

A SURVEY OF TCP OVER AD HOC NETWORKS

AHMAD AL HANBALI, EITAN ALTMAN, AND PHILIPPE NAIN, INRIA SOPHIA ANTIPOLIS FRANCE

ABSTRACT

The Transmission Control Protocol (TCP) was designed to provide reliable end-to-end delivery of data over unreliable networks. In practice, most TCP deployments have been carefully designed in the context of wired networks. Ignoring the properties of wireless ad hoc networks can lead to TCP implementations with poor performance. In order to adapt TCP to the ad hoc environment, improvements have been proposed in the literature to help TCP to differentiate between the different types of losses. Indeed, in mobile or static ad hoc networks losses are not always due to network congestion, as it is mostly the case in wired networks. In this report, we present an overview of this issue and a detailed discussion of the major factors involved. In particular, we show how TCP can be affected by mobility and lower-layer protocols. In addition, we survey the main proposals that are intended to adapting TCP to mobile and static ad hoc environments.

Ad hoc networks are complex distributed systems that consist of wireless mobile or static nodes that can freely and dynamically self-organize. In this way they form arbitrary, and temporary, “ad hoc” network topologies, allowing devices to seamlessly interconnect in areas with no pre-existing infrastructure. Recently, the introduction of new protocols such as Bluetooth [1], IEEE 802.11 [2], and Hyperlan [3] are making possible the deployment of ad hoc networks for commercial purposes. As a result, considerable research efforts have been made in this new challenging wireless environment. For simplicity, in this article we will use the term MANETs instead of mobile ad hoc networks, and SANETs instead of static ad hoc networks. Also we note that the term ad hoc networks will represent both mobile ad hoc networks (MANETs) and static ad hoc networks (SANETs).

TCP (Transmission Control Protocol) [4] was designed to provide reliable end-to-end delivery of data over unreliable networks. In theory, TCP should be independent of the technology of the underlying infrastructure. In particular, TCP should not care whether the Internet Protocol (IP) is running over wired or wireless connections. In practice, it does matter because most TCP deployments have been carefully designed based on assumptions that are specific to wired networks. Ignoring the properties of wireless transmission can lead to TCP implementations with poor performance.

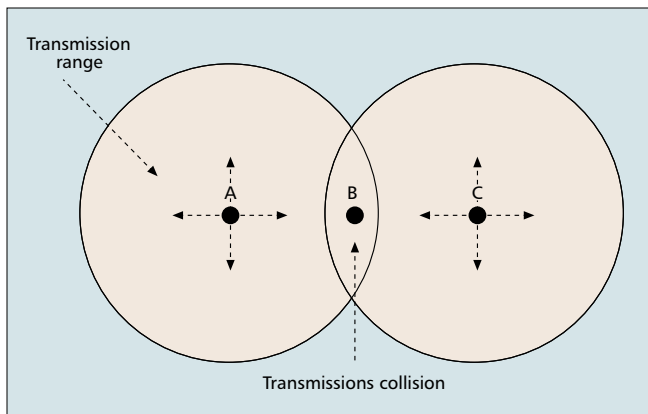
In ad hoc networks, the principal problem of TCP lies in performing congestion control in case of losses that are not induced by network congestion. Since bit error rates are very low in wired networks, nearly all TCP versions nowadays assume that packet losses are due to congestion. Consequently, when a packet is detected to be lost, either by timeout or

by multiple duplicated ACKs, TCP slows down the sending rate by adjusting its congestion window. Unfortunately, wireless networks suffer from several types of losses that are not related to congestion, making TCP not adapted to this environment. Numerous enhancements and optimizations have been proposed over the last few years to improve TCP performance over one-hop wireless (not necessarily ad hoc) networks. These improvements include infrastructure-based WLANs [5–8], mobile cellular networking environments [9, 10], and satellite networks [11, 12]. Ad hoc networks inherit several features of these networks, in particular high bit error rates and path asymmetry, and add new problems caused by mobility and multi-hop communications, such as network partitions, route failures, and hidden (or exposed) terminals. We note that the following TCP versions — Tahoe, Reno, Newreno, and Vegas — perform differently in ad hoc networks [13]. However, all these versions suffer from the same problem: the inability to distinguish between packet losses due to congestion and losses due to the specific features of ad hoc networks. For more details about TCP versions see the Appendix; for a survey of TCP versions we refer to [14].

By examining TCPs performance studies over MANETs and SANETs, we identified the following four major problems:

- TCP is unable to distinguish between losses due to route failures and losses due to network congestion.
- TCP suffers from frequent route failures.
- The contention on the wireless channel.
- TCP unfairness.

We note that the first two problems are the main causes of TCP performance degradation in MANETs; the other two



■ **Figure 1.** *Hidden terminal problem: packets sent to B by A and C will collide at B.*

problems are the main causes of TCP performance degradation in SANETs. Based on these four problems, the proposals that aim to improve TCP performance over ad hoc networks are regrouped in four sets. We classify the proposals of a set into two categories: cross layer proposals and layered proposals. In cross layer proposals, TCP and its underlying protocols work jointly. For example, considerable improvements are possible when TCP can differentiate between packet losses due to congestion, which should activate the congestion control algorithm, and losses due to the specific features of MANETs. In order to do that, some proposals suggest that when the routing layer detects a route failure, it should notify the TCP sender about a routing failure [15–19]. Upon receiving this notification, the TCP sender enters a *freezing* state. In this state TCP stops sending data packets, and it freezes all its variables to their current value, such as the congestion window and the retransmission timer. After route re-establishment, the TCP sender goes back to the normal state. In layered proposals, the problems of TCP are addressed at one of the OSI layers. For example, in [20] Altman and Jimenez use adaptive TCP delayed ACK to reduce the contention on the wireless channel in SANETs. In [21] Fu *et al.* propose two link layer techniques, called Link RED and adaptive pacing, to improve TCP performance over SANETs.

The rest of this article is organized as follows. We first discuss TCP’s challenges in ad hoc networks. We then survey papers on TCP performance over mobile and static ad hoc networks. We review the main proposals for improving TCP performance, and we compare and discuss these proposals. We then conclude the article and propose research directions.

TCP’S CHALLENGES IN AD HOC NETWORKS

The performance of TCP degrades in ad hoc networks, because TCP has to face new challenges due to several reasons specific to these networks: lossy channels, hidden and exposed stations, path asymmetry, network partitions, route failures, and power constraints.

LOSSY CHANNELS

The main causes of errors in wireless channels are the following:

- **Signal attenuation:** This is due to a decrease in the intensity of the electromagnetic energy at the receiver (e.g. due to long distance), which leads to low signal-to-noise ratio (SNR).
- **Doppler shift:** This is due to the relative velocities of the transmitter and the receiver. Doppler shift causes fre-

quency shifts in the arriving signal, thereby complicating the successful reception of the signal.

- **Multipath fading:** Electromagnetic waves reflecting off objects or diffracting around objects can result in the signal traveling over multiple paths from the transmitter to the receiver. Multipath propagation can lead to fluctuations in the amplitude, phase, and geographical angle of the signal received at a receiver.

In order to increase the success of transmissions, link layer protocols implement Automatic Repeat reQuest (ARQ) or Forward Error Correction (FEC), or both. For example, IEEE 802.11 implements ARQ, so when a transmitter detects an error, it will retransmit the frame; error detection is timer-based. Bluetooth implements both ARQ and FEC on some synchronous and asynchronous connections.

Note that packets transmitted over a fading channel may cause the routing protocol to incorrectly conclude that there is a new one-hop neighbor. This one-hop neighbor could provide a shorter route to even more distant nodes. Unfortunately, this new shorter route is usually unreliable. As an example, in [22] Chin *et al.* deploy DSDV [23] and AODV [24] routing protocols in a real network. They find that neither of these protocols can provide a stable multi-hop route because of the physical channel behavior, especially fading.

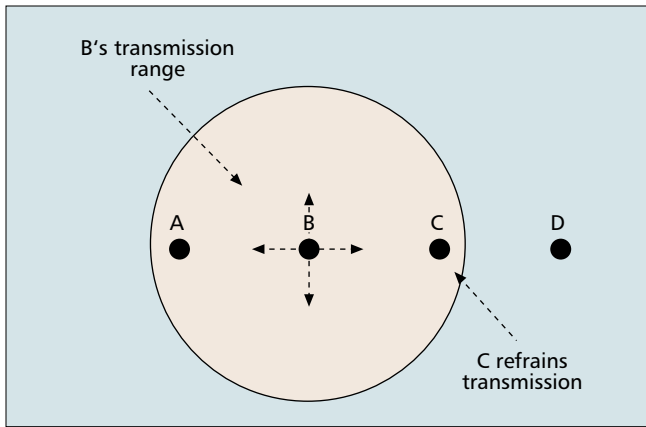
HIDDEN AND EXPOSED STATIONS

In ad hoc networks, stations may rely on physical carrier-sensing mechanisms to determine an idle channel, such as in the IEEE 802.11 DCF function. This sensing mechanism does not solve completely the *hidden station* and the *exposed station* problems [25]. Before explaining these problems, we need to clarify the term “transmission range.” The transmission range is the range, with respect to the transmitting station, within which a transmitted packet can be successfully received.

A typical hidden terminal situation is depicted in Fig. 1. Stations A and C have a frame to transmit to station B. Station A cannot detect C’s transmission because it is outside the transmission range of C. Station C (resp. A) is therefore “hidden” to station A (resp. C). Since the transmission areas of A and C are not disjoint, there will be packet collisions at B. These collisions make the transmission from A and C toward B problematic. To alleviate the hidden station problem, virtual carrier sensing has been introduced [2, 26]. It is based on a two-way handshaking that precedes data transmission. Specifically, the source station transmits a short control frame, called Request-To-Send (RTS), to the destination station. Upon receiving the RTS frame, the destination station replies by a Clear-To-Send (CTS) frame, indicating that it is ready to receive the data frame. Both RTS and CTS frames contain the total duration of the data transmission. All stations receiving either RTS or CTS will keep silent during the data transmission period (e.g. station C in Fig. 1).

However, as pointed out in [21, 27], the hidden station problem may persist in IEEE 802.11 ad hoc networks even with the use of the RTS/CTS handshake, because the power needed to interrupt a packet reception is much lower than that required to deliver a packet successfully. In other words, a node’s transmission range is smaller than the sensing node range. For more details see the model of the physical layer implemented in the NS-2 and Glomosim simulators [28, 29].

