# Experimental Analysis of TCP Performance in Static Multi-hop Ad Hoc Networks

**Giuseppe Anastasi, Emilio Ancillotti** **Marco Conti, Andrea Passarella**

*Pervasive Computing & Networking Lab. (PerLab)*

Dept. of Information Engineering
University of Pisa, Italy
{firstname.lastname}@iet.unipi.it

CNR-IIT
National Research Council, Italy
{firstname.lastname}@iit.cnr.it

## Abstract

Many previous papers have pointed out that TCP performance in multi-hop ad hoc networks (MANETs) is sub-optimal. This is due to several TCP design principles that reflect the characteristics of wired networks dominant at the time when TCP was designed that do not hold in MANETs. Based on this evidence, several TCP variants have been proposed in the literature. However, little effort has been devoted to investigate the performance of TCP in a real environment, even in a static scenario. Most of the work relies on simulation. In this chapter we provide an experimental analysis of TCP in static multi-hop ad hoc networks. We investigate the TCP performance in a simple, but interesting, scenario, i.e., a chain topology with different number of hops. We highlight some results contrasting with simulations and show that these discrepancies are due to the different protocols -- or different protocol implementations -- used in practice with respect to simulation tools.

## 1. Introduction

TCP (Transmission Control Protocol) is the *de facto* standard for reliable connection-oriented transport protocols, and is normally used over IP (Internet Protocol) to provide end-to-end reliable communications to Internet applications. Although TCP is independent from the underlying network technology, some assumptions in its design are clearly inspired from the characteristics of wired networks dominant at the time when it was conceived. TCP implicitly assumes that nodes are static (i.e., they do not change their position over time), and packet loss is almost always due to congestion phenomena causing buffer overflows at intermediate routers. These assumptions do not hold in MANETs as the network topology may change due to node movements and failures (e.g., because battery is exhausted). In addition, packet losses due to buffer overflow are rare event in MANETs while losses due to link layer contention are largely dominant [Fu03][1].

Many papers have pointed out that the drastic difference between MANETs and the legacy Internet may lead to poor performance of TCP over MANETs. Based on this observation, researchers have proposed (and are still proposing) TCP or 802.11 modifications aimed at addressing this problem (see Section 2 for a detailed discussion on the literature).

Almost all these studies rely on simulation, and many of them do not consider some important details (e.g., the routing protocol is often omitted). To the best of our knowledge, very few experimental analyses have been carried out so far [Gu04, Kaw05]. On the other side, previous

[1] Packet losses due to transmission errors are recovered through link-layer retransmissions.

[4] The threshold is given by ALLOWED_HELLO_LOSS*HELLO_INTERVAL, where HELLO_INTERVAL is the time interval (in seconds) between successive HELLO messages sent by the same node, while ALLOWED_HELLO_LOSS is the number of HELLO messages that must be lost before assuming that a link failure has occurred.

experimental studies have shown that certain aspects of real MANETs are often not effectively captured in simulation tools [Ana04]. Furthermore, available software and hardware products often use parameters settings different from those commonly assumed in simulation tools. Finally, real operating conditions are often different from those modeled in simulation experiments. For example, interferences caused by WiFi hotspots or other devices in the proximity are inevitable in practice. For all the above reasons, we believe it is of great importance to measure TCP performance in a real environment.

In this chapter we provide an experimental analysis of TCP over an IEEE 802.11 multi-hop ad hoc network in an indoor environment. For the sake of simplicity, and for better comparison with previous simulation results, our analysis is limited to static networks with a chain topology and a limited number of hops. However, our experimental testbed is made up with off-the-shelf products that are largely used within the community. Specifically, we consider two very popular routing protocols, i.e., AODV [Per03, AodvUU] and OLSR [Ton04, Cla03] which take a different approach on building routes (reactive vs. proactive). In addition, in our experiments we consider TCP NewReno, available with Linux distributions.

Our experimental outcomes are normally aligned with simulation results. However, we also found some results contrasting with simulation. We discovered that such discrepancies are due to different protocols – or protocol implementations – used in practice with respect to common simulation tools. For example, we have found that the TCP delayed ACK policy implemented in common simulation tools is not completely compliant with standard Linux implementations when the congestion window size is limited to very small values. We show that in the real world this moves the TCP optimal operating point with respect to what can be measured by simulation. Also, we have found a significant performance difference when AODV uses HELLO messages to detect link failures (which is the standard in the real implementation)with respect to the case when AODV relies on link-level notifications (which is the standard in the simulation tools).

