TCP in Wireless Mobile Ad Hoc Networks

Ruy de Oliveira

Torsten Braun

IAM-02-003

July 2002

TCP in Wireless Mobile Ad Hoc Networks*

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

Abstract

The Transmission Control Protocol (TCP) raises a number of issues when required to work in a wireless environment. In particular, within an ad hoc network, where changes can happen somewhat quickly and unpredictably, it has to deal with new tough challenges such as high probability of both network partition and route failures due to mobility. In order to adapt TCP to this demanding paradigm, some improvements have been proposed. Nevertheless, most of them present limited enhancements as they do not address important related issues. In this paper, we present an overall view on this subject in a detailed analysis of the major factors involved. Specifically, we show how TCP can be affected by mobility and lower layers strategies including path asymmetry. Additionally, we highlight the main features and weaknesses of the existing proposed schemes, and point out the main open issues in this area.

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

General Terms: Algorithms, Design, Performance.

Additional Keywords: Wireless ad hoc networks, Transmission Control Protocol, Congestion Control, Packet loss, Energy efficiency.

* This work was supported in part by the Swiss National Science Foundation under National Competence Center in Research for Mobile Information and Communication Systems (NCCR-MICS) project.

1. Introduction

Over the past few years, mobile ad hoc networks [RFC2501,MPC01,HDLP+02] have emerged as a promising approach for mobile IP applications of the future. This scheme can operate independently from existing underlying infrastructure and allows simple and fast implementation. Such features meet requirements primarily of applications for rescue operations, law enforcement, battlefield, etc. However, the more this technology evolves the higher is the probability that it will make part of the global Internet. Therefore, considerable research efforts have been put on this new challenging paradigm.

TCP was designed to work in wired networks and because of that its performance is quite poor when it is required to work in the typically *lossy* wireless environment. In recent years, several schemes have been proposed for satellite [DMT96,GD99,HK99] as well as for cellular [BB95,BSAK95,BS97,TB00,TMP00,ZT01] wireless networks, but much has still to be done on mobile ad hoc framework. In this sense, it is quite important the knowledge of both the main factors affection TCP performance in such an environment and the actual potentiality of the existing proposed solutions, which are addressed in this paper.

Unlike cellular networks, where only the last hop is based on a wireless medium, ad hoc networks are composed exclusively of wireless links, where *multihop* connections may be in place. Besides, in an ad hoc scenario all nodes can move freely and unpredictably, which makes the TCP congestion control quite hard since it is a clock based mechanism. Consequently, the error-detection and error-recovery strategies inherent in standard TCP need to be adapted in order to fit this environment. In particular, since the errors in this environment occurs not only due to congestion but also due to medium constraints and mobility, TCP needs to distinguish the nature of the error so that it can take the most appropriate action for each case. Additionally, the emerging link and network layer algorithms for this kind of network can play a key role on TCP performance. Likewise, factors such as *path asymmetry* (that may also be caused by lower layers strategies, among other elements) and *congestion window size* might also impair the performance of this protocol. All these issues are addressed in detail in the following sections.

Although there are a number of differences between cellular and ad hoc networks, some of the ideas developed in some proposed solutions for the former can be used in the latter as well. For example, in ad hoc environment TCP can be led to *persist mode* whenever a long disconnection occurs and under wireless medium induced losses it can simply retransmit the lost packet instead of invoking its congestion control mechanism. Both ideas have already been evaluated for cellular networks where only the last hop is of concern. In fact, most of the proposed solutions for TCP in ad hoc networks represent a mix of old concepts developed for cellular network properly adapted for this multihop scenario. However, we show here that distinct solutions are still demanded.

The rest of the paper is organized as follows. The next section explains the basic problems faced by TCP within ad hoc environments due to mobility and the effects of lower layers on TCP performance. In section 3 some existing proposed schemes for improving TCP in this scenario are presented. Directions toward a robust approach on this subject are summarized in section 4 and section 5 contains concluding remarks.

2. TCP interaction with wireless mobile ad hoc networks

Ad hoc networks pose some tough challenges to TCP due to the fact that it was not designed to operate in such a highly dynamic scenario in terms of topology. In reality, even though TCP has evolved significantly over the years toward a robust and reliable service protocol, the focus has

been primarily on wired networks. In this scenario, the additive-increase/multiplicative-decrease strategy coupled with the *fast recovery* and *fast retransmit* mechanisms [St94,RFC2581], inherent in most of current TCP versions, provide an effective congestion control at the end nodes. What is remarkable in the TCP congestion control mechanism is its ability to sharply decrease its transmission rate when the network faces congestion and make use of all leftover resources otherwise.

The key idea of TCP is to probe the network in order to determine the availability of resources. It injects packets at an increasing rate into the network until a packet loss is detected, whereby it infers the network is facing congestion. Then the TCP sender shrinks its congestion window (CWND), retransmits the lost packet and resumes transmission at a lower increasing rate. If the losses persist (no timely ACK received), at every retransmission the sender doubles (up to 64s) its wait timer, called Retransmission TimeOut (RTO), so that it can wait longer for the ACK of the current packet being transmitted. This is the *exponential backoff* strategy that is outlined in figure 2 as an example that is addressed later. More details on this mechanism can be found in [St94].

The explained mechanism works fine in a wired network where the Bit Error Rate (BER) is typically quite low allowing any lost packet to be treated as an indication of network congestion. In a wireless mobile ad hoc network, however, two more factors can induce packet losses in addition to network congestion: non-negligible *wireless medium losses* (high BER) and frequent *connectivity disruptions* caused by nodes mobility. As a result, TCP needs to be aware of what causes which loss.

Ad hoc networks face relevant level of BER because all links involved are wireless which suffer from fading and diverse interferences. Besides, since the nodes can move freely and unpredictably, there is a high probability of the peers of a ongoing connection get disconnected by lack of a route between them, and it can last a significant period of time. Such a discontinuity is called *partition* and it can be very degrading for TCP performance, as we explain later.

Another important characteristic of TCP regards its dependence on timely ACKs from the receiver to the sender in order to increase its data transfer rate. Thus, in case there is an asymmetrical path in place, imposing lower ACK rate compared to the data rate, TCP can be prevented from sending at a higher rate. In this way, considering that ad hoc networks are path asymmetry prone, as explained later, TCP algorithms in such an environment might also need to deal with *path asymmetry* issues.

At last, and no surprisingly, the lower layers (MAC and network) protocol strategies used in this scenario also play a key role on TCP performance, which demands special handling tailored to this ad hoc environment. In the following, we address all the issues above mentioned in order to clarify why and how they take place.

2.1 Effects of long Partitions on TCP

A network partition occurs when a given mobile node moves away, or is interrupted by the medium, thereby splitting its adjacent nodes into two isolated parts of the network that are called *partitions*. Figure 1 [LS01] depicts a condition in which a partition takes place and interrupts an ongoing connection between nodes 3 and 9. Figure 1a shows that node 5 moves away from node 3 and as a result the two partitions shown in figure 1b cannot communicate. This disruption will lead the connection to be halted. The reestablishment of the connection, through another route, is only possible after node 8 moves toward partition A as shown by figure 1c.

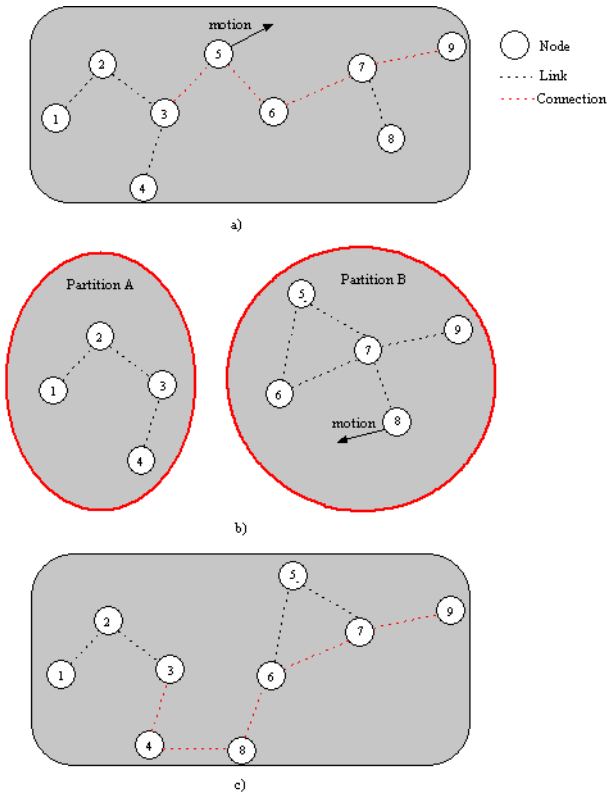

Figure 1 - Network partition:
a) Node 5 moves away from node 3.
b) No communication between the partitions.
c) Node 8 allows the reconnection.