The exposed station problem results from a situation where a transmission has to be delayed because of the transmission between two other stations within the sender’s transmission range. In Fig. 2 we show a typical scenario where the exposed terminal problem occurs. Let us assume that A and C are within B’s transmission range, and A is outside C’s transmission range. Let us also assume that B is transmitting to A, and C has a frame to be transmitted to D. According to the carrier sense mechanism, C senses a busy channel because of



■ **Figure 2.** Exposed terminal problem: Because of B's transmission, C refrains transmission to D.

B's transmission. Therefore, station C will refrain from transmitting to D, although this transmission would not cause interference at A. The exposed station problem may thus result in a reduction of channel utilization.

It is worth noting that hidden terminal and exposed terminal problems are correlated with the transmission range. By increasing the transmission range, the hidden terminal problem occurs less frequently. On the other hand, the exposed terminal problem becomes more important as the transmission range identifies the area affected by a single transmission.

PATH ASYMMETRY

Path asymmetry in ad hoc networks may appear in several forms as bandwidth asymmetry, loss rate asymmetry, and route asymmetry.

Bandwidth Asymmetry — Satellite networks suffer from high bandwidth asymmetry, resulting from various engineering tradeoffs (such as power, mass, and volume), as well as the fact that for space scientific missions, most of the data originates at the satellite and flows to the earth. The return link is not used, in general, for data transferring. For example, in broadcast satellite networks the ratio of the bandwidth of the satellite-earth link over the bandwidth of the earth-satellite link is about 1000 [11]. On the other hand, in ad hoc networks, the degree of bandwidth asymmetry is not very high. For example, the bandwidth ratio lies between 2 and 54 in ad hoc networks that implement the IEEE 802.11 version g protocol [2]. The asymmetry results from the use of different transmission rates. Because of this different transmission rates, even symmetric source destination paths may suffer from bandwidth asymmetry.

Loss Rate Asymmetry — This type of asymmetry takes place when the backward path is significantly more lossy than the forward path. In ad hoc networks this asymmetry occurs because packet losses depend on local constraints that can vary from place to place. Note that loss rate asymmetry may produce bandwidth asymmetry. For example, in multi-rate IEEE 802.11 protocol versions, senders may use the Auto-Rate-Fallback (ARF) algorithm for transmission rate selection [30]. With ARF, senders attempt to use higher transmission rates after consecutive transmission successes, and revert to lower rates after failures. So, as the loss rate increases the sender will keep using lower transmission rates.

Route Asymmetry — Unlike the previous two forms of asymmetry, where the forward path and the backward path can be

the same, route asymmetry implies that distinct paths are used for TCP data and TCP ACKs. This asymmetry may be an artifact of the routing protocol used. Route asymmetry increases routing overheads and packet losses in the case of a high degree of mobility,¹ because when nodes move, using a distinct forward and reverse route increases the probability of route failures experienced by TCP connections. However, this is not the case with static networks or networks that have a low degree of mobility, as in the case of a network with routes of high lifetime compared to the session transfer time. So it is up to the routing protocols to select symmetric paths when such routes are available in the case of ad hoc networks of high mobility.

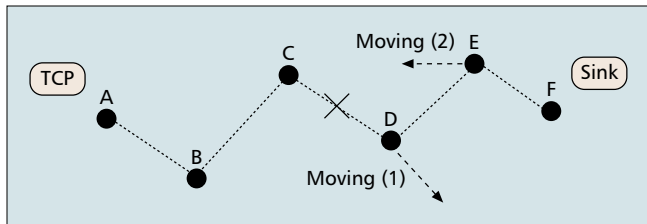
In the context of satellite networks, there has been much research on how to improve TCP performance. However, since satellite networks are out of the scope of this article, we will limit ourselves to list three techniques introduced by these proposals, which we believe might be useful in ad hoc networks.

The first technique is “TCP header compression,” which reduces the size of the TCP ACKs on the backward path [31]. The second technique is “ACK filtering,” which reduces the number of TCP ACKs transmitted by taking advantage of the fact that TCP ACKs are cumulative [32]. The third technique is “ACK congestion control,” which lets the receiver also control the congestion on the backward path. This is done by dynamically maintaining a delayed-ACK factor d by the receiver, and by sending one ACK for every d data packets received [32]. The difference between ACK filtering and ACK congestion control is that the first approach is a link-layer technique that can be implemented at intermediate nodes, while the second approach is a TCP-layer technique that is implemented at the TCP sink. Unfortunately, these techniques alone cause problems such as increasing the sender's burst traffic and also slowing the sender's congestion window growth. Thus it is necessary to adapt the sender congestion control algorithm to avoid these problems. For details about the sender adaptation techniques, we refer to [32]. The adaptive delayed-ACK proposed in [20] aims to reduce the contention on the channel by reducing the number of TCP ACKs transmitted. This proposal also alleviates the asymmetry problem in SANETs. We have not found any other proposal dealing with the asymmetry problem in ad hoc networks.

NETWORK PARTITION

An ad hoc network can be represented by a simple graph G . Mobile stations are the “vertices.” A successful transmission between two stations is an undirected “edge.” Network partition happens when G is disconnected. The main reason for this disconnection in MANETs is node mobility. Another factor that can lead to network partition is energy constrained operation of nodes. An example of network partition is illustrated in Fig. 3. In this figure dashed lines are the links between nodes. When node D moves away from node C this results in a partition of the network into two separate components. Clearly, the TCP agent of node A cannot receive the TCP ACK transmitted by node F. If the disconnectivity persists for a duration greater than the retransmission timeout (RTO) of node A, the TCP agent will trigger the *exponential backoff* algorithm [33], which consists of doubling the RTO whenever the timeout expires. Originally, TCP does not have an indication about the exact time of network reconnection. This lack of indication may lead to long idle periods during which the network is connected again, but TCP is still in the backoff state.

¹ A network with routes of short lifetime compared to the session transfer time is an example of a network with a high degree of mobility.



■ **Figure 3.** Network partition scenario: when *D* is moving away from *C*. The network is reconnected when *E* is moving toward *C*.

ROUTING FAILURES

In wired networks route failures occur very rarely. In MANETs they are frequent events. The main cause of route failures is node mobility. Another factor that can lead to route failures are the link failures caused by the contention on the wireless channel, which is the main cause of TCP performance degradation in SANETs. The route reestablishment duration after route failure in ad hoc networks depends on the underlying routing protocol, mobility pattern of mobile nodes, and traffic characteristics. As already discussed, if the TCP sender does not have indications on the route re-establishment event, the throughput and session delay will degrade because of the large idle time. Also, if the new route established is longer or shorter in term of hops, than the old route TCP will face a brutal fluctuation in round trip time (RTT). In addition, in ad hoc networks, routing protocols that rely on broadcast Hello messages to detect neighbors' reachability may suffer from the "communication gray zones" problem. In these zones data messages cannot be exchanged, although broadcast Hello messages and control frames indicate that neighbors are reachable. So on sending a data message, routing protocols will experience routing failures. In [34] Lundgren *et al.* have conducted experiments and have subsequently concluded that the origin of this problem is heterogeneous transmission rates, the absence of an acknowledgment for broadcast packets, small packet size of Hello messages, and fluctuations of wireless links.

POWER CONSTRAINTS

Because batteries carried by each mobile node have limited power supply, processing power is limited. This is a major issue in ad hoc networks, as each node is acting as an end system and as a router at the same time, with the implication that additional energy is required to forward and relay packets. TCP must use this scarce power resource in an "efficient" manner. Here, efficiency means minimizing the number of unnecessary retransmissions at the transport layer as well as at the link layer.² In general, in ad hoc networks there are two correlated power problems: the first problem is "power saving," which aims at reducing power consumption; the second problem is "power control," which aims at adjusting the transmission power of mobile nodes. Power saving strategies have been investigated at several levels of a mobile device, including the physical-layer transmissions, the operation systems, and the applications [35]. Power control can be jointly used with routing or transport agents to improve the performance of ad hoc networks [36, 37]. Power constraints on communications also reveal the problem of cooperation between nodes, as nodes may not participate in routing and forwarding procedures in order to save battery power.

² The IEEE 802.11 protocol implements a local retransmission at the link layer upon detecting a transmission error.

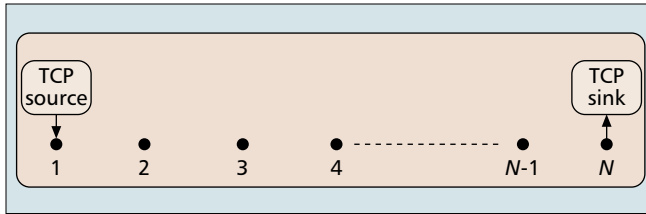
TCP PERFORMANCE OVER AD HOC NETWORKS

In this section we first report on the studies of TCP performance over MANETs, and then we report on the studies of TCP performance over SANETs. We conclude with a summary of the major problems of TCP over static and mobile ad hoc networks. In passing, we point out that routing protocols and link-layer protocols in ad hoc networks are beyond the scope of this article. Interested readers are referred to [38, 39] for details.

TCP PERFORMANCE OVER MANETS

In [16] Monks *et al.* investigate the impact of mobility on TCP throughput in MANETs. In their simulation scenarios, nodes move according to the random way-point model with 0 s pause time. The speed of the node was uniformly distributed in $[0.9v - 1.1v]$ for some mean speed v . At the routing layer the authors use DSR [40]. They report that when the mean speed increases from 2m/s to 10m/s the throughput drops sharply, but when it increases from 10m/s to 30m/s the throughput drops slightly. They also remark that, for a given mean speed, certain mobility patterns achieve throughput close to 0, although the other mobility patterns are able to achieve high throughput. By analyzing the simulations trace of patterns of low throughput, they found that the TCP sender's routing protocol is unable to quickly recognize and purge stale routes from its cache, which results in repeated routing failures and TCP retransmission timeouts. For patterns of high throughput they found that most of time the TCP sender and receiver are close to each other. By examining the mobility patterns, the authors observe that as the sender and receiver move closer to each other, DSR can maintain a valid route by shortening the existing route before a routing failure occurs. However, as the sender and receiver move away from each other, DSR waits until a failure occurs to lengthen a route. The route failure induces up to a TCP-window's worth of packet losses [41], and subsequent route discovery latency often results in repeated TCP timeout. To prevent TCP invocation of congestion control that deteriorates TCP throughput in case of losses induced by mobility, the authors suggest using the explicit link failure notification (ELFN) technique. An overview of this technique is discussed later.