The main contribution of this chapter is therefore to advocate the use of real experimentation when evaluating TCP performance over MANETs. While we acknowledge the importance of simulation in this field, we believe that, whenever feasible, a continuous verification of simulation results against real-world measurements is necessary to gain clear understanding of the TCP behavior.

The rest of the chapter is organized as follows. Section 2 is devoted to related work. Section 3 describes the testbed we relied upon for our experimental analysis, the methodology we used, and the performance indices we measured. Section 4 discusses the results obtained. Finally, Section 5 concludes the chapter.

## 2. Related Work

In the last years several papers have analyzed the TCP performance over MANETs. Most of them are targeted at demonstrating that TCP exhibits poor performance in MANETs. Typically, these papers propose enhancements to the standard protocol to improve performance. A comprehensive survey on TCP developments for multi-hop ad hoc networks is available in [Pap05]. In addition, some reliable transport protocols designed from scratch and optimized for the MANET environment have also been proposed in [Sun03, Ana05].

A lot of papers have pointed out that node mobility may severely degrade the TCP performance [Ahu00, Cha01, Dye01, Liu01, Hol02, Fu02, Sun03] due to the protocol inability to manage efficiently mobility effects. Node movements may cause route failures and route changes which results in packet losses and delayed ACKs at the sender side. TCP misinterprets these events as a sign of congestion and activates the congestion control mechanism. This leads to unnecessary retransmissions and throughput degradation [Ana05]. In addition, mobility may exacerbate the unfairness between competitive TCP sessions [Tan99].

Other papers have addressed TCP performance in static MANETs. In a static environment the maximum achievable throughput is limited by the interaction (at the MAC level) between neighboring nodes [Li01]. According to the IEEE 802.11 MAC protocol, each node must sense the medium before starting transmissions. In addition, interferences may cause collisions at the destination node. Hence, it can be shown that in a string (or chain) topology, like the one shown in Figure 1, the expected maximum bandwidth utilization is only 0.25 [Li01]. However, the 802.11 MAC protocol is not able to find the optimum schedule of transmissions by itself. In particular, in a chain topology it happens that nodes early in the chain starve later nodes (similar remarks apply to other network topologies as well). Thus, in practice performance is even worse than expected.


**Figure 1. A MANET with chain topology.**

The above limitations are inherent to the characteristics of multi-hop ad hoc networks, and cannot be accounted to the TCP protocol. However, the interaction between TCP mechanisms (mainly the congestion control algorithm) and MAC-layer issues (hidden/exposed node problem, exponential backoff scheme, etc) may lead to several, unexpected, serious instability and fairness problems in some specific scenarios, as shown in [Xu01, XuS02, XuB02].

The TCP congestion window size is also responsible for suboptimal performance in almost every scenario which may result in throughput degradation and instability [Xu01, Fu03]. In [Fu03] it has been shown that, for a given network topology and traffic pattern, there exist an optimal value of the TCP congestion window size at which the TCP throughput is maximized. However, TCP does not operate around this optimal value and typically grows its average window size much larger, leading to decreased throughput (throughput degradation is in the order of 5-30% with respect to the optimal case) and increased packet losses. The very reason for this suboptimal behavior is the origin of packet losses. Unlike traditional wired networks, in MANETs packet losses caused by buffer overflows at intermediate nodes are rare events, while packet losses due to link-layer contention are largely dominant.

An interesting issue is, thus, how to select the maximum congestion window size to achieve optimal throughput. Several papers have faced this challenge. They mainly refer to chain topologies for the sake of simplicity. [Fu03] and [Li01] suggest setting the maximum window size to ¼ of the chain length (i.e., $h/4$ if $h$ is the number of hops). The rule is based on consideration about spatial reuse. In [Che03, Che04] the problem of properly setting the maximum congestion window is bounded to the bandwidth-delay product. The authors provide a systematic solution to this problem by proposing an adaptive mechanism that sets the maximum congestion window according to the hop count of the TCP connection. An alternative scheme (SCA) that provides an effective spatial reuse without limiting the maximum window size is proposed in [Pap04]. SCA increases the sending rate during the congestion avoidance phase more slowly than in the legacy protocol, so as to reduce the number of on-the-fly packets. Simulation results show that the SCA mechanism stabilise the sending window to a relatively small value.