Actually, the real impact of a *partition* on TCP performance depends on its duration. As we explain below, a long partition will trigger the TCP backoff mechanishm, and possibly end up increasing the delay to the connection restoration.

Figure 2 depicts an example of a long partition triggering the TCP *exponential backoff* mechanism. The purpose of this figure is only to aid our explanation and not take into account all the details actually involved which can be found in [St94]. Besides, for the sake of simplicity, we use the term *packet* instead of *segment* throughout this work. With that in mind, figure 2 shows how the delayed answer of the *exponential backoff* mechanism can lead the TCP sender to a long idle period, which we call "*dead time*", subsequently to the link restoration. The example shows that both packet 3 (P3) and the acknowledgment of packet 2 (ACK3) are dropped due to a link failure (red or left line). As the sender does not receive the confirmation of packet 2 (P2) receipt, it retransmits P2 by timeout after 6 s (6 s is the typical initial RTO which changes over time according to measured Round Trip Times - RTTs), and doubles its retransmission timeout (RTO) value. Whenever the timeout period expires, TCP sender retransmits P2 and doubles the RTO up to the limit of 64s that refers to the maximum timeout allowed (note that after 12 unsuccessful attempts TCP would give up). The example shows that shortly after triggering its timer for 64s the link is recovered, but it is too late for TCP and it will stay *over 1 minute frozen*, which is indeed a *dead time* for the assumed starving connection. In terms of percentage, we have roughly 61% (100s/

163s) of interruption due to link failure and the remaining 39% (63s/163s) is completely caused by TCP, which is certainly too much to be acceptable.

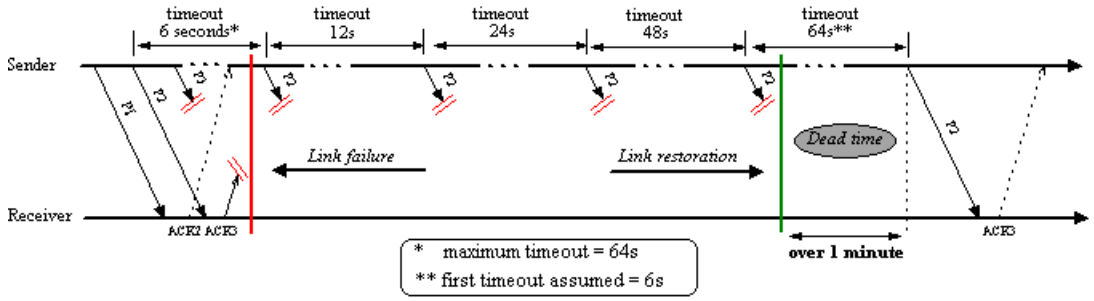

Figure 2 - Drawback of TCP *exponential backoff* mechanishm in ad hoc environment.

It is proper to mention that the infrequent (not necessary low) and transient packet losses, caused by either fading or not so long partitions, can also lead TCP mechanisms to take inappropriate actions. That is, even though such losses are improbably to cause long period of disconnection they do can undesirably lead TCP to invoke its *exponential backoff* mechanishm. As a result, TCP will be preventing the applications from using precious bandwidth by decreasing its transmission rate when it should not do that. More details about this sort of problems can be found in [CI94] where the focus is on cellular networks which can face identical trouble.

Therefore, based on what has been presented in this subsection, it is clear that the standard TCP needs to be adapted to work satisfactorily in this new paradigm. As stated before, an effective TCP algorithm must be capable of distinguishing the origin of a packet loss so as to take the most appropriate action. In fact, its error-detection mechanism needs to detect the nature of the error so that its error-recovery mechanism can be tailored for each specific case, as stated in [TM02].

2.2 Lower layers impacts on TCP

Given the fact that TCP is a reliable protocol providing end-to-end guarantees over a variety of local and unreliable protocols running in the lower layers, it is no surprise that its performance depends strongly on such protocols. What is not so clear, however, is how either TCP or lower protocols can be fine tuned to avoid as much as possible undesired interferences between them. In this subsection we present a brief insight into these subjects.

2.2.1 MAC layer impacts on TCP

Like the transport layer, the MAC layer also relies on error control mechanisms in order to improve the transmission efficiency. However, while the former deals with end-to-end recovery the latter concentrates on link (one hop) recovery. Hence, unless a well defined synchronism between these both protocols is put in place, interference can arise deteriorating substantially the end-to-end throughput provided by TCP.

The IEEE 802.11 “Distributed Foundation Wireless Medium Access Control” (DFWMAC) [std802.11,CWKS97] is the standard Medium Access Control (MAC) layer protocol, adopted for wireless mobile ad hoc networks. This MAC protocol, which defines both physical and link layer mechanisms, is intended for providing an efficient shared broadcast channel through which the involved mobile nodes can communicate. The main novelty of this protocol refers to the inclusion

of acknowledgment for data frames (link layer's ACKs) in addition to RTS/CTS (request-to-send/ clear-to-send) control frames to make it possible link layer retransmissions, as well as a *virtual carrier sense* mechanism to detect when the medium is busy.

In IEEE 802.11, for every link data frame to be sent a sequence of RTS-CTS-DATA-ACK frames needs to be exchanged between sender and receiver. Each of these frames carries the remaining duration for the full sequence to be transmitted, so that the other nodes can hear it and postpone their transmission accordingly. In fact, every node maintains an information parameter called *network allocation vector* (NAV) which is updated according to the other nodes transmission that it hears. In order to provide fair access to the medium, at the end of every such a sequence the nodes must await an *interframe space* (IFS) interval and then contend for the medium again[1]. The contention is carried out by means of a *binary exponential backoff* mechanism which imposes a further random interval aiming to avoid collisions and give to all nodes the same probability of gaining access to the medium. At every unsuccessful attempt this random interval tends to become higher (its range of randomness is doubled at every attempt), and after some number of attempts (typically 7 times) the MAC layer gives up and drops the data, which will be reported as a route failure to the network layer.

Therefore, this MAC protocol is certainly very powerful in dealing with collisions possibility in the wireless medium. By using short RTS/CTS control frames when attempting to reserve the medium it avoids bandwidth wastage in case of collisions since these frames are much smaller than data frames. Regarding the *virtual carrier sense* mechanism, it prevents the "well-known" *hidden node* [CGL00] problem where two hidden nodes from each other wish to communicate simultaneously with a third common node, which would result inevitably in collision. In such cases, under IEEE 802.11, the first node to succeed reserves the medium and the other (or others) becomes aware that it needs to await. Furthermore, although this MAC protocol was designed to provide a medium access control as fair as possible, it might also provide different levels of priority as long as distinct IFS intervals are used for certain flows. This feature is very appealing for QoS approaches [LAS01].

On the other hand, IEEE 802.11 relies on the assumption that every node can reach each other or at least sense any transmission into the medium, which is not always true in a mobile ad hoc scenario. Consequently, in some conditions the *hidden node* and *exposed node* problems can arise inducing *capture* effects [CGL00,TGC01], which can impair not only TCP throughput but also the fairness among simultaneous TCP connections. We explain in the following, by means of particular examples, how these problems can take place.

In figure 3 we consider a string topology in which each node can only communicate with its adjacent neighbors and for figures 3a and 3b only a single TCP connection. Suppose node 1 starts transmitting to node 5, as illustrated in figure 3a. After a while there might be a condition in which node 2 wishes to communicate with node 3 whereas node 4 is transmitting to node 5. As node 2 cannot hear node 4 transmission, it senses the medium idle (both physically and virtually from NAV) and so attempts its transmission by sending a RTS toward node 3.

1. In case the data frame is larger than a given threshold, it needs to be fragmented and only the first fragment is preced by a RTS/CTS frame exchange, the others are sent directly and using a short IFS in order to get precedence in accessing the medium. Short IFS is also used to send CTS and ACK with high priority due to obvious reason [CWKS97].