In [41], in addition to the problem highlighted in [16], i.e. "TCP treats losses induced by route failures as signs of network congestion," Anantharaman *et al.* identify a set of factors that also contribute to the degradation of TCP throughput in the presence of mobility. These factors are MAC failure detection and route computation latencies. The MAC failure detection latency is defined as the amount of time spent before the MAC concludes a link failure. They found that in the case of the IEEE 802.11 protocol, when the load is light (one TCP connection) this latency is small and independent of the speed of the nodes. However, in the case of high load, the value of this latency is magnified and becomes a function of the nodes' speed. The route computation latency is defined as the time taken to recompute the route after a link failure. They found that, as for MAC failure detection latency, the route computation latency increases with the load and becomes a function of the nodes' speed in the high-load case. Also, the authors identify another problem, called MAC packet arrival, that is related to routing protocols. In fact, when a link failure is detected, the link failure is sent to the routing agent of the packet that triggered the detection. If other sources are using the same link in the path to their destinations, the node that detects the link failure has to wait until it receives a packet from these sources before they are informed of the link failure. This also contributes to the delay after which a source realizes that a path is broken.



■ **Figure 4.** Multi-hop chain topology.

In [42] Dyer and Boppana report simulation results on the performance of TCP Reno over three different routing protocols (AODV [24], DSR [40], and ADV [43]). It is found that ADV performs well under a variety of mobility patterns and topologies. Furthermore, they propose a heuristic technique called fixed RTO to improve the performance of on-demand routing protocols (AODV and DSR). An overview of this technique is discussed later. In [44] Lim *et al.* show that TCP's performance degrades when the multipath routing protocol SMR [45] is used. Multipath routing effects TCP by two factors: the first factor is the inaccuracy of the average RTT measurement, which leads to more premature timeouts; the second factor is out-of-order packet delivery via different paths, which triggers duplicated ACK, which in turn triggers TCP congestion control.

TCP PERFORMANCE OVER SANETS

In [20, 21, 46, 47] the authors report simulation results on TCP throughput in a static linear multi-hop chain, where the IEEE 802.11 protocol is used. In Fig. 4 we display a multi-hop chain of N nodes. It is expected that as the number of hops increases, the spatial reuse will also increase. However, simulation results indicate that TCP throughput decreases "rapidly" up to a point as the number of hops increases. It is argued that this decrease is due to the hidden terminals problem, which increases frame collisions. After a repeated³ transmission failure the MAC layer will react with two actions: first, the MAC will drop the head-of-line frame destined for the next hop (we note this type of drop is known also as a drop due to contention on a wireless channel); second, the MAC will notify the upper layer about a link failure. When the routing protocol of a source node detects a routing failure, it will initiate a route re-establishment process. In general the route re-establishment duration is greater than the retransmission timer of the TCP agent; hence, the TCP agent will enter the backoff procedure and will set its congestion window to 1. Also, as the TCP sender does not have indications on the route re-establishment event, TCP will suffer from a long idle time. During this time the network may be connected again, but TCP is still in the backoff state.

In [13] Xu and Saadawi study the performance of TCP Tahoe, Reno, New Reno, Sack, and Vegas⁴ over the multi-hop chain topology shown in Fig. 4, in the case where the IEEE 802.11 protocol is used. It is shown that TCP Vegas delivers the better performance and does not suffer from instability. By tuning the sender TCP's maximum window size (advertised window) to approximately four packets, all TCP versions perform similarly. Furthermore, the authors investigate the performance of these TCP versions using the delayed-ACK option, as defined in RFC 1122. According to the mentioned RFC, the TCP sink will send one TCP

ACK packet for every two TCP packets received. This option will reduce the contention on the wireless channel, because the data and Ack packets share the same wireless channel. Simulating the multihop chain with the delayed-ACK option, they report an improvement of 15 percent to 32 percent.

In [46, 48] the authors study how TCP connections share the bandwidth of the channel in the context of an ad hoc network. They report some unfair bandwidth sharing using the actual MAC 802.11 in a multi-hop communication environment. Moreover, in [49] Xu *et al.* show that also in scenarios where TCP crosses wireless ad hoc and wired networks, the TCP unfairness problem persists. In [50] Xu *et al.* report that RED [51] did not solve TCP's unfairness in ad hoc networks because congestion does not happen in a single node, but rather in an entire area involving multiple nodes. Thus, the local packet queue at any single node cannot completely reflect the network congestion state. For this reason they define a new distributed queue that contains all packets whose transmissions will affect the node transmission in addition to its own packets. An overview of this proposal is given later. In [52] Anastasi *et al.* investigate the performance of an IEEE 802.11 ad hoc network by means of an experimental study. Their findings match those obtained by simulations, namely, TCP connections may experience significant unfairness. They mention several aspects that are usually neglected in simulation studies of the IEEE 802.11b protocol. For instance, since the control frames (RTS, CTS, ACK) and data frame may be transmitted at different rates, this produces different transmission ranges and carrier sensing ranges in the network.

In [53] Chen *et al.* study the impact of TCP's congestion window limit (CWL) on TCP throughput in SANETS. In fact, they relate the problem of setting TCP's optimal CWL to identifying the bandwidth-delay product (BDP) of multi-hop paths, as they argue that TCP's congestion window should be less than the BDP. They prove that regardless of the MAC layer being used, the value of the BDP in bytes at multi-hop routes cannot exceed the value of the round-trip hop-count (RTHC) times the size of data packets in bytes of these multi-hop routes. This is done by assuming similar bottleneck bandwidths along the forward and reverse route. In the case of the IEEE 802.11 MAC layer protocol, when a node is transmitting, all other nodes that are inside its interference range or that receive the RTS or CTS frames will remain silent until the transmission ends. The limit on CWL is illustrated in the scenario in Fig. 4: the distance between adjacent nodes is equal to the node transmission range, which is taken to be $250m$, and the node interference range is equal to $500m$. For that scenario, the authors report that the BDP is less than one fourth of the RTHC, as transmission of data packets can be done concurrently without collisions only 4-hops nodes away. Using a simulation of the previous scenario, they found that when the CWL exceeds one fifth of the RTHC, TCP throughput decreases substantially. Thus, based on this tighter bound the authors propose an algorithm to adjust TCP CWL according to the RTHC of the routes used. Using their algorithm they report an improvement in TCP performance of up to 16 percent even in mobile scenarios that contain multiple TCP connections. In [21] Fu *et al.* report that given a specific network topology and flow patterns, there exists an optimal TCP window size W^* . Using W^* , TCP achieves the best throughput via improved spatial reuse. But unfortunately, in practice TCP operates at an average window size that is much larger than W^* . This leads to increased packet loss due to the contention on the wireless channel. To help TCP operate at around W^* , they propose two techniques: link RED and adaptive pacing at the link layer. By coupling these two techniques they show a 30 percent improvement in performance. An overview of these techniques is given later.

³ On repeated transmission failure, 802.11 MAC is allowed to retransmit short frames seven times and long frames four times.

⁴ For details about TCP versions see Appendix I.

We conclude that two major problems cause TCP performance degradation in MANETs: TCP is unable to distinguish between losses due to route failures and losses caused by network congestion, and TCP suffers from frequent route failures. In SANETs, TCP faces two other major problems: the contention on wireless channel, and TCP unfairness. Based on these four problems, the TCP proposals in the next section will be grouped into four sets.

PROPOSALS TO IMPROVE TCP PERFORMANCE IN AD HOC NETWORKS

In this section we present the various proposals that have been made in the literature to improve the performance of TCP in ad hoc networks. We group these proposals into four sets according to the four problems identified previously. These four problems are:

- TCP is unable to distinguish between losses due to route failures and those due to network congestion.
- Frequent route failures.
- Contention on the wireless channel.
- TCP unfairness.

We note that the first two problems are the main causes of TCP performance degradation in MANETs. However, the last two problems are the main causes of TCP performance degradation in SANETs. Figure 5 shows a general classification of each set of proposals. We classify the proposals that belong to the same set to two types: cross layer proposals and layered proposals. The cross layer proposals rely on interactions between two layers of the Open System Interconnection (OSI) architecture. These proposals were motivated by the fact that “providing lower-layer information to the upper layer should help the upper layer perform better.” Thus, depending on which two OSI layers will have information exchanged between them, cross layer proposals can be further classified into four types: TCP and network, TCP and link, TCP and physical, and network and physical. Layered proposals rely on adapting OSI layers independently from other layers. Thus, depending on which layer is involved, layered proposals can be further classified into three types: TCP layer, network layer, and link layer proposals.

In general, cross layer solutions report better performance than layered solutions. However, layered solutions respect the concept of designing protocols in isolation, so they are considered to be long term solutions. Thus, to choose between cross layer and layered solutions we must first decide what is a priority for us: performance optimization or architecture. Performance optimization can lead to short term gain, while architecture is usually based on longer term considerations. In addition, cross layer solutions are more complex to implement and design than layered solutions, because the implementation of cross layer solutions requires at least modifications of two OSI layers, and their design requires that the system be considered in its entirety. For more discussions about cross layer design in ad hoc networks we refer the reader to [54].

PROPOSALS TO DISTINGUISH BETWEEN LOSSES DUE TO ROUTE FAILURES AND CONGESTION

The proposals that address the problem of TCP’s inability to distinguish between losses due to route failures and losses due to network congestion in MANETs can fall into two categories: TCP and network cross layer proposals, and TCP layer proposals.

TCP and Network Cross Layer Proposals

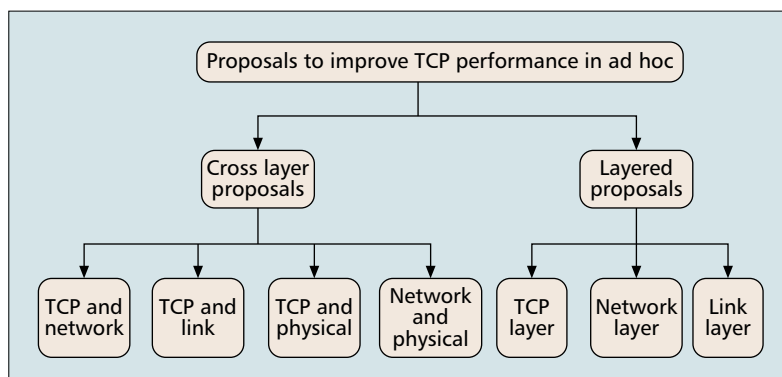
TCP-F — TCP Feedback [55] is a feedback-based approach to handle route failures in MANETs. This approach allows the TCP sender to distinguish between losses due to route failure and losses due to network congestion. This is done as follows. When the routing agent of a node detects the disruption of a route, it explicitly sends a Route Failure Notification (RFN) packet to the source. On receiving the RFN, the source goes into a snooze state. The TCP sender in snooze state will stop sending packets and will freeze all its variables such as timers and congestion window size. The TCP sender remains in this snooze state until it is notified of the restoration of the route through a Route Re-establishment Notification (RRN) packet. Upon receiving the RRN, the TCP sender will leave the snooze state and will resume transmission based on the previous sender window and timeout values. To avoid blocking scenario in the snooze state, the TCP sender, upon receiving the RFN, triggers a route failure timer. When this timer expires the congestion control algorithm is invoked normally.