Most of the analyses on TCP performance, especially under static conditions, do not consider any specific routing protocol. On the other hand, the effects of routing protocols have been investigated in [Ahu00, Dye01, Osi06]. In [Ahu00] four different routing protocols are considered: the Ad hoc On-demand Distance Vector (AODV [Per03]) protocol, the Dynamic Source Routing (DSR [Joh04]) protocol, the Destination-Sequenced Distance Vector (DSDV [Per94]) protocol, and the Signal Stability-based Adaptive (SSA, [Dub97]) protocol. In [Dye01] the authors consider two on-demand routing protocols (DSR [Joh04] and AODV [Per03]), and an adaptive proactive protocol

(ADV [Bop01]). The simulation results show that ADV maximizes the TCP performance under a variety of conditions. In [Osi06] the authors quantify the TCP performance degradation caused by the underlying routing traffic. Based on simulation measurements they determine the *admissible operation range* where the level of such degradation can be still acceptable for end users.

As there are different versions of the TCP protocol around (Tahoe, Reno, NewReno, SACK, Vegas, etc.), many authors have compared the performance of different TCP versions, mainly in terms of throughput and fairness [Xu01, Rak05, Kim05]. Conclusions are however contrasting. From [Rak05] it appears that in static MANETs with AODV routing protocol TCP-Vegas outperforms TCP-NewReno both in terms of maximum achievable throughput and fairness. However, [Kim05] shows that the performance of TCP-NewReno and TCP-Vegas depend on the underlying routing protocol. TCP-NewReno is actually less efficient than TCP-Vegas on top of AODV, but outperforms TCP-Vegas when using OLSR [Kim05].

Almost all the papers cited above rely on simulation. Real testbeds are seldom used and, in many cases, their use is aimed at validating simulation results. Recently, an experimental analysis in static conditions has been carried out in [Kaw05]. TCP is therein evaluated in chain and cross topologies with different number of hops. Performance metrics include throughput, average delay and delay standard deviation (jitter) experienced by TCP segments. TCP performance is investigated by varying three different parameters: RTS/CTS mechanism (enabled/disabled), Selective ACK (enabled/disabled), and congestion window size. The maximum congestion windows size is either unclamped or limited to *3/2 h*, where *h* is the number of hops between the sender and the destination node. It is not clear why they limit the maximum size to *3/2 h* (previous simulation analyses would suggest different values [Li01, Xu01, Fu03, Che04]). Finally, no routing protocol is used.
Another experimental analysis is reported in [Gu04] where the authors evaluate the impact of Rate Based Pacing (RBP, [Agg00]) on TCP performance. They also consider chain topologies with different number of hops, and use (a modified version of )AODV as the routing protocol. However, they do not evaluate the effects of the routing protocol on TCP performance. In addition, they assume unclamped congestion window size, and consider only a single TCP flow. Their experimental results show that TCP-Reno outperforms TCP RBP.

In our work we evaluated the effects of the routing protocol by considering two different routing protocols (i.e., AODV and OLSR). In addition, we considered different values for the congestion window size and evaluated the optimal window size in a real environment.

## 3. Experimental Environment

### 3.1. Testbed description

Our testbed consisted of *IBM R-50* laptops equipped with integrated *Intel Pro-Wireless 2200* wireless cards. All the laptops were running the Linux Kernel 2.6.12 with the latest available version of the `ipw2200` driver (1.1.2). Wireless cards follow the IEEE 802.11b specifications. We decided to limit their maximum bit rate to 2 Mbps to compare experimental results with simulation results available in the literature. The RTS/CTS was active and the threshold was set to 100 bytes so that RTS/CTS handshake was enabled for TCP data segments and disabled for TCP acknowledgments. The transmission power of the wireless cards was set to the minimum allowed value (-12db) so as to reduce the transmission range and make possible to perform real multi-hop experiments in an indoor environment.