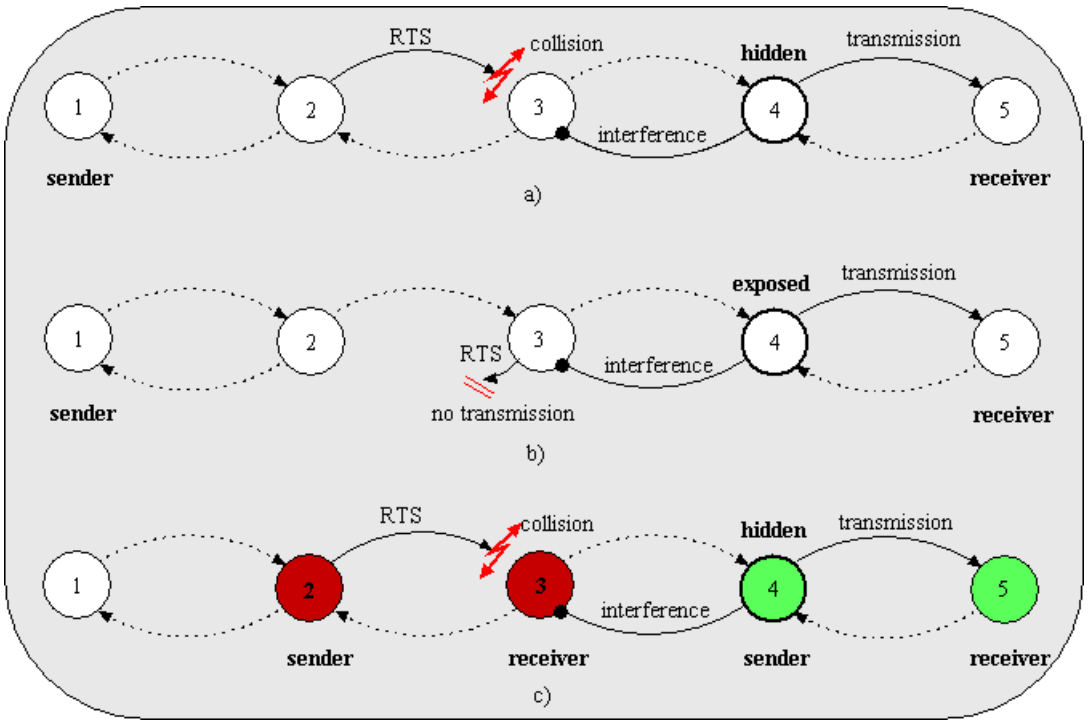

Figure 3 - MAC layer problems: a) hidden node, b) exposed node, c) capture effect

Nevertheless, since node 3 is within node 4 interference range (that is, node 4 transmission affects node 3 reception), it cannot receive any data which are dropped by collisions. This is a typical *hidden node* problem, where node 4 is the *hidden node* (in relation to node 2). figure 3b depicts a particular condition for *exposed node* problem, where node 3 has a data frame (that is related to a TCP ACK) to send to node 2. As node 4, which is within the sensing range of node 3 (that is, node 4 transmission affects node 3 transmission ability), is transmitting to node 5, node 3 must wait for the end of current transmission and then contend for the medium. Here node 4 is the *exposed node* (in relation to node 3). Note that collisions occurs only at the receiver, and so node 2 could receive the frame from node 3 correctly despite node 4 simultaneous conversation with node 5 since node 4 is out of interfering range of node 2.

Both mentioned problems can affect TCP throughput as follows: When a *hidden node* condition as illustrated in figure 3a takes place, the MAC layer of node 2 invokes its backoff mechanism which retransmits (locally) the lost frame a maximum number of times (typically 7). In case it does not succeed (e.g., due to high traffic between nodes 4-5), node 2 will drop the packet and send back a route failure message to node 1. Then the routing protocol in node 1 will attempt to find a new route to the destination, which will by itself delay the forwarding and in the worst case the TCP sender will time out and further delay the retransmission. Likewise, under the *exposed node* depicted in figure 3b as long as the traffic between nodes 4-5 is large enough to delay the pending TCP ACK beyond the TCP timeout interval, the TCP sender at node 1 will also back off by timeout. In reality, the TCP sender can miss not only delayed ACKs but also lost ACKs which can happen in the backward flow due to the same *hidden node* problem mentioned above. Specifically, TCP backward ACKs need to compete with the TCP forward data for the medium. How-

ever, in terms of link layer frames, there are many more data frames related to TCP forward packets than with TCP ACK packets because ACKs size are much smaller. So, the medium will be on average most loaded by TCP data related frames and as a result significant amount of ACKs will be lost.

Therefore, both *hidden* and *exposed node* problems can cause lack of ACKs at the TCP sender, which is characterized as a form of asymmetrical PATH problem for TCP as discussed later in subsection 2.2.3. Such a problem will either trigger a TCP retransmission by timeout when the lost ACK regards a confirmation of a successfully received data packet or impair the TCP *fast retransmit* mechanism for which three duplicate ACKs are needed to trigger a fast retransmission without being necessary to wait for the timeout event. Hence, in general it is expected that the more node a TCP connection needs to cross the lower end-to-end throughput can be achieved, since more contention for the medium will be necessary. In this way, [TG99,HV99] show that in fact TCP throughput over the IEEE 802.11 decrease severely and exponentially in function of number of hops, as depicted in figure 4.

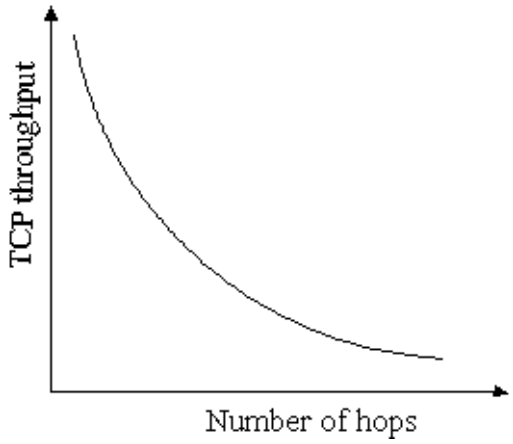

Figure 4 - TCP throughput decreases sharply with the number of hops.

Similar problems were evaluated in [XS01a,XSL01] where the authors show that using smaller values for both *packet size* and *maximum window size* in TCP setup can mitigate such problems in some extent. The idea behind this configuration regards the fact that the smaller these parameters the less traffic is sent at once into the network since TCP is an ACK-clocked based protocol whereby an ACK receipt is necessary to trigger the next transmission (or retransmission). As a result, the probability of collisions are decreased and the local MAC retransmission scheme has greater chance to succeed by attempting seven times.

In the cited references it was found that a *maximum window size* of four segments should be enough to provide maximum throughput in a stable way. Nevertheless, further analysis on that seems to be needed in order to validate such results for higher speed networks, such as the new IEEE 802.11a, where the limited size for that parameter might represent a throughput bottleneck.

Additionally, [XS01b] showed that TCP throughput in such a scenario can also be improved by using the *delayed ACK* option in which an ACK is sent for every two received data packets. In principle, this is a really interesting idea since it decreases the traffic inside this critical scenario.

IEEE 802.11 has also been raising serious *unfairness* concerns in ad hoc networks due to mainly the inappropriateness of its *binary exponential backoff* mechanism for this dynamic sce-

nario, which leads to *capture* behavior. In fact, such a phenomenon might be related to either *hidden node* or *exposed node* conditions as shown in [TCG01] and [XS01a], respectively.[1] For simplicity, we explain here only the situation in which the *hidden node* problem induces the *capture* effect. In figure 3c we assume that there are two independent connections, one between nodes 2-3 (connection 2-3) and another between nodes 4-5 (connection 4-5). Assuming that connection 2-3 experiences collisions due to *hidden node* problem caused by the active connection 5-6 as explained above, node 2 will back off and retransmit the lost frame. As explained earlier, at every retransmission the *binary exponential backoff* mechanism tends to impose an increasingly (although random) backoff interval, and it decreases the possibility of success for the connection 2-3. Besides, if the MAC retransmission scheme fails at all, the TCP will time out and also invoke its exponential backoff mechanism further increasing the delay for the next attempt. In consequence, the connection 2-3 will hardly obtain access to the medium and the another connection will *capture* it. Note that the MAC protocol is designed in such a way that if the connection between nodes 4-5 has a large data to transfer, it will fragment and transmit it in smaller data frames with higher priority over all the other nodes, which is done by using a short IFS between the transfer of each fragment. Clearly, it will also contribute to the unfairness behavior here shown.