The authors report an improvement by using TCP-F over TCP. The simulation scenario is basic and is not based on an ad hoc network. Instead, they emulate the behavior of an ad hoc network from the viewpoint of a transport layer.

ELFN-based Technique — The Explicit Link Failure Notification technique [16] is similar to TCP-F. However, in contrast to TCP-F, the evaluation of the proposal is based on a real interaction between TCP and the routing protocol. This interaction aims to inform the TCP agent about route failures when they occur. The authors use an ELFN message, which is piggybacked onto the route failure message sent by the routing protocol to the sender. The ELFN message is like a “host unreachable” Internet Control Message Protocol (ICMP) message, which contains the sender receiver addresses and ports, as well as the TCP packet’s sequence number. On receiving the ELFN message, the source responds by disabling its retransmission timers and enters a “standby” mode. During the standby period the TCP sender probes the network to check if the route is restored. If the acknowledgment of the probe packet is received, the TCP sender leaves the standby mode, resumes its retransmission timers, and continues the normal operations.

In the mentioned reference, the authors study the effect of varying the time interval between probe packets. They also evaluate the impact of the RTO and the Congestion Window (CW) on the restoration of the route. They found that a probe interval of 2 sec. performs the best, and they suggest making this interval a function of the RTT instead of giving it a fixed value. For the RTO and CW values upon route restoration, they found that using the prior values before route failure performs better than initializing CW to 1 packet and/or RTO to 6 sec., the latter value being the initial default value of RTO in TCP Reno and New Reno versions.

This technique provides significant enhancements over



■ **Figure 5.** Classification of proposals to improve TCP performance in ad hoc networks.

standard TCP, but further evaluations are still needed. For instance, different routing protocols should be considered other than the reactive protocol DSR considered in [16], especially proactive protocols such as OLSR [56]. In [41] Anantharaman *et al.* report that in the case of high load⁵ ELFN performs worse than standard TCP because ELFN is based on probing the network periodically to detect route re-establishment. In addition, in [57] Monks *et al.* find that even in the case of light load ELFN performs worse than standard TCP by 5 percent in the case of static ad hoc networks.

ATCP — Ad hoc TCP [17] also utilizes network layer feedback. In addition to the route failures, ATCP tries to deal with the problem of high Bit Error Rate (BER). The TCP sender can be put into persist state, congestion control state, or retransmit state. A layer called ATCP is inserted between the TCP and IP layers of the TCP source nodes. ATCP listens to the network state information provided by ECN (Explicit Congestion Notification) messages [58] and by ICMP “Destination Unreachable” messages; then ATCP puts the TCP agent into the appropriate state. On receiving a “Destination Unreachable” message, the TCP agent enters a persist state. During this state the TCP agent is frozen and no packets are sent until a new route is found by probing the network. The ECN is used as a mechanism to explicitly notify the sender about network congestion along the route being used. Upon reception of the ECN, TCP congestion control is invoked normally without waiting for a timeout event. To detect packet losses due to channel errors, ATCP monitors the received ACKs. When ATCP sees that three duplicate ACKs have been received, it does not forward the third duplicate ACK but puts TCP in the persist state and quickly retransmits the lost packet from TCP’s buffer. After receiving the next ACK, ATCP will resume TCP to the normal state. Note that ATCP allows interoperability with TCP sources or destinations that do not implement ATCP.

ATCP was implemented in a testbed and evaluated under different scenarios, such as congestion, lossy links, partition, and packet reordering. In all cases the transfer time of a given file using ATCP yielded better performance than TCP. However, the used scenario was somewhat special, since neither wireless links nor ad hoc routing protocols were considered. In fact, the authors used an experimental testbed consisting of five PCs equipped with Ethernet cards. With these PCs, the authors formed a four-hop network.

In addition to route failure, ATCP tries to deal with the problem of high BER, network congestion, and packet reorder. This advantage makes ATCP a more robust proposal for TCP in MANETs. However, certain assumptions, such as an ECN-capable node as well as the sender node being always reachable, might somehow be hard to meet in a mobile ad hoc context. Also, the probing mechanism used to detect route re-establishment generates problems in the case of high load, as in the ELFN proposal.

TCP-BuS — TCP Buffering capability and Sequence information [19], as in previous proposals, uses network feedback to detect route failure events and to take convenient action in response to these events. The novel scheme in this proposal is the introduction of *buffering capability* in mobile nodes. The authors select the source-initiated on-demand ABR [59] (Associativity-Based Routing) routing protocol. The following enhancements are proposed.

Explicit Notification: Two control messages are used to notify the source about the route failure and the route re-establishment. These messages are called Explicit Route Disconnection Notification (ERDN) and Explicit Route

Successful Notification (ERSN). On receiving the ERDN from the node that detected the route failure, called the Pivoting Node (PN), the source stops sending. Similarly, after route re-establishment by the PN using a Localized Query (LQ), the PN will transmit the ERSN to the source. On receiving the ERSN, the source resumes data transmission.

Extending Timeout Values: During the Route ReConstruction (RRC) phase, packets along the path from the source to the PN are buffered. To avoid timeout events during the RRC phase, the retransmission timer value for buffered packets is doubled.

Selective Retransmission Request: As the retransmission timer value is doubled, the lost packets along the path from the source to the PN are not retransmitted until the adjusted retransmission timer expires. To overcome this, an indication is made to the source so that it can retransmit these lost packets selectively.

Avoiding Unnecessary Requests for Fast Retransmission: When the route is restored, the destination notifies the source about the lost packets along the path from the PN to the destination. On receiving this notification, the source simply retransmits these lost packets. However, the packets buffered along the path from the source to the PN may arrive at the destination earlier than the retransmitted packets. In this case the destination will reply by duplicate ACK. These unnecessary request packets for fast retransmission are avoided.

Reliable Retransmission of the Control Message: In order to guarantee the correctness of TCP-BuS operation, the authors propose to transmit reliably the routing control messages ERDN and ERSN. The reliable transmission is done by overhearing the channel after transmitting the control messages. If a node has sent a control message but did not overhear this message relayed during a timeout, it will conclude that the control message is lost and it will retransmit this message.

This proposal introduces many new techniques for the improvement of TCP. The novel contributions of this paper are the buffering techniques and the reliable transmission of control messages. In their evaluation the authors found that TCP-BuS outperforms the standard TCP and TCP-F under different conditions. The evaluation is based only on the ABR routing protocol, and other routing protocols should be taken into account. Also, TCP-BuS did not take into account that the pivoting node may fail to establish a new partial route to the destination. In this case, what will happen to the packets buffered at intermediate nodes is not handled by the authors.

TCP Layer Proposals

Fixed RTO — This technique [42] is a sender-based technique that does not rely on feedback from the network. In fact, the authors employ a heuristic to distinguish between route failures and congestion. When two timeouts expire in sequence, which corresponds to the situation in which the missing ACK is not received before the second RTO expires, the sender concludes that a route failure event has occurred. The unacknowledged packet is retransmitted but the RTO is not doubled a second time. This is in contrast to standard TCP, in which an “exponential” backoff algorithm is used. The RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged.

In [42] Dyer *et al.* evaluate this proposal by considering different routing protocols as well as the TCP selective and the delayed acknowledgment options. They report that significant enhancements are achieved when using fixed-RTO with on-demand routing protocols. Nevertheless, as stated by the authors themselves, this proposal is restricted to wireless networks only, a serious limitation since interoperation with wired networks is clearly necessary. Also, the supposition that two consecutive timeouts are the exclusive results of route failures need more analysis, especially in cases of congestion.

⁵ 25 TCP connections.

TCP Door — TCP Detection of Out-of-Order and Response (DOOR) is an end-to-end approach [18]. This approach, which does not require the cooperation of intermediate nodes, is based on out-of-order (OOO) delivery events. OOO events are interpreted as an indication of route failure. The detection of OOO events is accomplished either by means of a sender-based or a receiver-based mechanism. The sender-based mechanism uses the non-decreasing property of the ACKs sequence numbers to detect the OOO events. In the case of duplicate ACK packets, these ACKs will have the same sequence number, so that the sender needs additional information to detect an OOO event. This information, called an ACK Duplication Sequence Number (ADSN), is a one-byte option added to ACKs. The ADSN is incremented and transmitted with each duplicate ACK. However, the receiver-based mechanism needs an additional two-byte TCP option, called the TCP Packet Sequence Number (TPSN), to detect OOO events. The TPSN is incremented and transmitted with each TCP packet, including the retransmitted packets. If the receiver detects an OOO event, it should notify the sender by setting a specific option bit, called the OOO bit, in the ACK packet header.

Once the TCP sender knows about an OOO event, it takes the following two response actions: it temporarily disables congestion control, and instantly recovers during congestion avoidance. In the former action, the TCP sender disables the congestion algorithm for a specific time period (T_1). In the latter action, if the congestion control algorithm was invoked during the past time period (T_2), the TCP sender should recover immediately to the state before the invocation of the congestion control. In fact, the authors make the time periods T_1 and T_2 a function of the RTT.

In the simulation study presented in [18], different scenarios are considered by combining all the mechanisms and actions mentioned above. Their results show that sender-based and receiver-based mechanisms behave similarly. Thus, they recommend the use of the sender detection mechanism as it does not require notifications from the sender to the receiver. Regarding the two actions mentioned above to be taken upon an OOO event detection, they have found that both actions lead to significant improvement. In general, TCP DOOR improves TCP performance up to 50 percent. Nevertheless, the supposition that OOO events are the exclusive results of route failure deserves much more analysis. Actually, multipath routing protocols such as TORA [60] may produce OOO events that are not related to route failures.

