these scenarios are typical when the network is lightly loaded and are appropriate for studying the effects of mobility. When the network is heavily loaded increase in power or proactively searching for new routes might not be appropriate and might increase congestion levels. We re-iterate that our schemes will have to be supplemented by other schemes that can estimate the congestion levels in the network.

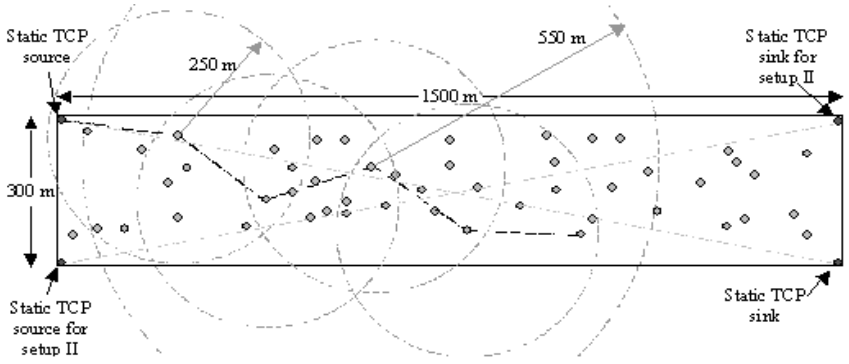

Fig. 3. Simulation Scenario

The traffic over each TCP connection is a file transfer of infinite length, i.e. a TCP source will send TCP data packets for the entire duration of the simulation. The de-

fault transmission range of each node is 250 meters and the interference range is 550 meters. A TCP packet travels an average of about 8 hops to get from a source to the corresponding sink. The pause time is zero seconds and the maximum speed of the mobile nodes is set to 0, 4, 8, 12, 16, and 20 m/s for different simulation runs. The simulation time is 600 seconds. These parameters are typically used and are appropriate to study the effects of our protocols with varying levels of mobility.

Proactive LM notifies the routing protocol when the distance estimation to a neighbor for *current time + 0.1 seconds* becomes greater than the default transmission range. This time seems appropriate since we do not want it to be too long (routes are left unused even when the link is fairly stable). The high power transmission range for Reactive LM is 285 meters. This is a system parameter and should be set depending upon the traffic conditions. We choose a reasonable value in this study (approximately a 10% increase from the original range). Table 1 summarizes the simulation parameters.

We used the following metrics to qualify the performance of TCP:

- *Packet loss:* Ratio of the number of dropped TCP packets to the total number of sent TCP packets.
- *TCP goodput:* Number of TCP data packets received by the *application layer* at the TCP sink. Goodput does not count retransmitted packets.
- The number of TCP retransmission timeouts per delivered packet.

Parameter	**Value**
Simulation time	600 s
Number of mobile nodes	50
Number of static nodes	2 in setup I, 4 in setup II
Default transmission range	250 m
Proactive LM time to link breakage	0.1 s
Reactive LM high power range	285 m
Traffic	
Type	FTP with infinite backlog over TCP
Packet size	1460 bytes
Number of connections	1 in setup I, 2 in setup II
Movement	
Pause time	0 s
Maximum speed	0, 4, 8, 12, 16, and 20 m/s

Table 1. Summary of Simulation Parameters

In the following two subsections, we shall evaluate our protocols by comparing the performance of five versions: (1) The *Original scheme,* which is the unchanged version with the IEEE 802.11 MAC, (2) Including the *Persistent MAC,* (3) Including the

Proactive LM, (4) Including the *Reactive LM,* and (5) A *Combined scheme,* which includes all of the above methods, i.e., the Persistent MAC, the Proactive LM, and the Reactive LM. For the Original scheme, we used the original AODV and for versions (2) to (5), the modified AODV was used. Each point in a graph represents an average of thirty simulation runs with different random movement patterns described earlier.