Fairness problems due to capture conditions were investigated in [GBT99,TG99,TCG01], where the authors found that it can be mitigated by properly adjusting some MAC layer timers. Specifically, they show that regarding IEEE 802.11, better fairness can be achieved by increasing of the IFS interval (therein called *yield time*). However, it comes at the cost of the aggregate throughput, which is somewhat predictable once it will make the medium idle for longer time. Therefore, it is clear that alternative solutions for the inherently unfair behavior detected in this environment is really necessary. Furthermore, no solution appears to effective enough by simply configuring either TCP or MAC parameters. Rather, *hidden* and *exposed node* problems have to be deeper addressed toward a robust approach that would provide not only a fairer but also throughput effective MAC protocol.

Apart from what has been explained here with regards to MAC-TCP interactions, there are many other issues that could be discussed since they are potential source of complications for this scenario. For example, if the nodes have different interfering (and sense) ranges from the communication range as claimed in [XS01a] as being a standard design characteristic of nowadays wireless LANs, then the probability of mainly *exposed node* problem would be much higher. Likewise, either *hidden* or *exposed node* problems would be quite difficult to be controlled if the nodes had different battery power levels, which is quite likely to be the case in an actual network [PKD01]. Additionally, the inherently mobile scenario can arise synchronization issues, which would compromise the effectiveness of the reservation scheme provided by the MAC protocol. Hence, indeed the explained here represents only a brief insight into such a wide issue and more research on that is surely needed.

In short, this subsection showed that although the MAC layer problems are not directly linked to transport layer, they can indeed hurt the TCP performance. In particular, it was shown that smaller *maximum window size* should be used in order to minimize MAC collisions probability. Also, a greater idle interval between the sequence of frames (IFS) could be attempted as way of

1. In fact, those studies rely on different simulation models which is most likely responsible for some discrepancy found between them. However, both models seem to be consistent justifying credibility.

decreasing such probabilities. By doing so, most of the retransmission would be done locally at the MAC layer rather than by TCP in an end-to-end basis. Such a configuration illustrates how important the cooperation between both protocols is, in getting the most of the network. Further improvements, however, demand new protocol strategies.

2.2.2 Network layer impacts

As ad hoc networks consist of a highly dynamic environment, where frequent route changes are expected, routing strategies play a key role on TCP performance as well. Unlike the MAC layer, for which IEEE 802.11 protocol has been adopted as the standard, network layer has been subject of most research efforts on ad hoc networks area toward standardization. So, there have been a lot of proposed routing schemes for this scenario and generally each of them has different effects on TCP. In this subsection, we only address two of the major routing protocols since our purpose is exclusively to show how a network layer protocol can affect the TCP operation in this scenario, rather than approach several routing protocols.

Dynamic Source Routing (DSR): This protocol [JM96] operates on an on-demand basis in which a node wishing to find a new route broadcasts a *route request* packet. Then the destination node or any other node which knows a route to it response with a *route reply* packet. This packet informs the sender node the exact path to be followed by the data packets which are sent with a list of nodes through which they must pass inside their header. In addition, each node keeps a cache of routes it has learned or overheard[1]. As a result, the intermediate nodes do not need to keep an up to date table of routes, thereby avoiding periodic route advertisements that cause considerable overhead. The problem with this approach concerns the high probability of *stale routes* in this environment where mobility as well as medium constraints are normally present. That can happen, for instance, when a *route reply* message is in its way back to the sender but the replied route is no longer valid due to either an involved node that has moved away or a link that has somehow been interrupted. The problem is exacerbated by the fact that other nodes can overhear the invalid route reply and populate their buffers with stale route information. So, unless the stale route can be detected and recovered in a fast way, TCP can be led to backoff state, which will deteriorate its performance critically.

This problem was studied in [HV99a,HV99b] where the authors show that it can be mitigated by either manipulating TCP to tolerate such a delay or making the delay shorter so that the TCP can deal with them smoothly. In these studies they show that disallowing route replies from caches can improve route accuracy at the expense of the routing performance in terms of overhead, since every new route discovery implies in a new broadcast to be sent. On the other hand, they show that in general such an additional overhead is outweighed by the accuracy in the route determination, mainly for high mobility conditions, resulting in enhanced TCP throughput.

Temporally-Ordered Routing Algorithm (TORA): This is also an on-demand based protocol but has pro-active features as well. TORA [PC97] was designed to be highly dynamic by establishing routes quickly and concentrating control messages within a small set of nodes close to the place where the topological change happened. It is accomplished by maintaining multiple routes between any possible peers. In consequence, most topological changes should entail no reaction at all concerning route discovery, as it only reacts when all routes to a specific destination are lost. TORA makes use of directed acyclic graphs (DAG), where every node has a path to a given desti-

1. Overhear refers to the ability of a given node to hear other nodes transmission not addressed to it.

nation. In other words, all neighbors of a given sender have an alternative path to a given destination, which define multiple potential paths for every peer. The DAG is established initially by each node advertising a query packet and receiving update packets when it first tries to discovery a route. A new query is only necessary when no more routes are available in a given sender. This can happen since the invalid routes, caused by partition, are removed from the nodes by the affected node sending a clear packet. It is important to note that the philosophy of this protocol is to lessen the overhead inherent in frequent routing discovery procedure, and because of that route optimization (shortest path routing) is considered less relevant.

From the TCP viewpoint, this protocol can also suffer from stale route problem as explained above for DSR protocol. Nevertheless, since route discovery procedures are confined to situations less probable (no available path), such a drawback can be considered not too harmful to TCP. On the other hand, multiple path routing can indeed cause significant performance degradation to it. The problem occurs mainly because TORA does not prioritize shorter paths, which can yield considerable amount of out-of-sequence packets for the TCP receiver, triggering retransmission of packets. A typical situation could be to send an earlier packet through a longer path and then, due an instantaneous route problem, a new packet being forced through a shorter enough path to arrive first. Therefore, it should be interesting to have a self-adaptive mechanism to avoid this possibility. In section 4, we discuss this subject further.

In summary, the characteristics here presented of the two routing protocols reveal that the design of a route protocol should take into consideration its impacts on the upper layer. Once more, solutions can be placed in both layers and cooperation between them is very important.

2.2.3 PATH asymmetry

As TCP relies on time sensitive feedback information to perform its flow control, asymmetrical paths can seriously compromise its performance. The point is that, if TCP does not receive timely ACKs it cannot expand its CWND to make use of the full available capacity of the channel in place, thereby wasting bandwidth. Hence, in case the *forward path* characteristics are too different from the ones of the *backward path*, TCP will quite likely face performance problems.

There have been several studies on TCP asymmetry issues for satellite networks such as [DMT96,GD99,HK97], where this problem has been more notable. In this sort of network, high levels of asymmetry are quite common, since the satellites are optimized to transfer unidirectionally due to cost considerations. This is feasible to some extent because there are a number of applications that can tolerate reasonably well slower throughput in one direction of the connection, such as WEB, FTP, email, etc. For these applications, only the *forward path* (downstream) is really required to be fast, the returning packets can come through a slower path such as a modem line, for example.

In an ad hoc network, where the topology as well as the environment conditions can vary quite frequently and unpredictably, asymmetry can occur by different reasons, including lower layer strategies. Based on the work presented in [BP01,BPK97,KVR98,DMT96,VPI00], we classify the asymmetry in a tcp-based wireless mobile ad hoc network into the following classes:

- **Bandwidth asymmetry:** This is the classical asymmetry found in satellite networks in which forward and backward data follow distinct paths with different speeds. In ad hoc networks this can happen as well, since not necessarily all nodes have the same interface speed. So, even if a common path is used in both directions of a given flow, not necessarily they have the same

bandwidth. Besides, as the routing protocols can assign different paths for forward and backward traffics, asymmetry can definitely occur in ad hoc networks.