Comparison — Six proposals have been presented. These proposals address the problem of TCP's inability to distinguish between losses due to route failures and losses due to network congestion. The authors of these proposals proceed with a TCP and network cross layer solution, such as TCP-F, ELFN, ATCP, and TCP-BuS, or a TCP layered solution, such as Fixed RTO and TCP-DOOR. The four TCP and network cross layer proposals, TCP-F, ELFN, ATCP, and TCP-BuS, are based on an explicit notification from the network layer to detect route failures, but they differ in how to detect route reestablishments. TCP-F and TCP-BuS use the explicit notification from the network layer, while ELFN and ATCP use a probing mechanism. Compared with the explicit notification, the probing mechanism is easy to implement, but what is the optimal value of the probing interval? And what is the implication of this mechanism in the case of high load? In particular we saw that in the case of high load, the ELFN proposal performs worse than standard TCP. Among the four cross layer proposals, ATCP emerges as a robust proposal as it supports mechanisms to alleviate the negative impact of high BER and packets out-of-order on TCP performance. However, this proposal needs to revise certain assumptions, such as the existence of an ECN-capable node. Table 1 compares the

key features of the four TCP and network cross layer proposals. The TCP layer proposals, Fixed RTO and TCP DOOR, employ an end-to-end TCP layer approach to distinguish between packet losses induced by routes failures and losses caused by congestion. In Fixed RTO, this is done by considering two consecutive timeouts as a sign of route failures. In TCP-DOOR, when the source or the sink receives out-of-order packets, it concludes that a route failure has occurred. The main advantage of these two proposals is that they do not require explicit notification from the routing layer, nor do they require the cooperation of other nodes to detect route failures. Comparing these two proposals, TCP DOOR performs better than fixed RTO, but at the cost of more modifications.

PROPOSALS TO REDUCE ROUTE FAILURES

The proposals that address the problem of frequent route failures in MANETs can be classified into three categories: TCP and network cross layer proposals; network and physical cross layer proposals; and network layer proposals.

TCP and Network Cross Layer Proposal

Split TCP — TCP connections that have a large number of hops suffer from frequent route failures due to mobility. To improve the throughput of these connections and to resolve the unfairness problem, the Split TCP scheme was introduced to split long TCP connections into shorter localized segments [61] (see Fig. 6). The interfacing node between two localized segments is called a proxy. The routing agent decides if this node has the role of proxy according to the inter-proxy distance parameter. The proxy intercepts TCP packets, buffers them, and acknowledges their receipt to the source (or previous proxy) by sending a local acknowledgment (LACK). Also, a proxy is responsible for delivering the packets at an appropriate rate to the next local segment. Upon receipt of a LACK (from the next proxy or from the final destination), a proxy will purge the packet from its buffer. To ensure source-to-destination reliability, an ACK is sent by the destination to the source, similar to what occurs in standard TCP. In fact, this scheme also splits the transport layer functionalities into end-to-end reliability and congestion control. This is done by using two transmission windows at the source, the congestion window and the end-to-end window. The congestion window is a sub-window of the end-to-end window. While the congestion window changes in accordance with the rate of arrival of LACKs from the next proxy, the end-to-end window will change in accordance with the rate of arrival of the end-to-end ACKs from the destination. At each proxy there would be a congestion window that would govern the rate of sending between proxies.

Simulation results indicate that an inter-proxy distance of between three and five impact favorably on both throughput and fairness. The authors report that an improvement of up to 30 percent can be achieved in the total throughput by using Split TCP. The drawbacks are large buffers and network overhead. Also, this proposal makes the role of proxy nodes more complex, as for each TCP session they have to control packet delivery to succeeding proxies.

Network and Physical Cross Layer Proposals

Preemptive Routing in Ad Hoc Networks — As we reported earlier, in MANETs TCP may suffer from long idle periods induced by frequent route failures. This proposal [62] addresses this problem by reducing the number of route failures. In addition, it also reduces route reconstruction latency. These are achieved by switching to a new route when a link of the current route is expected to fail in the future (see below). This technique is coupled with the on-demand routing protocols AODV and DSR. The mechanism to detect failure is

	TCP-F	ELFN	ATCP	TCP-BuS
High BER packet loss	Not handled	Not handled	Handled	Not handled
Route failures (RF) detection	RFN packet freezes TCP sender state	ELFN packet freezes TCP sender state	ICMP "destination unreachable" freezes TCP sender state	ERDN packet freezes detection TCP sender state
Route reconstruction (RR) detection	RRN packet resumes TCP to normal state	Probing mechanism	Probing mechanism	ERSN packet resumes TCP to normal state
Packet reordering	Not handled	Not handled	Handled	Not handled
Congestion window and RTO after RR	Old CW and RTO	Old CW and RTO	Reset for each new route	Old CW and RTO
Reliable transmission of control messages	Not handled	Not handled	Not handled	Handled
Evaluation	Emulation no routing protocol considered	Simulation	Experimental no routing protocol considered	Simulation

■ Table 1. Comparison of TCP and network cross layer proposals to distinguish between losses due to route failures and congestion.

power-based. More specifically, when an intermediate node along a route detects that the signal power of a packet received from its upstream node drops below a given threshold, called the preemptive threshold, this intermediate node will detect a routing failure. For example, in Fig. 7, when node 4 senses that the signal power of a packet received from node 5 drops below the preemptive threshold, it will detect the routing failure event. On detecting this event, node 4 will notify the source of the route, node 1. On receiving this notification, the source's routing agent proactively looks up a new route. When the new route is available, the routing agent switches to this new route. The value of the preemptive threshold appears to be critical. Indeed, in the case of a low threshold value, there will not be sufficient time to discover an alternate path before the route fails. Also, in the case of a high threshold value the warning message will be generated too early. To overcome the fluctuations of the received signal power due to channel fading and multipath effects, which may trigger a preemptive route warning and cause unnecessary route request floods, the authors use a repeated short message probing to verify the correctness of the warning message. For example, in Fig. 7 when the signal power of the packet sent from node 5 and received at node 4 drops below the preemptive threshold, node 4 sends a ping message to node 5. On receiving this message, node 5 replies with a pong message. On receiving the pong message, node 4 checks the signal power of the pong message to verify that the link 4 – 5 is going to fail. The ping-pong handshake is repeated several times.

Using simulations the authors show that their scheme yields a reduction of the number of route failures and decreases latency by 30 percent. It should be noted that this scheme is "packet receipt event-driven" and that failures cannot be detected if no packets are transmitted.

Signal Strength-based Link Management in Ad Hoc Networks — This algorithm [36] is similar to the previous algorithm. However, in this algorithm each

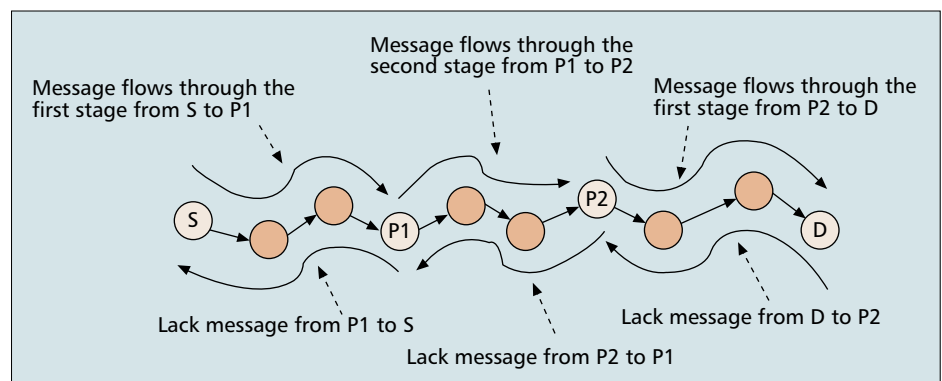
node keeps a record of the received signal strengths of 1-hop neighboring nodes. Using these records, the routing protocol predicts link break events in the immediate future.⁶ This prediction is called Proactive Link Management. On detecting this event, the source's routing agent is notified by a Going Down message (see Fig. 7). On receiving this message the source's routing agent stops sending packets and initiates a route discovery procedure. The novelty of this proposal is the Reactive Link Management mechanism, which increases the transmission power to re-establish a broken link. Reactive and Proactive Link Management mechanisms can be coupled in the following way: on predicting that a link is going to be down, the node's routing agent notifies the source to stop sending, and this node increases its transmitting power to handle packets in transit that use this link.

The authors use simulations to show that their scheme yields a 45 percent improvement in TCP performance. Note that only light load scenarios are considered.

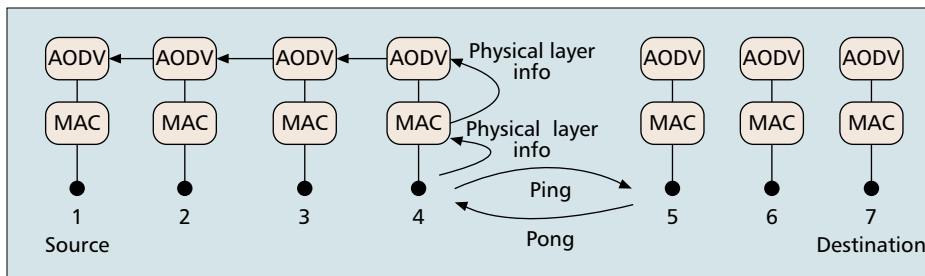
Network Layer Proposal

Backup Path Routing — This proposal [44] aims to improve the path availability of TCP connections using multipath routing. The authors found that the original multipath routing deteriorates TCP performance due to the inaccuracy in average

⁶ Immediate future means after 0.1sec.



■ Figure 6. TCP split.



■ **Figure 7.** Network and physical cross layer.

RTT measurement and out-of-order packet delivery. Thus, they introduce a new variation of multipath routing, called backup path routing. The backup path routing proposal maintains several paths from source to destination, but it only uses one path at any time. When the current path breaks, it can quickly switch to an alternate path. Using simulations, the authors observed that maintaining one primary path and one alternate path for each destination yields the best TCP performance. For the selection criteria of paths, the authors consider two schemes. The first scheme consists of selecting the shortest-hop path as the primary and the shortest-delay path as the alternate. The second scheme consists of selecting the shortest-delay path as the primary and the maximally disjoint path as the alternative. The alternate maximally disjoint path is the path that has the fewest overlapped intermediate nodes with the primary path. Comparing the two selection schemes, they found that the first scheme outperforms the second scheme. As a result of using the second scheme, routes tend to be longer in number of hops. Comparing TCP performance over the DSR routing protocol with backup routing, the authors report an improvement in TCP throughput of up to 30 percent with a reduction in routing overheads. These results are based on various mobility and traffic load scenarios. However, the authors did not evaluate the scheme in which the shortest path is selected as the primary and the maximal disjoint as the alternative.