Figure 2 shows the indoor environment where the experiments were carried out. It is a real working environment with offices and labs. In particular, there are several WiFi Access Points in the proximity. Although this environment influenced significantly our performance measures, we

believe that it is important to test the TCP performance in a real working environment. In our experiments we considered a chain topology with five nodes deployed as in

Figure 2. In all the experiments node N1 was the sender, while the receiver (and the number of active nodes) depended on the specific scenario. Specifically, we considered four different scenarios with hop count ranging from 1 to 4. For example, in the 3-hop scenario, node N4 was the receiver (node N5 was not active). The distance between nodes was chosen in such a way that only adjacent nodes were within the transmission range of each other.

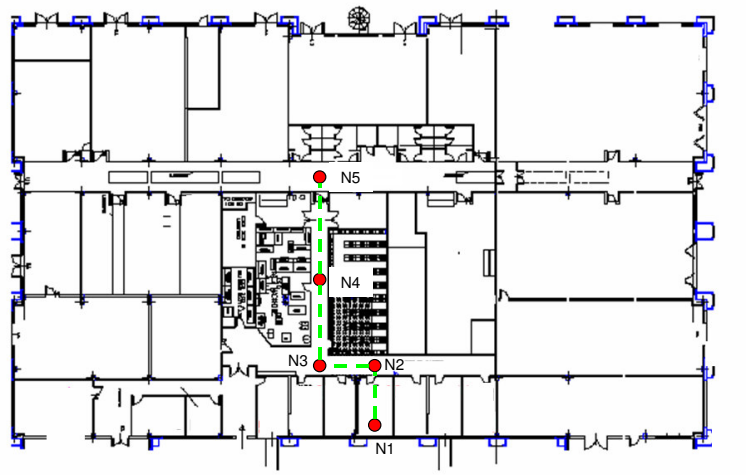


**Figure 2. Indoor environment and network topology used in our experiments.**

We used ftp-like traffic, i.e., the sender node had always a packet ready to send. To this end, we developed a simple client/server application using Linux sockets. At the server side we used the `TCP_WINDOW_CLAMP` socket parameter to bind the size of the advertised window to the desired maximum value. The segment size was constant in all the experiments with the transport-layer payload size set to 1460 byets. To capture TCP segment we used `tcpdump`, while to analyze the experiments results we used `tcpstat` and `tcptrace` (enhanced by our shell scripts).

As anticipated, we considered two different routing protocols, i.e., AODV and OLSR. AODV (Ad hoc On-demand Distance Vector, [Per03]) is a well-known reactive protocol that uses RREQ, RREP and RERR messages to discover and maintain routes to the destination, and can use two different mechanisms for neighbour discovery and local connectivity maintenance, i.e., link layer information provided by the underlying MAC protocol, or HELLO messages that are periodically exchanged between all nodes in the MANET. In our experiments we used the AODV implementation for Linux by the Uppsala University (AODV-UU [AodvUU]) version 0.9.1. To maintain one-hop connectivity we used HELLO messages since the `ipw2200` driver doesn't provide link-layer failure information. All the AODV parameters were set to their default values.

OLSR (Optimized Link State Routing, [Cla03]) is an optimization of the classical link state algorithm for mobile ad hoc networks (it is thus a proactive protocol). OLSR periodically floods the network with route information so that each node can build locally a routing table containing the complete information of routes to all possible destinations within the ad hoc network. Similarly to AODV, OLSR employs a neighbour discovery procedure based on HELLO messages. In our testbed we used the OLSR_UniK implementation for Linux version 0.4.10 [Tøn04]. We set all the parameter values to their default values. We only disabled the OLSR hysteresis process because it was shown to degrade TCP throughput in a not acceptable way [Anc06].

### 3.2. Performance measures

In our analysis we considered the following two performance measures.

- *Throughput*, i.e., the average number of byte successfully received by the final destination per unit time.
- *Retransmission index*, i.e., the percentage of segments re-transmitted by the sender TCP.

The *throughput* was measured at the application layer as the number of bytes (successfully) received by the destination process in a given time interval, divided by the duration of the time interval.

The *re-transmission index* (*rtx*) was obtained as

$$rtx = \frac{\#\ of\ segments\ \ retransmitted\ by\ the\ source}{\#\ of\ non-duplicated\ segments\ \ successfully\ received\ by\ the\ destination}$$

The *re-transmission index* allows us to evaluate the ability of TCP to handle transmission in an efficient way. It is worthwhile to emphasize that re-transmitted segments consume energy both at the sender and intermediate nodes. As nodes in a MANET may have limited power budget, it is important to manage (re-)transmission efficiently. Therefore, a small value for the re-transmission index is highly desirable.