- **Loss rate asymmetry:** This sort of asymmetry takes place when the *backward path* is significantly more *lossy* than the *forward path*. In ad hoc environment this can be a serious factor as all links involved are wireless which is high error-prone and dependent on local constraints that can vary from place to place and also due to mobility patterns.
- **Media access asymmetry:** It can occur due to the characteristics of the shared wireless medium used in ad hoc networks. Specifically, as explained in subsection 2.2.1, in this kind of network TCP ACKs have to contend for the medium along with TCP data, and it may cause excessive delay as well as discarding on TCP ACKs.
- **Route asymmetry:** Unlike the three forms of asymmetry above, where *forward* and *backward path* can be the same, route asymmetry implies in distinct paths in both directions. Route asymmetry is associated with the possibility of different transmission range for the nodes in this scenario. In fact, the transmission range of each node depends on its instantaneous battery power level that, in most cases, is likely to vary over time. The inconvenience with different transmission range is that it can lead to conditions in which the forward data follow a considerably shorter path than the backward data (TCP ACK) due to lack of power in one (or more) of the nodes in the *backward path*. When a given node has low power to transmit, instead of directly communicating with the destination, it has to communicate through a multihop connection. However, as shown in subsection 2.2.1, multihop connections are prone to be low throughput effective. Consequently, the TCP ACKs may face considerable disruptions. Furthermore, mobility and variation in the battery power level make the problem even worse since they may cause frequent route changes.

In summary, all these form of asymmetry can induce lack of ACKs in the sender node, thereby impairing the forward throughput. The problem might be exacerbated when the ACKs arrive bunched up at the sender causing bursty traffic in the *forward path*, which is known as *ACK compression* phenomenon [KVR98]. Additionally, asymmetry condition can lead to inaccuracy in RTT estimation. As stated in [BP01,BPK97], potential solutions should either mitigate the low ACK flow problem or deal with that in a as effective as possible manner. Some proposed solutions are outlined in the following.

Based on the fact that most field of TCP header are unchanged in a stream of packets, *TCP header compression* [RFC1144] has been proposed to reduce the size of ACK packets in the *backward path*. However, as stated by [BP01] this option alone can be ineffective. The problem is that certain MAC protocols, as is the case with IEEE 802.11, present considerable overhead which impairs the enhancement provided by such a compression. Furthermore, despite being compressed, the packets still have to interact with other backward traffics anyway.

ACK filtering [BPK97,DMT96] is another approach that attempts to minimize the amount of ACK in the *backward path*. This scheme takes advantage of the fact that ACK packets are cumulative. Thus, when a new ACK is to be enqueued, the receiver node first verifies whether there are outstanding ACKs to be sent. If so, then only the latest ACK is sent while the others are discarded. State information need to be maintained for the connections with packet enqueued, which should be a disadvantage of this approach.

ACK congestion control and *ACK-first scheduling* schemes proposed in [BPK97,BP01] extend congestion control for ACKs packets and give priority for them over data packets, respectively. The former relies on RED mechanism [RED] for detecting congestion in the *backward path*,

which is signaled back to the sender that in turn slows down its transmission rate. The latter assumes that ACKs must be scheduled with high priority so that the sender does not starve.

The schemes above attempt to avoid lack of ACKs at the sender, but they do not guarantee that the sender will not starve of ACK. Thus, in order to mitigate the effects of reduced ACK feedbacks, even when some of the above schemes are in place, [BPK97,BP01] proposed the *TCP sender adaptation* and *ACK reconstruction* techniques. The former relies on the idea that the sender node should avoid slowing down in its CWND by considering the amount of data acknowledged by each ACK, instead of the number of ACKs themselves, to trigger CWND increases. In addition, the sender should estimate the admitted rate of the connection (CWND/SRTT) so that it never sent excessive bursts into the network. *ACK reconstruction* attempts to avoid standard TCP senders from being affected by reduced ACK frequency. This technique encompasses a soft-state agent (in the sender) called *ACK reconstructor* which receives the spaced ACKs from the receiver and paces its relaying to the receiver in a regulated rate. This rate is based on the amount of data acknowledged by the received ACK and also on the actual rate of the *backward path*. As a result, the CWND growing at the sender is controlled by the actual rate in the *backward path* and no burst behavior arises. More details on that are found on the mentioned references.

As mentioned above, Asymmetry can also induce inaccuracy on RTT estimation. TCP estimates its RTT based on measurements of the delay suffered by a packet in both direction of the flow. So, it can happen that such an estimation does not suit the actual *forward path* necessity, which might lead TCP to retransmit prematurely or too late.

In [PA99], *TCP Santa Cruz* is proposed as a solution for decoupling the CWND growing from the number of ACKs received. This proposal relies on measurements of relative delay that one packet experiences in relation to the previous packet, rather than on measurements of absolute delay for sampled packets, as standard TCP does. In this way, it pursues not only to provide enhanced RTT estimation but also to be resilient to ACK losses. In [VPI99], a somewhat related idea is proposed.

Thus, under the assumption that ad hoc networks will really experience asymmetrical paths, the above mechanisms as well as other related ones should be carefully investigated toward this challenging environment. A more detailed review on TCP along with asymmetry networks is given by [BPFS01].

3. Main existing TCP scheme proposals for ad hoc environment

In this section we present some schemes that have been proposed to improve TCP performance in ad hoc wireless networks. Indeed, not many solutions are found and the published ones have limitations that can compromise their widespread deployment. In the following we highlight the main features of each one.

3.1 TCP-Feedback

TCP-F [CRVP97] is a feedback based scheme in which the TCP sender can distinguish between route failure and network congestion by receiving Route Failure Notification (RFN) from intermediate nodes. The idea is to push the TCP into a “snooze state” when such messages are received. In this state the TCP stops sending packets and freezes all its variables such as timers and CWND size, which makes sense once there is no available route to the destination. Upon receipt of a Route Re-establishment Notification (RRN), via routing protocol, indicating that there is again an available path to the destination, the sender leaves the frozen state and resumes trans-

mission using the same variables values prior to the interruption. In addition, a *route failure timer* is used to prevent infinite wait for RRN messages. It is triggered whenever a RFN is received, and in case it expires the frozen timers are reset allowing the TCP congestion control to be invoked normally.

Results from this approach showed gains over standard TCP in conditions where the route re-establishment delay are high, which happens thanks to a fewer number of retransmission involved in it. Its performance was superior for scenarios with high rates as well. Nevertheless, the simulation scenario was quite simplified and so the results might not be much representative. For example, the RFN and RRN messages should be carried by the routing protocol, but no such a protocol was considered in the evaluation. Therefore, it is necessary more analysis toward this goal.

3.2 ELFN-based approach

In this approach [HV99a], TCP also interacts with the routing protocol in order to detect route failure and take appropriate actions when that is detected. This is done via Explicit Link Failure Notification (ELFN) messages that are sent back to the sender from the node detecting the failure. Such messages are carried by the routing protocol that needs to be adapted for this purpose. In fact, the DSR's route failure message was modified to carry a payload similar to the "host unreachable" ICMP message. Basically, the ELFN messages contain sender and receiver addresses and ports, as well as the TCP sequence number. In this way, the modified TCP is able to distinguish losses caused by congestion from the ones due to mobility.

When the TCP sender receives an ELFN message it enters a "stand-by" mode, which implies that its timers are disable and probes packets are sent regularly towards the destination in order to detect the route restoration. Upon receiving an ACK packet the sender leaves the "stand-by" mode and resume transmission using its previous timer values normally.

This scheme was only evaluated for the DSR routing protocol where the stale route problem, as stated in subsection 2.2.2, was found to be crucial for the performance of this modified TCP as well. Additionally, the length of the interval between probes packets and the choice of which sort of packet to send as a probe were also evaluated. Only the former showed to be really relevant. It suggests that a varying interval based on RTTs values could perform better than the fixed probe interval used in this algorithm. Another interesting investigation performed by this study was the impact of ARP protocol on TCP efficiency, which call for improvements.

In general this approach provided meaningful enhancements over standard TCP, as shown in the mentioned reference, but further evaluation is still needed. For instance, different routing protocols and under congestion conditions should be considered. Yet, more appropriated values for the probe interval should be determined.

3.3 Fixed RTO

This scheme [DB01] relies on the idea that routing failure recovery should be accomplished in a fast fashion by the routing algorithm. As a result, any disconnection should be treated as a transitory period which does not justify the regular *exponential backoff* mechanism being invoked, as this can cause unnecessarily long recovery delay. So, it disables such a mechanism when two successive retransmissions due to timeout happen, assuming it indicates route failure. By doing so, it allows the TCP sender to retransmit at regular intervals instead of at increasingly exponential ones.