Comparison — Four proposals have been presented. These proposals address the problem of frequent route failures that induce long idle time. The authors of these proposals proceed with a TCP and network cross layer solution such as Split TCP, or a network and physical cross layer solution such as preemptive routing, and signal strength-based links, or a network layer solution, such as backup routing. The main cause of route failures in MANETs is node mobility. Split TCP is based on splitting long TCP connections, in term of hops, into short segments to decrease the number of routing failures. However, this generates increased overhead. In preemptive routing and signal strength-based link management, the problem is addressed by predicting link failures and initiating a route reconstruction before the current route breaks. Signal strength-based link management proposals use a more robust approach to predict failures than preemptive routing. Backup routing improves TCP path availability by storing an alternate path that is used when a routing failure is detected. Backup routing reports an improvement in TCP throughput of up to 30 percent with a reduction in routing overhead, but further evaluations are needed, especially for the route selection criteria.

PROPOSALS TO REDUCE WIRELESS CHANNEL CONTENTION

Proposals that address the problem of contention on the wireless channel in SANETs can be grouped into three categories: TCP layer proposals, network layer proposals, and link layer proposals.

TCP Layer Proposals

Dynamic Delayed Ack — This approach [20] aims to reduce the contention on wireless channels by decreasing the number of TCP ACKs transmitted by the sink. It is a modification of the delayed ACK option (RFC 1122) that has a fixed coefficient $d = 2$. In fact, d represents the number of TCP packets that the TCP sink should receive before it acknowledges these packets. In this approach, the value of

d is not fixed and it varies dynamically with the sequence number of the TCP packet. For this reason the authors define three thresholds, l_1 , l_2 , and l_3 , such that $d = 1$ for packets with sequence number N smaller than l_1 , $d = 2$ for packets with $l_1 \leq N \leq l_2$, $d = 3$ for $l_2 \leq N \leq l_3$, and $d = 4$ for $l_3 \leq N$. In their simulations they study the packet loss rate, throughput, and session delay of TCP New Reno, in the case of short and persistent TCP sessions on a static multihop chain. They show that their proposal, with $l_1 = 2$, $l_2 = 5$, and $l_3 = 9$, outperforms standard TCP as well as the delayed ACK 21 option for a fixed coefficient $d = 2, 3, 4$. They suggest that better performance could be obtained by making d a function of the sender's congestion window instead of a function of the sequence number.

Network Layer Proposals

COPAS — The COnTention-based PAtH Selection proposal [63] addresses the TCP performance drop problem caused by contention on the wireless channel. It implements two techniques: the first employs disjoint forward and reverse routes, which consists of selecting disjoint routes for TCP data and TCP ACK packets; the second technique is dynamic contention-balancing, which consists of dynamically updating disjoint routes. When the contention of a route exceeds a certain threshold, called the backoff threshold, a new and less contended route is selected to replace the high contended route. In this proposal the contention on the wireless channel is measured as a function of the number of times a node has backed off during each interval of time. Also, any time a route is broken, in addition to initiating a route re-establishment procedure, COPAS redirects TCP packets using the second alternate route. Comparing COPAS and DSR, the authors found that COPAS outperforms DSR in term of TCP throughput and routing overheads. The improvement of TCP throughput is up to 90 percent. However, the use of COPAS, as reported by the authors, is limited to static networks or networks with low mobility because as nodes move faster, using a disjoint forward and reverse route increases the probability of route failures experienced by TCP connections. This may induce more routing overhead and more packet losses.

Link Layer Proposals

Link RED — Link Random Early Detection (RED) [21] aims to reduce contention on the wireless channel by monitoring the average number of retransmissions (avg) at the link layer. When the avg number becomes greater than a given threshold, the probability of dropping/marking is computed according to the RED algorithm [51]. Since it marks packets, Link RED can be coupled with ECN to notify the TCP sender about the congestion level [58]. However, instead of notifying the TCP sender about the congestion level, the authors increase the backoff time at the MAC layer.

Adaptive Pacing — The goal of this proposal [21] is to improve spatial channel reuse. In the current IEEE 802.11 protocol, a node is constrained from contending for the chan-

nel by a random backoff period, plus a single packet transmission time that is announced by the RTS or CTS frame. However, the exposed receiver problem persists due to the lack of coordination between nodes that are two hops away from each other. Adaptive pacing solves this problem by increasing the backoff period by an additional packet transmission time. This proposal works together with Link RED as follows. Adaptive pacing is enabled by Link RED. When a node finds its average number of transmission retries to be less than a threshold, it calculates its backoff time as usual. When the average number of retries goes beyond this threshold, adaptive pacing is enabled and the backoff period is increased by a duration equal to the transmission time of the previous packet. Working together, Link RED and adaptive pacing have reported an improvement in TCP throughput as well as the fairness between multiple TCP sessions. However, in this proposal the additional backoff time is based on the packet size, so the existence of different data packet sizes in the network should be inspected.

Comparison — Three proposals have been presented. These proposals address the problem of contention on wireless channel, which is the main cause of TCP performance degradation in SANETs. The authors of the proposals proceed with a TCP layer proposal, such as Dynamic delayed ACK, or with a network layer proposal, such as COPAS, or with a link layer proposals, such as LRED/adaptive pacing. Dynamic delayed ACK is a simple approach that aims to reduce the contention on wireless channels by decreasing the number of TCP ACKs transmitted by the sink. However, COPAS attacks this problem by using disjoint forward and reverse routes, and dynamic updating of disjoint routes based on contention level. COPAS reports an improvement in TCP throughput of up to 90 percent. LRED and adaptive pacing attack the contention problem from its origin at the link layer by detecting contention and increasing the backoff time at the link layer before packet transmission. However, these two proposals are based on the packet size to compute the backoff time, and this may generate problems in the case of a network that supports multiple packet sizes.

PROPOSALS TO IMPROVE TCP FAIRNESS

The proposals that address the problem of TCP unfairness in SANETs can be classified into one category: link layer proposals.

Link Layer Proposals

Non Work-Conserving Scheduling — The goal of this proposal [64] is to improve fairness among TCP flows crossing wireless ad hoc and wired networks. The authors adopt the “non work-conserving scheduling” policy for ad hoc networks instead of the “work-conserving scheduling.” This is done as follows. The link layer queue⁷ sets a timer whenever it sends a data packet to the MAC. The queue outputs another packet to the MAC only when the timer expires. The duration of the timer is updated according to the queue output rate value. Specifically, the duration of the timer is a sum of three parts, D_1 , D_2 , and D_3 . D_1 is equal to the data packet length divided by the bandwidth of the channel. D_2 is a delay, the value of which is decided by the recent queue output rate. D_3 is a random value uniformly distributed between 0 and D_2 . D_3 is used to avoid synchronization problems and to reduce collisions.

The queue calculates the output rate by counting the number of bytes, C , it outputs in every fixed interval T . To determine the value of D_2 , the authors set three thresholds, X , Y , and Z ($X < Y < Z$) for C , and four delay values, D_{21} , D_{22} ,

D_{23} , and D_{24} ($D_{21} < D_{22} < D_{23} < D_{24}$) for D_2 , as shown in Eq. 1. The heuristic behind Eq. 1 is to penalize greedy nodes with a high output rate by increasing their queuing delay D_2 and to favor nodes with small output rates. By means of simulations, the authors report that their scheme greatly improves fairness among TCP connections at the cost of moderate total throughput degradation.

$$\begin{cases} D_2 = D_{21} & \text{for } C \leq X \\ D_2 = D_{22} & \text{for } X < C \leq Y \\ D_2 = D_{23} & \text{for } Y < C \leq Z \\ D_2 = D_{24} & \text{for } C > Z \\ 0 \leq D_{21} < D_{22} < D_{23} < D_{24} \end{cases} \quad (1)$$

Neighborhood RED — This proposal [50] aims to enhance TCP fairness in MANETs. Unlike in wired networks, the authors show that RED does not solve TCP’s unfairness in MANETs because the congestion does not happen in a single node, but in an entire area involving multiple nodes. The local packet queue at any single node cannot completely reflect the network congestion state. For this reason the authors define a new distributed queue, called the neighborhood queue. At a node, the neighborhood queue should contain all packets whose transmissions will affect its own transmission in addition to its packets.⁸ Since it is difficult to get information about all these packets without introducing significant communication overhead, which may need 2-hop information exchange, a simplified node neighborhood queue is introduced. It aggregates the node’s local queue and the upstream and downstream queues of its 1-hop neighbors. Now the RED algorithm is based on the average queue size of the neighborhood queue. The authors use a distributed algorithm to compute the average queue size. In this algorithm the time is slotted. During each time slot, the idle period of the channel is measured. Using this measurement, a node estimates the channel utilization and the average neighborhood queue size. The accuracy of the estimation is controlled by the duration of the slots. Using simulations of SANETs, they verify the effectiveness of their proposal and the fairness improvement of TCP.

Comparison — Two proposals have been presented. These proposals address the problem of TCP unfairness in SANETs. To deal with the TCP unfairness problem, the authors of the proposals proceed with link layer proposals, such as non work-conserving scheduling and neighborhood RED. In the non work-conserving scheduling proposal, this is done by penalizing greedy nodes with high output rate by increasing their queuing delays at the link layer. However, in neighborhood RED it is up to TCP to regulate its transmission rate when it senses packets being dropped. The RED algorithm in this proposal is applied to a distributed queue, called the neighborhood queue, which does not contain the node local queue as well as the upstream and downstream queues of its 1-hop neighbor. Neighborhood RED is better than the non work-conserving approach as it does not cause degradation in total throughput.

GENERAL COMPARISON AND DISCUSSION

It is difficult to compare the different proposals that aim to improve TCP’s performance in MANETs. Some of them have been designed to solve different problems. By examining all of these proposals, we identify four major problems:

⁷ In NS-2, Link layer queue is called InterFace queue (IFq).

⁸ Affect means: prevent a node from transmitting because of channel capturing, or to induce transmission failure to carrier sensing MAC protocols.

- TCP is unable to distinguish between losses due to route failures and losses due to network congestion.
- Frequent route failures.
- Wireless channel contention.
- TCP unfairness.

We note that the first two problems are the main causes of TCP performance degradation in MANETs. However, the second two problems are the main causes of TCP performance degradation in SANETs. To solve these problems the authors of the proposals proceed with a cross layer solution or a layered solution. In terms of complexity, cross layer solutions are more complex to implement and design than layered solutions, because the implementation of cross layer solutions requires at least the modification of two OSI layers. Also, as they break the concept of designing protocols to be independent of other layer protocols, their design requires that we consider the system in its entirety. In the following, we compare the proposals that share a common problem. Table 2 contains a detailed comparison of these proposals. The comparison is based on five features:

- The first one is the solution type, which can be cross layer or layered.
- The degree of complexity.
- The network type.
- TCP connection load and type.
- The basis of evaluation.

We identify three degrees of complexity: high, medium, and low. Layered solutions are considered to be of medium or low complexity, whereas cross layer solutions are considered to be of medium or high complexity.