### 3.3. Methodology

When dealing with real testbeds one of the main difficulties is that experiments cannot be repeated exactly in the same way as external conditions may vary from time to time -- sometimes during the same experiment -- and there is definitely no control on them. Therefore, successive experiments carried out under the same operating conditions may provide outcomes that differ significantly from each other. In addition, comparison of performance measurements obtained in different scenarios or operating conditions becomes hard or even impossible. To achieve more statistical accuracy, we replicated each experiment 5 times, and averaged the performance measures over the entire set of 5 replicas. In addition, experiments with similar parameter values (e.g., with different maximum *cwnd* size but with all other parameters set to the same values) were performed in an interleaved way. For instance, we performed, back to back, the first replica of experiments with maximum *cwnd* size equal to 2, 3, 4, and 32, respectively. Then, we performed the second replica of each experiment, and so on.

In the next section, we always make reference to average values. In addition, in many cases we also show the maximum and minimum values measured in the 5 replicas. Each replica was 120 s long, and consisted of a file transfer. To perform multiple replicas the whole process of experimentation (data generation, logging and archiving) was automated using shell scripts.

## 4. Experimental Results

In this section we describe the results obtained from our experiments in different scenarios. Specifically, we considered four chain topologies with different number of hops (from 1 to 4) as shown in Figure 2. In the first set of experiments we considered AODV as the routing protocol, and assumed that there is single TCP flow in the network. The purpose was to investigate the influence of the maximum congestion window size on TCP performance, and compare our experimental results with previous simulation results. Then, we extended our analysis with AODV routing protocol by considering the effects of interfering traffic. Finally, we repeated the above experiments by using OLSR instead of AODV.

### 4.1. Influence of the maximum congestion window size

To evaluate the influence of the maximum congestion window (*cwnd*) size we clamped the congestion window to some specific values. Specifically, we considered maximum *cwnd* sizes of 2,

3, and 4, and performed also experiments where the window size is unclamped. In the plots below the latter case is referred to as window size equal to 32.

Previous simulation studies [Che04, Fu03] have shown that in our scenarios, the optimal value for TCP *cwnd* is 2. However, previous studies do not highlight the behavior of TCP over OLSR for varying *cwnd* size, nor the performance of TCP over AODV when Hello Messages are used to discover neighbor nodes. Therefore we evaluated through ns-2 [Ns-2] the optimal value of TCP *cwnd* in these configurations, using the same operational parameters used in experimental analysis. Figure 3 shows that also our simulative analysis indicates that 2 is the optimal value for TCP *cwnd*.