In fact, the TCP sender doubles the RTO once and if the missing packet does not arrive before the second RTO expires, the packet is retransmitted again and again but the RTO is no longer increased. It remains fixed until the route is recovered and the retransmitted packet is acknowledged.

In [DB01] the authors evaluated this proposal considering different routing protocols as well as the TCP selective and delayed acknowledgements options. They report that significant enhancement were achieved using fixed-RTO with on-demand algorithms, and only marginal improvements were noticed regarding the TCP options mentioned. Nevertheless, as stated by the authors themselves, this proposal is limited to wireless networks only, which makes it somewhat discouraging as interoperation with wired networks seems to be really necessary in the future.

3.4 ATCP Protocol

Differently from the previously discussed approaches, ATCP [LS01] does not impose changes to the standard TCP itself. Rather it implements an intermediate layer between network and transport layers, similar to [BSAK95] considering cellular networks, in order to lead TCP to an enhanced performance and still maintain interoperation with non-ATCP machines. In particular, this approach relies on the ICMP protocol and ECN scheme to detect network partition and congestion, respectively. In this way, the intermediate layer keeps track of the packets to and from the transport layer so that the TCP congestion control is not invoked when it is not really needed, which is done as follows. When three duplicate ACKs are detected, indicating a *lossy* channel, ATCP puts TCP in "persist mode" and quickly retransmits the lost packet from the TCP buffer; after receiving the next ACK the normal state is resumed. In case an ICMP "Destination Unreachable" message arrives, pointing out a network partition, ATCP also puts the TCP in "persist mode" which only ends when the connection is reestablished. At last, when network congestion is detected by the receipt of an ECN message, the ATCP does nothing but forwards the packet to TCP so that it can invoke its congestion control normally.

This model was implemented in a testbed and evaluated under different constraints such as congestion, *lossy* scenario, partition, and packet reordering. In all cases the transfer time of a given file by ATCP yielded better performance comparatively to TCP. However, again the used scenario was somewhat special, since neither wireless links nor ad hoc routing protocols were considered. In fact, such experiments relied on a simple *ethernet* networks connected in series in which each node had two *ethernet* cards. Moreover, some assumptions such as ECN-capable nodes as well as sender node being always reachable might be somehow hard to be met. In case the latter is not fulfilled, for example, the ICMP message might not even reach the sender which would retransmit continuously instead of entering "persist mode". Also, ECN scheme deployment raises security concerns [ECN], and it might compromise the viability of this scheme.

3.5. TCP DOOR

This approach [WZ02] imposes changes to TCP code but does not require intermediate nodes cooperation, which represents its main differentiation from all four previously addressed proposals. It focuses on the idea that out-of-order (OOO) packets can happen frequently in this environment as a result of nodes mobility, and it might be enough to indicate link failure inside the network. In this way, TCP OOO detects OOO events and responds accordingly as explained below.

Based on the fact that not only data packets but also ACK packets can experience OOO deliveries, this algorithm implements a precise detection of such deliveries at both entities: sender and receiver. For this, additional ordering information is used in both kind of packets, which are conveyed as TCP options being one-byte option for ACKs and two-byte option for data packets. So, for every packet sent the sender increments its *own stream sequence number* inside the two-byte option regardless whether it is a retransmission or not (standard TCP does not increment sequence number of retransmitted packets). This allows the receiver to precisely detect OOO data packets and notify the sender via a specific bit into ACK packet. Additionally, because all ACKs associated with a given missing data packet have identical contents, the receiver increments its *own ack stream sequence number* inside the one-byte option for every "retransmitted" ACK, so that the sender can distinguish the exact order of *every* (retransmitted or not) sent packet. Therefore, the explained mechanisms provide the sender with reliable information about the order of the packet stream in both directions, allowing TCP sender to act accordingly.

After detecting OOO events, TCP sender can respond with two mechanisms: *temporarily disabling congestion control*, and *instant recovery during congestion avoidance*. In the former, TCP sender keeps its state variables constant for a while (T1) after the OOO detection. The rational is that such condition might be short (route change) not justifying the invocation of the congestion avoidance mechanism. In the latter, when an OOO condition is detected TCP sender checks if the congestion control mechanism has been invoked in the recent past (T2). If so, the connection state prior to the congestion control invocation is restored, since such an invocation may have been caused by temporarily disruption instead of by congestion itself.

In terms of evaluation, different scenarios combining all the mechanisms above mentioned were simulated. Also, the effects of the route cache property of DSR routing protocol on TCP DOOR performance were considered. The main results showed that: Only sender detection mechanism (ACK OOO detection) should suffice. Both responses mechanisms showed to be important and *instant recovery during congestion avoidance* performed better than *temporarily disabling congestion control*. The DSR route cache impaired the results. In general TCP DOOR improved TCP performance significantly, 50% on average.

Therefore, the overall published results were positive. Nevertheless, the assumption that OOO packets are exclusive results of route disturbance deserves much more careful analysis. As stated by the authors themselves, multipath routing algorithms (like TORA) can also induce OOO packets that are not necessarily related to route failures. Besides, as stated in section 2.2.3, diverse factors can cause *path asymmetry* inducing OOO events as well. On the other hand, the independence from intermediate nodes makes this proposal quite attractive which calls surely for further developments on it toward a more general approach.

3.6. Discussions

In summary, as the main drawbacks of the proposed schemes, the tree approaches that rely on feedback information from inside the network (TCP-F, ELFN-based, ATCP) may fail in situations where TCP sender is unable to receive data from the next hop node (e.g., due to mobility). In such cases, TCP sender would retransmit continuously instead of entering a frozen state. Furthermore, the usage of explicit notification by the intermediate nodes, such as ECN, can raise security concerns [ECN]. Fixed RTO scheme is completely inappropriate for possible future interoperation with wired networks. Also, the assumption that OOO packets are exclusive results of route disturbance deserves much more careful analysis.

In fact, the main concern addressed by the five approaches presented is how to avoid the *exponential backoff* mechanism when losses take place by factors other than congestion. However, as discussed in the previous section, other factors such as path asymmetry, and fine tuning with lower layers, among others, should also be considered in an effective approach design. Moreover, as explained in more details in the next section, challenging approaches should also address prominent issues like *power management*, *interoperation* with wired networks, *security*, etc. Thus, it is noticeable that the proposed approaches are somewhat limited, which highlights the necessity of further investigation in this area.

4. Directions toward a Robust Approach

According to what has been discussed in the preceding sections, it is clear that several issues remain to be addressed on the studied subject toward a robust approach. In reality, some of the ideas developed by existing proposed solutions may be improved, but new paradigms really need to be designed. We summarize here the main points to be tackled as well as some potential directions to be followed as a base for future researches.

From the existing proposals for improving TCP performance in ad hoc scenarios, presented in section 3, all of them except ATCP use the old value of the CWND after a route reestablishment. In ATCP, the authors argue that it is more appropriate to restart that, since the new route might have lower available bandwidth than the older one (too conservative approach). On the other hand, the other approaches justify the use of the old CWND value by the fact that the involved bandwidth is a scarce resource and should not be wasted (too aggressive approach). Therefore, it should be interesting to determine an optimal value between these two extremes. In principle, that may be done adaptively according to the new route features, using similar idea of the probe-based approach proposed in [ZT01].

Concerning the error-detection strategies used in each approach, they may be classified as *network detection* and *end node detection*. For example, TCP-Feedback, ELFN-based and ATCP approaches rely on network signaling for detecting path anomalies, and so they are related to *network detection* strategy. On the other side, both OOO and Fixed RTO approaches count on *end node detection*, as they do not need explicit information from inside the network. According to the previous section, each approach has its advantages and disadvantages, and the ideal should be to combine the advantages of each one. In terms of advantages, the *network detection* approach provides the end nodes with more accurate information since the intermediate nodes detect failures quicker than the end nodes would do by using the *end node detection* scheme. On the other side, *end node detection* does not need intermediate nodes cooperation, which is highly desirable for the sake of overhead, security and so on. Therefore, a hybrid approach based on a proper tradeoff appears to be quite encouraging.