CONCLUSION

We have presented the state-of-the-art of TCP over static and mobile ad hoc networks (SANETs and MANETs). The principal problem of TCP in this MANET environment is clearly its inability to distinguish between losses induced by network congestion and others types of losses. TCP assumes that losses are always due to network congestion. While this assumption in most cases is valid in wired networks, it is not true in MANETs. In MANETs, there are indeed several types of losses, including losses caused by routing failures, by network partitions, and by high bit error rates. Performing congestion control in these cases, as TCP does, yields poor performance. In static multi-hop ad hoc networks the principal problem of TCP is the contention on the wireless channel that induces route failures and losses. In order to solve these problems, several proposals have been made in the literature. We classified these proposals as layered proposals and cross layer proposals. In cross layer proposals, TCP and the underlying protocols cooperate to improve ad hoc network performance. For example, TCP and network cross layer proposals use explicit notification to inform TCP about route failures, which requires cooperation from intermediate nodes as in TCP-F, TCP-ELFN, ATCP, and TCP-BuS. In layered proposals, one OSI layer is adapted. For example, TCP layer proposals require only the cooperation of the sender and receiver, as in TCP-DOOR and Fixed-RTO. However, cross layer proposals yield higher improvement than layered proposals.

The frequent route failures and route re-establishments in MANET environments introduce a new challenge to the TCP congestion control algorithm, and leads us to pose the following questions: Does the actual congestion control algorithm behave efficiently in dynamic environments such as MANETs? Are the parameter values that are used in TCP to compute the retransmission timeout after a route re-establishment valid in MANETs? As an extension of this work we intend to answer these questions.

ACKNOWLEDGEMENTS

We would like to thank Daniele Miorandi and Urtzi Ayesta for their help. This work was supported in part by the EuroNGI Network of Excellence (NoE).

REFERENCES

- [1] "Bluetooth Special Interest Group," Web site: <http://www.bluetooth.com>
- [2] "IEEE 802.11 WLAN standard," Web site: <http://standards.ieee.org/getieee802>
- [3] "Broadband Radio Access Networks (BRAN): High Performance Local Area Network (HiperLAN) type 2," Tech. Rep. 101 683 V1.1.1, ETSI.
- [4] "Transmission Control Protocol," RFC 793, Sep. 1981.
- [5] L. Andrew, S. Hanly, and R. Mukhtar, "CLAMP: Differentiated Capacity Allocation in Access Networks," *Proc. IEEE Int'l. Performance Comp. and Commun. Conf.*, Phoenix, AZ, USA, Apr. 2003, pp. 451–58.
- [6] H. Balakrishnan *et al.*, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE Trans. Net.*, vol. 5, no. 6, Dec. 1997, pp. 756–69.
- [7] H. Balakrishnan *et al.*, "Improving TCP/IP Performance over Wireless Networks," *Proc. ACM MOBIHOC*, Berkeley, CA, USA, 1995, pp. 2–11.
- [8] A. V. Bakre and B. R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE Trans. Net.*, vol. 46, no. 3, Mar. 1997, pp. 260–78.
- [9] K. Brown and S. Singh, "M-TCP: TCP for Mobile Cellular Networks," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 27, no. 5, Oct. 1997, pp. 19–43.
- [10] H. Balakrishnan, S. Seshan, and R. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," *ACM Wireless Networks*, vol. 1, no. 4, Dec. 1995, pp. 469–81.
- [11] R. Durst, G. Miller, and E. Travis, "TCP Extensions for Space Communications," *Proc. ACM MOBICOM*, Rye, NY, 1996, pp. 15–26.
- [12] T. Henderson and R. Katz, "Transport Protocols for Internet-compatible Satellite Networks," *IEEE JSAC*, vol. 17, no. 2, Feb. 1999, pp. 345–59.
- [13] S. Xu and T. Saadawi, "Performance Evaluation of TCP Algorithms in Multi-hop Wireless Packet Networks," *Journal of Wireless Commun. and Mobile Computing*, vol. 2, no. 1, pp. 85–100, 2002.
- [14] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in Heterogeneous Networks: A Survey," *IEEE Commun. Mag.*, vol. 38, no. 1, Jan. 2000, pp. 40–46.
- [15] K. Chandran *et al.*, "A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks," *Conf. Distributed Comp. Systems*, Amsterdam, The Netherlands, May 1998, pp. 472–79.
- [16] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *ACM Wireless Networks*, vol. 8, no. 2, Mar. 2002, pp. 275–88.
- [17] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," *IEEE JSAC*, vol. 19, no. 7, pp. 1300–1315, July 2001.
- [18] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad Hoc Networks with Out-of-order Detection and Response," *Proc. ACM MOBIHOC*, Lausanne, Switzerland, June 2002, pp. 217–25.
- [19] D. Kim, C. Toh, and Y. Choi, "TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks," *J. Commun. and Net.*, vol. 3, no. 2, June 2001, pp. 175–86.
- [20] E. Altman and T. Jimenez, "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks," *Proc. Pers. Wireless Commun.*, Venice, Italy, Sep. 2003, pp. 237–53.
- [21] Z. Fu *et al.*, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," *Proc. IEEE INFOCOM*, San Francisco, USA, Apr. 2003.
- [22] K. Chin *et al.*, "Implementation Experience with MANET Routing Protocols," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 32, no. 5, Nov. 2002, pp. 49–59.
- [23] C. Perkins and T. Watson, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proc. ACM SIGCOMM*, London, UK, 1994.

Problem	Proposal	Solution type	Complexity degree	Network type	TCP connections load/type	Evaluation
TCP is unable to distinguish between losses due to route failures and congestion	ELFN [16]	TCP-net. cross layer	Medium	Random mobile	One persistent	Simulation no routing
	ATCP [17]	TCP-net. cross layer	High	Random mobile	One persistent	Experimental no routing
	TCP-BuS [19]	TCP-net. cross layer	High	Random mobile	Multiple persistent	Simulation no routing
	TCP-F [55]	TCP-net. cross layer	Medium	Random mobile	One persistent	Emulation no routing
	TCP-DOOR [18]	TCP layer	Low	Random mobile	One persistent	Simulation
	TCP-RTO [42]	TCP layer	Low	Random mobile	One persistent	Simulation
	Frequent route failures	Split TCP [61]	TCP-network cross layer	Medium	Random mobile	One persistent
Preemptive routing [62]		Network-physical cross layer	Medium	Random mobile	no TCP load	Simulation
Signal strength based link [36]		Network-physical Cross	High	Random mobile	two persistent	Simulation
Backup routing [44]		Network layer	Low	Random mobile	one persistent	Simulation
Contention on wireless channel	Dynamic delayed ACK [20]	TCP layer	Low	Static chain	One persistent/short	Simulation
	COPAS [63]	Network layer	Medium	Static random	Multiple persistent	Simulation
	LRED/adaptive pacing [21]	Link layer	Low	Static	Multiple persistent	Simulation
TCP unfairness	Non work conserving [64]	Link layer	Low	Static chain	Multiple persistent	Simulation
	Neighborhood RED [50]	Link layer	Low	Static/mobile	Multiple persistent	Simulation

■ Table 2. General comparison of proposals.

- [24] C. Perkins, E. Belding-Royer, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, Category: Experimental, July 2003.
- [25] F. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II — The hidden Terminal Problem in Carrier Sense Multiple-Access Modes and the Busy-Tone Solution," *IEEE Trans. Net.*, vol. 23, no. 12, 1975, pp. 1417–33.
- [26] V. Bharghavan *et al.*, "MACAW: A Media Access Protocol for Wireless LAN's," *Proc. ACM SIGCOMM*, London, 1994, pp. 212–25.
- [27] K. Xu, M. Gerla, and S. Bae, "Effectiveness of RTS/CTS Handshake in IEEE 802.11-Based Ad Hoc Networks," *Ad Hoc Net. J.*, Elsevier, vol. 1, no. 1, July 2003, pp. 107–23.
- [28] "The Network Simulator NS-2," Web site: <http://www.isi.edu/nsnam/ns/index.html>.
- [29] "Global Mobile Information Systems Simulation Library GloMoSim," Web site: <http://pcl.cs.ucla.edu/projects/gloimosim/>.
- [30] A. Kamerman and L. Monteban., "Wavelan II: A High-Performance Wireless LAN for the Unlicensed Band," *Bell Labs Tech. J.*, Summer 1997, pp. 118–33.
- [31] V. Jacobson, "Compression TCP/IP Headers for Low Speed Serial Links," RFC 1144, Category: Proposed Standard, Feb. 1990.
- [32] H. Balakrishnan and V. Padmanabhan, "How Network Asymmetry Affects TCP," *IEEE Commun. Mag.*, Apr. 2001, pp. 2–9.
- [33] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," RFC 2988, Category: Standard Track, Nov. 2000.
- [34] H. Lundgren, E. Nordstro, and C. Tschudin, "Coping with Communication Gray Zones in IEEE 802.11b-Based Ad Hoc Networks," *Proc. ACM Wksp. Wireless Mobile Multimedia*, Atlanta, GA, USA, Sept. 2002, pp. 49–55.
- [35] C. Jones *et al.*, "A Survey of Energy Efficient Network Protocols for Wireless and Mobile Networks," *ACM Wireless Net.*, vol. 7, no. 4, 2001, pp. 343–58.
- [36] F. Klemm, S. Krishnamurthy, and S. Tripathi, "Alleviating Effects of Mobility on TCP Performance in Ad Hoc Networks using Signal Strength-Based Link Management," *Proc. Pers. Wireless Commun.*, Venice, Italy, Sept. 2003, pp. 611–24.