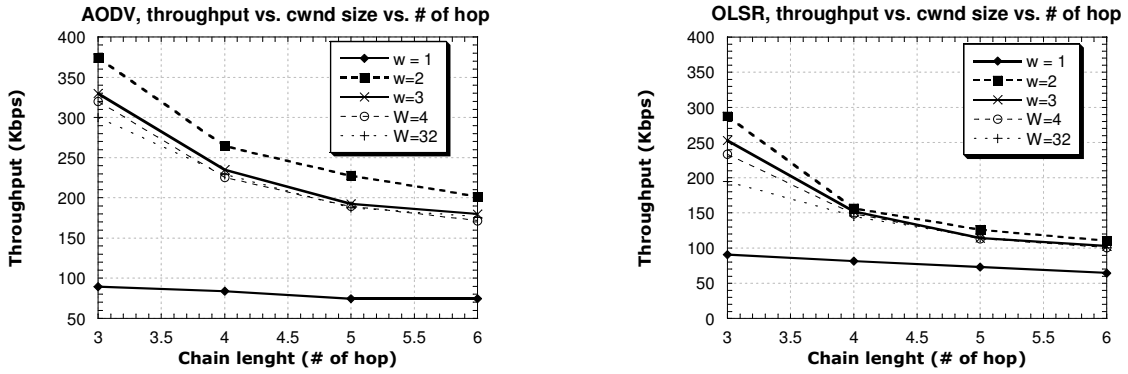


**Figure 3. Throughput over AODV (left) and OLSR (right) vs. number of hops vs. *cwnd* size. NS-2 results.**

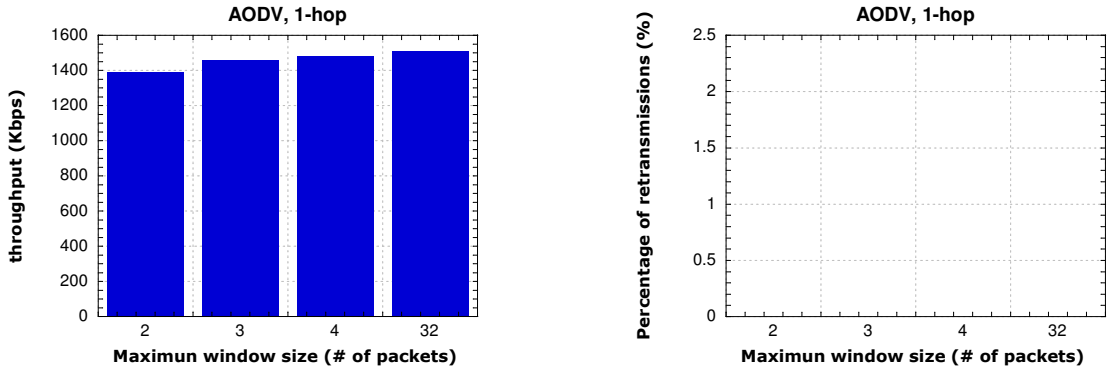


**Figure 4. Throughput (left) and percentage of retransmitted segments (right) vs. maximum congestion window size in the 1-hop scenario. The routing protocol is AODV.**

The results obtained in our experimental analysis, both in terms of throughput and percentage of re-transmissions, are shown in Figure 4 through Figure 7. We found that in our experiments a maximum *cwnd* of size 2 never provides optimal performance. Specifically, in the 1-hop scenario an unclamped congestion window seems to be the best choice. This is because in the 1-hop scenario there is no competition between neighboring nodes as in multi-hop scenarios. In the other scenarios, a *cwnd* limitation appears to be beneficial but the optimal *cwnd* size appears to be 3 (in the 2-hop scenario the throughput with maximum *cwnd* equal to 4 is slightly better, but the re-transmission is significantly higher).

This discrepancy with previous simulation results is due to a different behavior between the TCP version implemented in the Linux distribution used in our testbed (Linux Kernel 2.6.12) and the one implemented in common simulation tools (e.g ns-2 [Ns-2]) when the maximum *cwnd* of size is set to 2. By a detailed analysis of traces we found that the simulated TCP receiver sends back one acknowledgement *every other* segment, while the real (i.e., Linux) TCP receiver sends back one

acknowledgement *every* segment. When the maximum *cwnd* size is 3 (or larger) both the real and simulated TCP send back one acknowledgement *every other* segment. The increased number of acknowledgments managed in the real testbed when the maximum *cwnd* size is equal to 2, makes the throughput suboptimal.

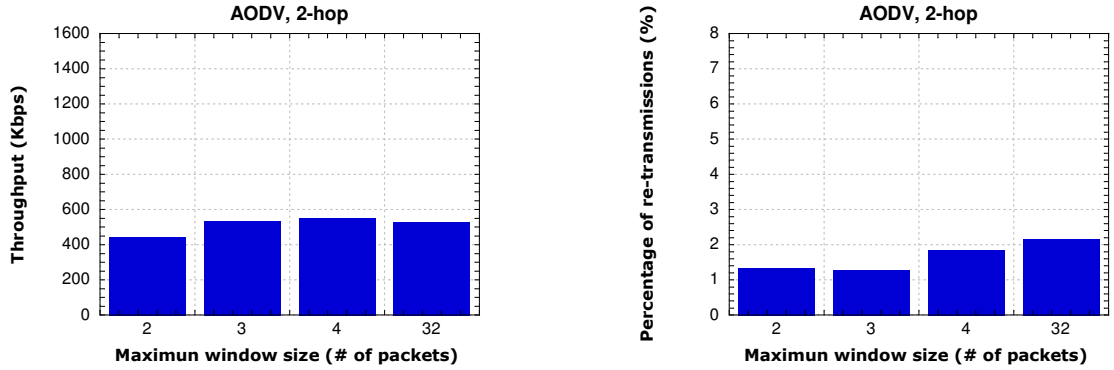


**Figure 5. Throughput (left) and percentage of retransmitted segments (right) vs. maximum congestion window size in the 2-hop scenario. The routing protocol is AODV.**