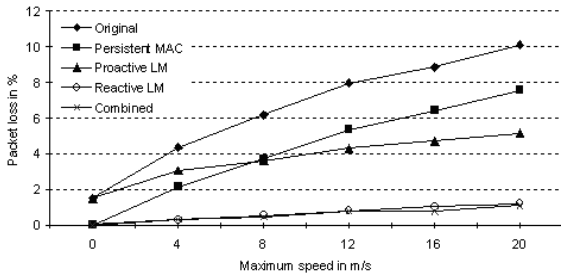

Fig. 4. Performance of our schemes with one TCP session

Figure 4 shows the packet loss as function of maximum speed with set up I. The original scheme drops between 1.5 and 10 percent of the packets. Persistent MAC can significantly reduce the loss in scenarios with low mobility, in which contention-induced link failures dominate. There is no improvement with the Proactive LM in static scenarios as all packet losses are caused by false link failures. When node mobility increases, Proactive LM can approximately reduce the number of losses by half. Reactive LM and Combined MAC can reduce the packet loss to less than 1 percent in all mobility scenarios. The reason why the reactive scheme outperforms the proactive scheme is because, in the proactive scheme, although a route failure is initiated prior to the actual failure, the source does not stop sending in the meantime. Hence, packets in transit are still lost when the link actually fails. In the reactive scheme, when the link fails the packets in transit are salvaged.

Figure 5 shows the effects of decreased packet loss on TCP goodput with set up I. Persistent MAC, Proactive LM, Reactive LM, and the Combined scheme can increase the TCP goodput, especially in high mobility scenarios. The reason for higher goodput is the decreased number of TCP retransmission timeouts. Figure 6 shows the number of TCP retransmission timeouts per delivered data packet. With the Original scheme, TCP times out over 3 times per 100 delivered data packets in high mobility scenarios, whereas with the Combined scheme, there is only about 1 timeout per packet.

Figure 7 compares the Original scheme and the Combined scheme for one and two TCP connections. With two TCP connections, the percentage of dropped packets is higher than with one connection. The reason for this increase in packet losses is increased link layer contention, which leads to a higher percentage of false link failures (Figure 8). At higher levels of congestion, one would notice that the percentage of dropped packets increases. Thus, one would expect that the proposed schemes will be

beneficial only at light loads wherein mobility is primarily responsible for link failures.

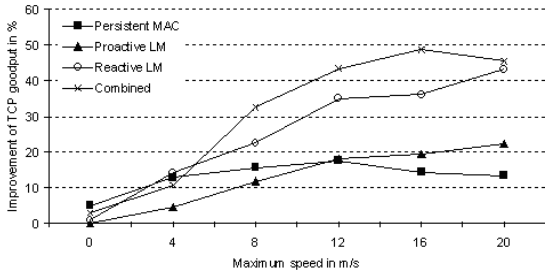

Fig. 5. Improvement of TCP Goodput versus maximum speed for one connection.

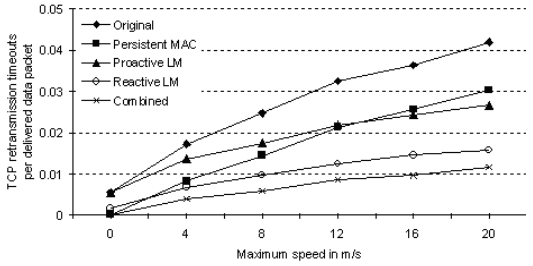

Fig. 6. TCP retransmission time-outs per delivered packet as a function of maximum speed.

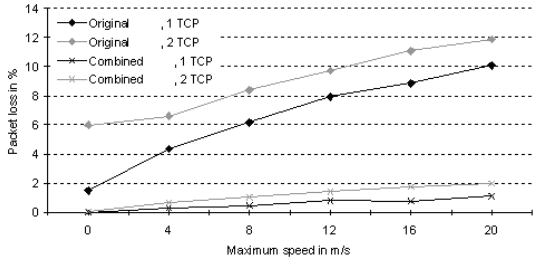

Fig. 7. Comparing the performance in the presence of one and two TCP connections.

Figure 9 shows the goodput improvement the Combined scheme for one and two TCP connections. The total goodput improvement for two TCP connections is lower than for one, except with low node mobility. With increased network contention, it is more difficult for the Proactive and Reactive LM to salvage packets in transit as it takes longer for these packets to traverse the weak or high power link. With low node

mobility, Persistent MAC is more effective in reducing false link failures in the scenario with two connections.

In summary, the reduction of packet loss results in fewer TCP retransmission timeouts and therefore in higher TCP goodput. The higher the packet loss, the greater is the improvement by the proposed mechanisms. In our simulation scenario, the Original scheme drops up to 10 percent of all packets on a long multi-hop route with high node mobility. Combined MAC can improve the TCP goodput by up to 45 percent. These simulation results are for 8-hop TCP connections. With shorter connections, there is less packet loss and therefore less improvement of TCP goodput.