- [37] M. Chiang, "Balancing Transport and Physical Layers in Wireless Ad Hoc Networks: Jointly Optimal TCP Congestion Control and power control," *IEEE JSAC*, vol. 23, no. 1, Jan. 2005, pp. 104–16.
- [38] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A Review of Routing Protocols for Mobile Ad Hoc Networks," *Journal of Ad Hoc Networks*, Elsevier, vol. 2, no. 1, Jan. 2004, pp. 1–22.
- [39] I. Chlamtac, M. Conti, and J. Liu, "Mobile Ad Hoc Networking: Imperatives and Challenges," *Ad Hoc Networks Journal*, Elsevier, vol. 1, no. 1, July 2003, pp. 13–64.
- [40] D. Johnson, D. Maltz, and Y. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," Internet draft, Apr. 2003.
- [41] V. Anantharaman *et al.*, "TCP Performance over Mobile Ad Hoc Networks: A Quantitative Study," *J. Wireless Commun. and Mobile Computing*, vol. 4, no. 2, Mar. 2004, pp. 203–22.
- [42] T. Dyer and R. Boppana, "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks," *Proc. ACM MOBIHOC*, Long Beach, CA, USA, 2001, pp. 56–66.
- [43] R. Boppana and S. Konduru, "An Adaptive Distance Vector Routing Algorithm for Mobile, Ad Hoc Networks," *Proc. IEEE INFOCOM*, Anchorage, Alaska, USA, Apr. 2001.
- [44] H. Lim, K. Xu, and M. Gerla, "TCP Performance over Multipath Routing in Mobile Ad Hoc Networks," *Proc. IEEE ICC*, Anchorage, Alaska, May 2003.
- [45] S. Lu and M. Gerla, "Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks," *Proc. IEEE ICC*, Helsinki, Finland, June 2001.
- [46] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks," *IEEE Commun. Mag.*, vol. 39, no. 6, June 2001, pp. 130–37.
- [47] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multi-Hop Networks," *Proc. IEEE WMCSA*, New Orleans, LA, USA, 1999.
- [48] K. Tang and M. Gerla, "Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks," *Proc. IEEE Multiclass Mobility and Teletraffic for Wireless Commun. Wksp.*, Venice, Italy, Oct. 1999.
- [49] K. Xu, S. Bae, S. Lee, and M. Gerla, "TCP Behavior Across Multihop Wireless Networks and the Wired Internet," *Proc. ACM Wksp. Wireless Mobile Multimedia*, Atlanta, GA, USA, Sep. 2002, pp. 41–48.
- [50] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP Fairness in Ad Hoc Wireless Networks using Neighborhood Red," *Proc. ACM MOBICOM*, San Diego, CA, USA, Sep. 2003, pp. 16–28.
- [51] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE Trans. Net.*, vol. 1, no. 4, Aug. 1993, pp. 397–413.
- [52] G. Anastasi, M. Conti, and E. Gregori, "IEEE 802.11 Ad Hoc Networks: Performance Measurements," *Proc. Wksp. Mobile and Wireless Network (MWM 2003) in conjunction with ICDCS 2003*, May 2003.
- [53] K. Chen, Y. Xue, and K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks," *Proc. IEEE ICC*, Anchorage, Alaska, USA, May 2003.
- [54] V. Kawadia and P. Kumar, "A Cautionary Perspective on Cross Layer Design," *IEEE Wireless Commun. Mag.*, vol. 12, no. 1, Feb. 2005, pp. 3–11.
- [55] K. Chandran *et al.*, "A Feedback-Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks," *Proc. Int'l. Conf. Distributed Computing Systems (ICDCS'98)*, Amsterdam, Netherlands, May 1998.
- [56] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Category: Experimental, Oct. 2003.
- [57] J. Monks, P. Sinha, and V. Bharghavan, "Limitations of TCP-ELFN for Ad Hoc Networks," *Proc. Mobile and Multimedia Commun.*, Tokyo, Japan, Oct. 2000.
- [58] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, Category: Standards Track, Sept. 2001.
- [59] C. Toh, "Associativity-Based Routing for Ad Hoc Mobile Networks," *J. Wireless Pers. Commun.*, vol. 4, no. 2, Mar. 1997, pp. 103–39.
- [60] C. Perkins, *Ad Hoc Networking*, Addison-Wesley, Upper Saddle River, NJ, USA, 2001.
- [61] S. Kopparty *et al.*, "Split TCP for Mobile Ad Hoc Networks," *Proc. IEEE GLOBECOM*, Taipei, Taiwan, Nov. 2002.
- [62] T. Goff *et al.*, "Preemptive Routing in Ad Hoc Networks," *Proc. ACM MOBICOM*, Rome, Italy, 2001, pp. 43–52.
- [63] C. Cordeiro, S. Das, and D. Agrawal, "COPAS: Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-Hop Wireless Networks," *Proc. IC3N*, Miami, USA, Oct. 2003, pp. 382–387.
- [64] L. Yang, W. Seah, and Q. Yin, "Improving Fairness among TCP Flows Crossing Wireless Ad Hoc and Wired Networks," *Proc. ACM MOBIHOC*, Annapolis, Maryland, USA, June 2003, pp. 57–63.
- [65] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM*, Vancouver, Canada, Aug. 1998.
- [66] K. Fall and S. Floyd, "Simulation-Based Comparisons of Tahoe, Reno, and Sack TCP," *Comp. Commun. Review*, Jul. 1996.
- [67] J. Heo, "Improving the Start Up Behavior of a Congestion Control Scheme for TCP," *Proc. ACM SIGCOMM*, CA, USA, Aug. 1996.
- [68] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP over Satellite Channels using Standard Mechanisms," RFC 2488, Jan. 1999.

BIOGRAPHIES

AHMAD AL HANBALI (Ahmad.Al_Hanbali@sophia.inria.fr) is a Ph.D. student at the University of Nice - Sophia Antipolis, France. In July 2002 he received his engineering degree in Telecommunications and Computer Science from the Lebanese University of Beirut. In June 2003 he received the Master's degree in networking and distributed systems from the University of Nice — Sophia Antipolis, France. He then joined the Ph.D. program with MAESTRO research group at INRIA - Sophia Antipolis. His main research interests are performance evaluation of TCP and mobility models of ad hoc networks.

E. ALTMAN (Eitan.Altman@sophia.inria.fr) received the B.Sc. degree in electrical engineering in 1984, the B.A. degree in physics in 1984, and the Ph.D. degree in electrical engineering in 1990, all from the Technion-Israel Institute, Haifa. In 1990 he received his B.Mus. degree in music composition at Tel-Aviv University. Since 1990 he has been with INRIA (National Research Institute in Informatics and Control) in Sophia-Antipolis, France. His current research interests include performance evaluation and control of telecommunication networks, and in particular congestion control, wireless communications, and networking games. He is on the editorial board of several scientific journals: *Stochastic Models*, *JEDC*, *COMNET*, *SIAM SICON*, and *WINET*. He has been the co-chairman of the program committee of several international conferences and workshops on game theory, networking games, and mobile networks.

PHILIPPE NAIN (Philippe.Nain@sophia.inria.fr) received the Maîtrise Es-Sciences in mathematics in 1978, the Diplôme d'Etudes Approfondies in statistics in 1979, and the Doctorat de 3ème cycle specializing in modeling of computer systems in 1981, all from the University of Paris XI, Orsay, France. In 1987 he received the Doctorat d'Etat in applied mathematics from the University Pierre and Marie Curie, Paris, France. Since December 1981 he has been with INRIA, where he is currently the head of the research project Maestro devoted to the modeling of computer systems and telecommunications networks. He has held visiting appointments at the University of Massachusetts (1993-94), at the University of Maryland (1987), and at North Carolina State University (1988). His research interests include modeling and performance evaluation of communication networks. He is an associate editor of *Performance Evaluation and Operations Research Letters*, and was an associate editor of *IEEE Transactions on Automatic Control*. He was a co-program chair of the ACM Sigmetrics 2000 conference and the general chair of Performance 2005. He is a member of IFIP WG 7.3.

APPENDIX: OVERVIEW OF TCP VERSIONS

TCP is a window-based acknowledgment-clocked flow control protocol. It uses an additive-increase/multiplicative decrease strategy for changing its window as a function of network con-

ditions. Packets of a TCP connection are sent with increasing consecutive sequence numbers. In the simplest operation of TCP, at each arrival of a packet at the destination, an ACK is sent back to the source with information about the next sequence number that is expected. Thus, if all packets up to packet $n - 1$ have reached the destination, then the last arrival will trigger an ACK with sequence number n . If a packet n is lost in the network and packet $n + i$, $i = 1, 2, 3$ arrives at the destination, then each of these packets will trigger an acknowledgment indicating that the destination is expecting packet n . These are called duplicated ACKs. In the absence of losses, starting from one packet or from a larger value, the window is increased exponentially by one packet every non-duplicate ACK until the source estimate of network capacity is reached. This is the slow start (SS) phase, and the capacity estimate is called the slow start threshold (ssthresh). In most versions of TCP (Tahoe, Reno, and New Reno) once ssthresh is reached, the source switches to a slower increase in the window by one packet for every window's worth of ACKs. This phase is called the congestion avoidance (CA) phase. The window increase is interrupted when a loss is detected. Two mechanisms are available for the detection of losses: the expiration of a retransmission timer (timeout), or the receipt of three duplicate ACKs (the latter is called the fast retransmit (FRXT) phase). The source then sets its estimation of the capacity to half the current window. (This action is due to the fact that when these TCP versions were developed, losses were an indication of congestion, as TCP was then deployed only over wireline networks.) Tahoe [65], the first version of TCP to implement congestion control, at this point sets the window to one packet and enters the slow start phase to reach the new ssthresh. Slow starting after every loss detection deteriorates performance, given the low bandwidth utilization during SS. When the loss is detected via timeout, a more drastic reaction is taken as a more drastic congestion is understood to occur,

since the ACK stream has stopped. In the case of FRXT, ACKs still arrive at the source, and losses are recovered without SS. This is the behavior of the newer versions of TCP (Reno, NewReno, SACK, etc.) that call a Fast Recovery (FRCV) algorithm to retransmit the losses while maintaining enough packets in the network to preserve the ACK clock. Once the losses are recovered, this algorithm ends and normal CA is called. If FRCV fails to recover the losses, the ACK stream stops, a timeout occurs, and the source resorts to SS, as with Tahoe. Among the TCP versions that use the FRCV, the difference is in the estimation of the number of packets in flight during FRCV. Reno [66] considers every duplicate ACK a signal that a packet has left the network. The problem with Reno is that it leaves FRCV when an ACK for the first loss window is received. This prohibits the source from detecting the other losses with FRXT. A long timeout is required to detect the other losses. NewReno [67] has been proposed to overcome this problem. The idea is to stay in FRCV until all the losses in the same window are recovered. Another problem with Reno and NewReno is that they rely on ACKs to estimate the number of packets in flight. ACKs can be lost on the return path, which results in an underestimation of the number of packets that have left the network. More information is needed to estimate more precisely the number of packets in the pipe. This information is provided by the selective ACK (SACK) [68], a TCP option containing the three blocks of contiguous data most recently received at the destination. Finally, we mention TCP Vegas, which aims to decouple congestion detection from losses. In TCP Vegas, the RTT of the connection and the window size are used to compute the number of packets in the network buffers. The window is decreased when this number exceeds a certain threshold and is increased when it is below some threshold. In other words, Vegas also uses delay as a congestion indication and then reacts to reduce its throughput.