The coexistence of TCP and MAC protocols was said (subsection 2.2.1) to improve by using either smaller values for *maximum window size* or larger IFS intervals in each of these protocols, respectively. In addition, there might be useful to investigate the possibility of increasing the maximum number of possible retransmission in the MAC layer as an attempt to rise the probability of success of the local retransmission scheme. However, as such an increase may trigger TCP timeouts, a compromise between these both factors has to be taken into account. In fact, approaches to improve MAC layer performance under TCP represent a very broad open area.

In terms of routing protocols effects on TCP, as stated in subsection 2.2.2, cache of routes as well as multipath routing strategies can disrupt TCP performance seriously. Up to now, evaluations of schemes based on cache of routes have revealed that the overhead of discovering new

routes at every enquiry is outweighed by the accuracy of the new routes achieved. So, it seems that such an approach (as originally proposed) is not viable from TCP perspective.

With regards to multipath routing strategy, further evaluation on it, toward improvements under TCP, appears to be needed. This approach is quite interesting because it makes it possible efficient bandwidth utilization by using several routes in parallel. Additionally, it may be quite useful in providing routing robustness by establishing redundant paths between sender and receiver. Thus, improvements on either network or transport layers in order to avoid TCP disruption due to out-of-order packets are desirable. At first, it should be possible to have a self-adaptive mechanism to avoid out-of-order packets effects at the receiver. For instance, TCP sender could delay, for a while, the response for a out-of-order packet in the hope that meanwhile an in-order packet arrives, similarly to what is done in [WZ02]. Standardized routing protocols for ad hoc networks are expected soon, and then specific evaluation on them will also be necessary.

Regarding *Path asymmetry*, presumably, its impacts on TCP depend a lot on the routing protocol strategy in place, among other factors as mentioned in subsection 2.2.3. Since this subject has not yet been addressed in ad hoc scenario, its quantitative analysis needs to be performed in order to clarify to what extent it impairs TCP. This will define whether specific mechanisms, such as the ones explained in the referred subsection, do need to be considered.

In terms of innovation, future TCP approaches should also handle additional features so as to make future implementations of ad hoc networks not only more efficient but also feasible. In the following we outline the main ones associating them, whenever possible, with existing schemes that can be taken as a starting point for new developments. Indeed, these factors represent a very wide open area as well.

Power management is a very important topic within ad hoc networks since they are supposed to be composed mostly of battery powered devices. So, it is increasing the interest for power aware approaches, but nothing notable has been done under TCP concern despite its necessity once TCP is not power aware. That is, TCP retransmits continuously regardless the destination is reachable, and it may cause considerable energy wastage. Hence, some effort in this field is highly required. At first, TCP could be fine tuned in order to stop transmitting when no connectivity to the next hop is available; which is possible by providing TCP with MAC layer signal strengthness information. Selective acknowledgement, like TCP SACK [RFC2018], might also be useful here in avoiding unnecessary retransmission upon a link recovery. Lastly, some options inherent in modern TCP versions can also be fine adjusted in order to meet this requirement, as shown in [AS01].

Interoperation between wireless mobile ad hoc networks and wired networks is another subject that has not yet been addressed at all (at least not from TCP perspective). However, it is high the probability that ad hoc networks will need to communicate with the Internet as a natural trend. Consequently, effective approaches have to take this issue into consideration as well. In this case, some concepts of cellular networks such as base station will have to be used and improved for this challenging environment. In particular, the multihop scheme in the wireless side poses new tough challenges for TCP congestion control which will need to distinguish between losses in both sides of the base station so as to take proper actions. In addition, *scalability* support will also be necessary as not only different but also of varying size networks can be in place. Therefore, future approaches should indeed address these both parameters. *Scalability* has been receiving a great deal of attention in the outstanding project [MICS], which can provide interesting insights into this issue.

Security considerations represents so far a hot issue in wireless environments, since this environment is much more susceptible to malicious users than the wired ones. Thus, many studies

have been carried out on this area, and in particular IPSEC [IPSEC] appears as a promising open standard for security in the Internet to be adopted in future implementations (including IPv6). In this case, however, approaches that rely on explicit feedback from intermediate nodes, like ECN, can face problems, since no direct access to the IP header is allowed for such nodes. Furthermore, ECN may conflict with IPSEC implementation. In order to mitigate such a problem, some effort has been put on that, but a really robust solution seems to be still missing. In other works, the proposed solutions will in the best case render further processing overhead. A detailed discussion about this issue can be found in [ECN,ECNSEC]. Therefore, alternative ways of detecting congestion inside the network, concerning TCP standpoint, need to be designed for this environment. In this sense, the idea of the congestion control mechanism of TCP Vegas [BP95] appears to be appropriate here, as it probes the network to detect congestion and adjusts its CWND accordingly, without requiring any specific support from intermediate nodes.

Finally, it is important to note that most of the issues presented in this section are interrelated, which can make their effective control quite difficult. In particular, appropriate metrics need to be determined, which in principle is not so easy due to the overlapping effects of some of the many factors involved.

5. Conclusions

We have presented an overview on the *state of the art* regarding TCP in mobile ad hoc networks. The issues here explained are very important and because of that we strongly believe that most of them should be taken into consideration in the design of future robust approaches. The paper focus on describing in details why TCP performs poorly in ad hoc environment and pointing out the potential directions whenever possible. Existing proposed solutions and the main remaining open issues are addressed as well.

The principal problem of TCP in this environment is clearly its inability to distinguish loses induced by mobility or *lossy* channel from the ones due to network congestion. We showed how the *exponential backoff* mechanism can lead TCP to idle states (*dead time* intervals), of over one minute long, subsequently to a partition recovery. It indicates that standard TCP is not only prone to poor throughput values but also to unacceptably long idle states.

From the lower layers analysis, we conclude that the MAC layer should be further improved in order to mitigate mainly *hidden* and *exposed node* problems in the context of a multihop network. These problems can impact TCP thoughput and also induce unfairness in the medium sharing. Some work has been done on that, in terms of proper adjustment of certain parameters on either TCP or MAC layer protocol, but more specific solutions are demanded. With regards to network layer, we showed that the routing protocols based on cache of routes and multipath paradigms need to be rethought taking into account TCP requirements in this environment. Otherwise, TCP has to be adapted to deal with their inherent problems. In general, routing protocols should avoid using information that are cached for too long and struggle to provide symmetrical and shortest paths. This would mitigate possible lack of ACKs at the transmitter and improve accuracy on RTT computations. The overall conclusion here is that protocol layers cooperation is essential in ad hoc network.

The few proposed solutions for TCP in ad hoc networks fails primarily in the following aspects. They do not consider scalability, including interoperation with fixed network, which in our view is quite necessary to leverage the developing of ad hoc networks. They also do not address power consumption issues despite its increasingly notable necessity. Assumptions such as sender node always connected to the next hop node, ECN-capable intermediate nodes and out-of-

order packets being always associated with mobility, demanded by some proposals, are quite restrictive. Cooperation from intermediate nodes can raise security concerns, as is the case with ECN standard, but it has not been considered either.

Therefore, all topics cited above make it clear that a lot of open issues remain to be addressed in this promising scenario. In particular, solutions encompassing heterogeneous network concepts seem to be of great merit. As an extension of this work we intend to carry out a quantitative analysis of the main issues here addressed.

References

[AS01] S. Agrawal and S. Singh. An Experimental Study of TCP's Energy Consumption over a Wireless Link. In *Proceeding of the 4th European Personal Mobile Communication*, February 2001.

[BB95] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *Proceedings of the IEEE ICDCS '95*, pages 136-143, 1995.

[BP01] H. Balakrishnan and V. N. Padmanabhan. How Network Asymmetry affects TCP. *IEEE Communication Magazine*, pages 2-9, April 2001.

[BP95] L. Brakmo and L. Peterson. TCP Vegas: End to End Congestion Avoidance on Global Internet. *IEEE Journal on Selective Areas of Communication*, October 1995.

[BPFS01] H. Balakrishnan, V. N. Padmanabhan, G. Fairhurst and M. Sooriyabandara. TCP Performance Implications of Network Path Asymmetry. *Internet Draft. draft-ietf-pilc-asym-05.txt*, June 2001.

[BPK97] H. Balakrishnan, V. N. Padmanabhan and R. H. Katz. The Effects of Asymmetry on TCP Performance. In *Proceedings of 3rd ACM/IEEE Mobicom conference,* September 1997.