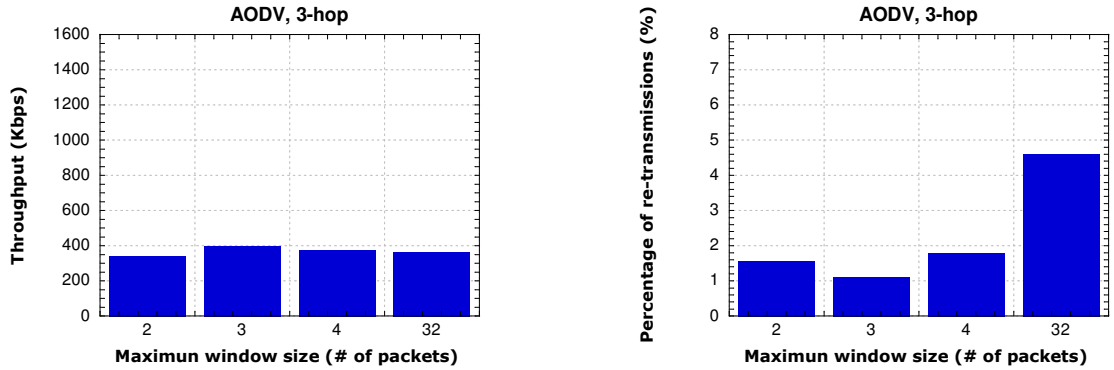


**Figure 6. Throughput (left) and percentage of retransmitted segments (right) vs. maximum congestion window size in the 3-hop scenario. The routing protocol is AODV.**

To confirm our conclusion we used the ns-2 simulation tool [Ns-2], and ran simulation experiments where we modeled the above conditions (note that we used the same AODV-UU code both in the real and in the simulated experiments). Specifically, we set the delayed ACK option at the receiver TCP when the maximum *cwnd* size was equal to 3, 4 and, 32, respectively. Instead, we disabled this option in case of maximum *cwnd* size was equal to 2. Therefore, in the latter case the TCP receiver sends back one TCP ACK *every* segment received , while in all other cases it sends one TCP ACK every other segment. We observed that the optimal window was 3 as in the real experiments.

From the above results it also appears that TCP throughput with optimal *cwnd* size is not so different from that with unclamped congestion window. This is in contrast with previous simulation studies which observe a significant throughput improvement with optimal *cwnd* size. This discrepancy can be explained in terms of the mechanism used by the AODV routing protocol for detecting link failures. More precisely, the link failure detection mechanism based on HELLO messages generates frequent route failures with associated throughput oscillations and performance degradation. This issue is described in detail in the next section.

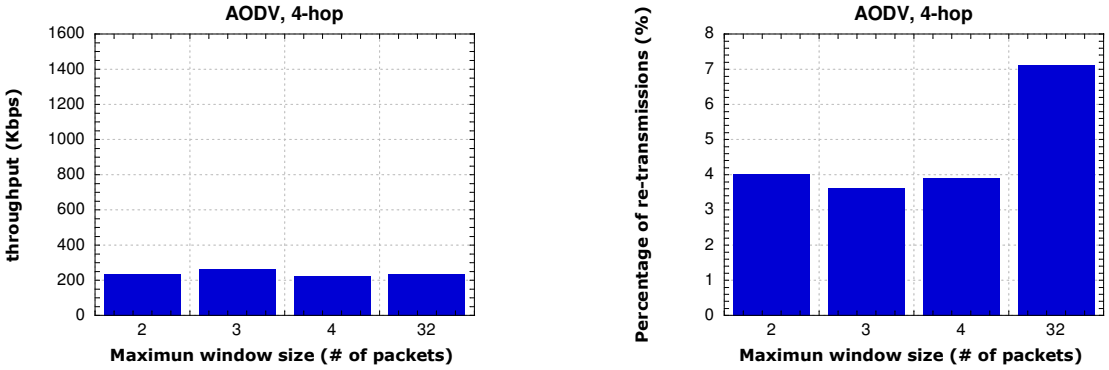


**Figure 7. Throughput (left) and percentage of retransmitted segments (right) vs. maximum congestion window size in the 4-hop scenario. The routing protocol is AODV.**