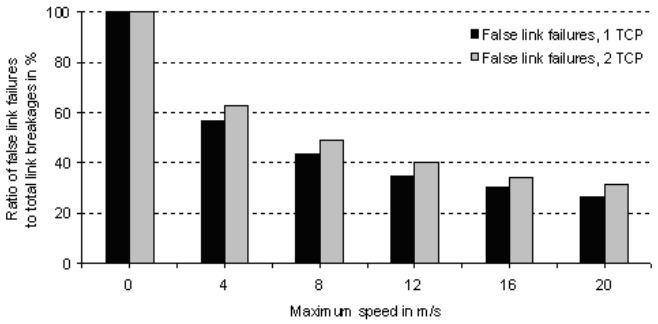

Fig. 8. Fraction of False Link Failures

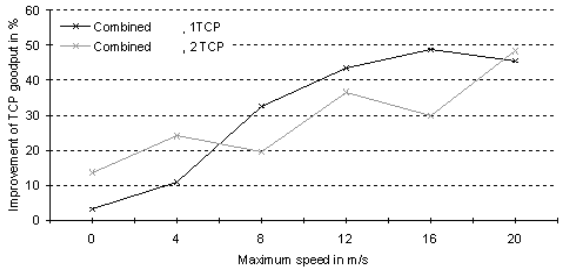

Fig. 9. Improvement in TCP goodput versus maximum speed with one and two TCP connections

5. Conclusions and Future Work

In this paper our objective is to reduce the packet losses due to mobility in ad hoc networks and thereby improve the performance of TCP. We propose a temporary increase in transmit power level when a node moves out of range due to mobility to

temporarily re-establish the failed link. This would enable the TCP packets that are already in flight to traverse the link.

The use of the IEEE 802.11 MAC protocol causes *false link failures* due to congestion. We propose a methodology that allows us to distinguish between true link failures due to mobility and false link failures. The methodology is based on the measurement of signal strength at the physical layer and using it to determine if a node is still within range at the MAC layer. We then increase power levels to temporarily re-establish a failed link only if it is determined to be mobility induced. We also investigate a proactive scheme, in which weak links are identified based on these signal strength measurements and routes are proactively found prior to failure. This in turn helps in switching to the new route even before failure occurs and thus can stem packet losses.

The simulation results with ns-2 showed that these mechanisms could considerably reduce the number of packet losses. Consequently, the number of TCP retransmission timeouts was reduced and the TCP sources sent more packets, thereby increasing the goodput by up to 45 percent.

However, our methods are applicable when there is little congestion in the network since they are primarily geared towards coping with mobility. Additional mechanisms are required to correctly determine the levels of congestion in the network. These schemes are to be used only at light loads. At higher loads, mobility induced failures are likely to be much lower as compared to congestion induced losses. Other mechanisms for coping with such losses are needed.

6. Bibliography

1 S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multi-hop wireless ad hoc networks," *IEEE Communications Magazine*, vol.39, Issue 6., June 2001.

2 T.Goff, *et al*, "Pre-emptive Routing in Ad Hoc Networks*", Proceedings of ACM MOBICOM*, Italy 2001

3 L.S. Brakmo, and L..Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet",. *IEEE Journal On Selected Areas In Communications*, Vol. 13, No. 8, October 1995

4 K. Chandran, K et al, "A Feedback based Scheme for Improving TCP Performance in Ad hoc Wireless Networks", *Proc. ICDCS*, Amsterdam, May 1998.

5 D.Kim, *et al*, "TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks", *Journal Of Communications And Networks*, June 2001.

6 N. Poojary, *et al*, "Medium Access Control in a Network of Ad Hoc Mobile Nodes with Heterogeneous Power Capabilities", *Proceedings of IEEE ICC*, Helsinki 2001.

7 K. Fall and K. Varadhan, *The ns Manual.* 2001.

8 G. Holland and N. Vaidya, “Analysis of TCP Performance over Mobile Ad Hoc Networks*”, Proceedings of ACM MOBICOM* , p. 219-230, Seattle 1999

9 http://grouper.ieee.org/groups/802/11/main.html.

10 Information Sciences Institute (ISI). *The Network Simulator - ns-2.* http://www.isi.edu/nsnam/ns/

11 C.E. Perkins, and E. M. Royer, “Ad-hoc On-Demand Distance Vector Routing”, *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

12 T.Saadawi and S.Xu, “Performance Evaluation of TCP Algorithms in Multi-hop Wireless Packet Networks”, *Journal of Wireless Communications and Mobile Computing* 2002.

13 Kopparty, S. et al, “Split-TCP for Ad Hoc Networks”, *Proceedings of IEEE GLOBECOM* 2002.