[BS97] K. Brown and S. Singh. M-TCP: TCP for Mobile cellular Networks. In *Proceedings of the ACM SIGCOMM Computer Communication Review*, pages 19-43, 1997.

[BSAK95] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP Performance over Wireless Networks. In *Proceedings of the 1st ACM Int'l Conf. on Mobile Computing and Networking (mobicom),* November 1995.

[BV98] S. Biaz and N. H. Vaidya. Distinguishing Congestion Losses from Wireless Transmission Losses:A Negative Result. *IEEE 7th Int. Conf. on Computer Communications and Networks*, October 1998.

[BWK00] B. Bensaou, Y. Wang, and C. C. Ko. Fair Medium Access in 802.11 Based Wireless Ad-Hoc Networks. In *proceedings of 1st Annual IEEE and ACM International Workshop on Mobile Ad Hoc Networking and Computing*, August 2000.

[CGL00] A. Chandra, V. Gummalla, and J. O. Limb. Wireless Medium Access Control Protocols. In *IEEE Surveys and Tutorials, vol. 3, no. 2*, Second Quarter 2000.

[CI94] E. Caceres, L. Iftode. Effects of Mobility on Reliable Transport Protocols. In *proceedings of the 4th Intl. Conf. on Distributed Computer Systems,* June 1994.

[CRVP97] K. Chandran, S. Raghunathan, S. Venkatesan, R. Prakash. A Feedback Based Scheme For Improving TCP Performance In Ad-Hoc Wireless Networks. In *Proceedings of International Conference on Distributed Computing Systems- ICDCS '98. pp. 472-479,* 1997.

[CWKS97] B. P. Crow, I. Widjaja, J. G. Kim and P. T. Sakai. IEEE 802.11 Wireless Local Area Networks. *IEEE Communications Magazine*, September 1997.

[DB01] T. D. Dyer, R. V. Boppana. A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks. *ACM Symposium on Mobile Ad Hoc Networking & Computing - Mobihoc*, October 2001.

[DMT96] R. C. Durst, G. J. Miller and E. J. Travis. TCP Extensions for Space Communications. In *Proceedings of ACM MOBICOM*, November 1996.

[ECN] ECN (Explicit Congestion Notification) in TCP/IP. http://www.icir.org/floyd/ecn.html

[ECNSEC] IPSec Interactions with ECN. IETF draft-ietf-ipsec-ecn-02.txt.

[GD99] N. Ghani and S. Dixit. TCP/IP Enhancements for satellite Networks. *IEEE Communication Magazine*, July 1999.

[GTB99] M. Gerla, K. Tang, R. Bagrodia. TCP Performance in Wireless Multihop Networks. In *Proceedings of IEEE WMCSA '99*, February 1999.

[HDLP+02] Z.J. Haas et al. Wireless Ad Hoc Networks. Encyclopedia of Telecommunications, *John Proakis*, John Wiley, 2002.

[HK99] G. R. Henderson and H. Katz. Transport Protocols for Internet-Compatible satellite Networks. *IEEE journal on selected areas in communications*, February 1999.

[HV99a] G. Holland and N. H. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. *5th Annual International Conference on Mobile Computing and Networking - MOBICOM*, August 1999.

[HV99b] G. Holland and N. H. Vaidya. Impact of Routing and Link Layers on TCP Performance in Mobile AD Hoc Networks. In *Proceedings of IEEE WCNC*, September 1999.

[IPSEC] IP Security Protocol (ipsec). http://www.ietf.org/html.charters/ipsec-charter.html

[JM96] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *T. Imielinski and H. Korth, editors, Mobile computing. Kluwer Academic*. 1996.

[KVR98] L. Kalampoukas, A. Varma and K. K. Ramakrishnan. Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions. In *Proceedings of ACM Sigmetrics '98*, June 1998.

[LAS01] A. Lindgren, Andreas Almquist and Olov Schelén. Quality of Service Schemes for IEEE 802.11 - A Simulation Study. In *Proceedings of the Ninth International Workshop on Quality of Service (IWQoS 2001)*, June 2001.

[LS01] J. Liu, S. Singh. ATCP: TCP for Mobile Ad Hoc Networks. *IEEE Journal on selected areas in communications, vol. 19, No. 7*, July 2001.

[MICS] MICS - Mobile Information and Communication Systems. http://www.terminodes.com/

[MPC01] J. P. Macher, V. D. Park, and S. Corson. Mobile and Wireless Internet Services: Putting the Pieces Together. *IEEE Communication Magazine. Vol 39, No. 6, pages 148 -155*, June 2001.

[PA99] C. Parsa and G. Aceves. Improving TCP Congestion Control over Internets with Heterogeneous Transmission media. *IEEE Intl. Conf. on Network Protocols (ICNP '99)*, October 1999.

[PC97] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE INFOCOM'97 Conf.*, April 1997.

[PKD01] N. Poojary, S. V. Krishnamurthy and S. Dao. Medium Access Control in a Network of Ad Hoc Mobile Nodes with Heterogeneous Power Capabilities. In proceedings of ICC2001, June 2001.

[RED] RED (Random Early Detection). http://www.icir.org/floyd/red.html

[RFC1144] V. Jacobson. Compressing TCP/IP Header for Low-Speed Serial Links. *IETF RFC 2018*, February 1990.

[RFC2018] M. Mathias, J. Mahdavi, S. Floyd and A. Romanow. TCP Selective Acknowledgement Options. *IETF RFC 2018,* October 1996.

[RFC2501] Mobile ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. *IETF RFC 2501*, January 1999.

[RFC2581] M. Allman,V. Paxson and W. Stevens. TCP Congestion Control. *IETF RFC 2681*, April 1999.

[RPI00] V. Raisinghani, A. Patil and Sridhar Iyer. Mild Aggression: A new approach for improving TCP performance in asymmetric networks. *Asian International Mobile Computing Conference*, November 2000.

[St94] W. Richard Stevens. TCP/IP Illustrated Volume 1. Addison Wesley, 1994.

[std802.11] The Institute of Electrical and Electronics Engineers, inc. IEEE std 802.11 - *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. *The Institute of Electrical and Electronics Engineers, inc.*,1999 edition.

[TB00] V. Tsaoussidis and H. Badr. TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains. In *Proceedings of the 8th IEEE International Conference on Network Protocols,* 2000.

[TCG01] K. Tang, M. Correa and M. Gerla. Effects of Ad Hoc MAC Layer Medium Access Mechanisms Under TCP. In *Mobile Networks and Applications, vol. 6, n. 4*, August 2001.

[TG99] K. Tang and M. Gerla. Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks. In *Proceedings of IEEE MMT '99,* October 2000.

[TM02] V. Tsaoussidis and I. Matta. Open Issues on TCP for Mobile Computing. In *the Journal of Wireless Communications and Mobile Computing*, *Wiley Academic Publishers, Issue 2*, vol. 2, February 2002.

[TMP00] T. Goff, J. Moronski, and D. Phatak. Freeze-TCP: A True End-to-End Enhancement Mechanism for Mobile Environments. In *Proceedings of INFOCOM,* 2000.

[To01] C. K. Toh. Maximum Battery Life routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks. *IEEE Communications Magazine, **Vol 39, No. 6**, pages 138 -147*, June 2001.

[WZ02] F. Wang and Y. Zhang. Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response. In *Proceeding of Mobihoc'02*, June 2002. (Available at http://www.cs.utexas.edu/users/wangf/).

[XS01a] S. Xu and T. Saadawi. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? *IEEE Communications Magazine, Vol 39, No. 6, pages 130 -137*, June 2001.

[XS01b] S. Xu and T. Saadawi. Evaluation for TCP with Delayed ACK Option in Wireless multi-hop Networks. In *proceeding of Semiannual Vehicular Technology Conference (VTC2001 Fall)*, October 2001.

[XSL01] S. Xu, T. Saadawi and M. Lee. On TCP over Wireless Multi-hop Networks. In *proceedings of IEEE MILCOM2001,* October 2001.

[ZT01] C. Zhang and V. Tsaoussidis. TCP Real: Improving Real-time Capabilities of TCP over Heterogeneous Networks. In *Proceedings of the 11th IEEE/ACM NOSSDAV 2001*, New York, 2001.