### 4.2. Influence of HELLO messages

AODV may take two different approaches for link failure detection. It can either exploit link failure notifications from the underlying layer (provided that this service is available), or rely upon a periodic exchange of HELLO messages. In the former case AODV learns that a link failure has occurred as soon as it receives an explicit notification from the underlying layer (hereafter, this approach will be referred to as AODV-LL). In the latter case each node listens for HELLO messages that are periodically broadcast by each other node in the network. A node assumes that a link failure has occurred if it has previously received a HELLO message from a neighbor and, then, for that neighbor does not receive any packets (HELLO messages or anything else) for more than a predefined threshold[4] (hereafter, this approach will be referred to as AODV-HELLO). In the AODV-UU implementation only HELLO messages and AODV control messages (e.g., RREQ and RREP) are considered for neighbor connectivity assessment (i.e., data messages are not taken into consideration).

The AODV protocol in our testbed uses HELLO messages since the ipw2200 driver (version 1.1.2) does not provide link failure notifications. Assuming default parameter values (`HELLO_INTERVAL` = 1s, and `ALLOWED_HELLO_LOSS` = 2), HELLO messages are sent every 1s, and the timeout associated with link failure detection is 2s. In other words , a link failure is assumed in our testbed if a node fails to receive two consecutive HELLO messages from its neighbor. To compare the TCP behavior with AODV-LL and AODV-HELLO we thus used the ns-2 simulation tool [Ns-2]. Specifically, we assumed that the interference range (IF_Range) is equal to the carrier sensing range (CS_Range) and both are twice as large as the transmission range, and set all the other parameters as in the experimental testbed.

Figure 8 and Figure 9 show the throughput (left-side plot) and congestion window size (right-side plot) vs. time with AODV-LL and AODV-HELLO, respectively. These results are related to the 3-hop scenario with maximum *cwnd* of size 2. However, we found similar results for the other scenarios and maximum *cwnd* sizes as well.

We can observe that with AODV-HELLO, the short link-failure detection timeout (2s) causes false link-failure detections that forces AODV to trigger a new route discovery process. During route discovery process no segment is transmitted towards the final destination and the instant throughput decreases to zero, as shown in Figure 9 (left). In addition, the TCP sender experiences delayed ACKs and/or timeouts which trigger the congestion control mechanism. This is because the *cwnd* size decreases to one when the throughput is null as shown in Figure 9 (right).

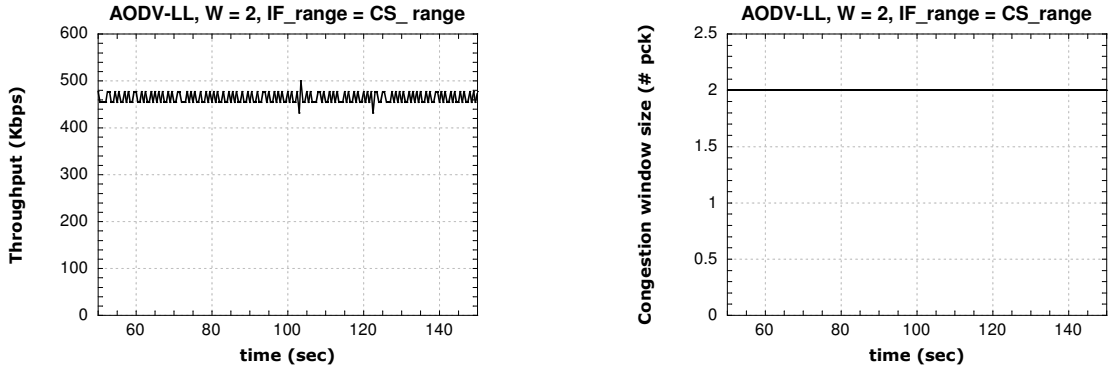


**Figure 8. Throughput (left) and congestion window size (right) vs. time when using AODV-LL in the 3-hop scenario with maximum cwnd of size 2. The interference range (IF_Range) is assumed equal to the Carrier Sensing Range (CS_Range).**

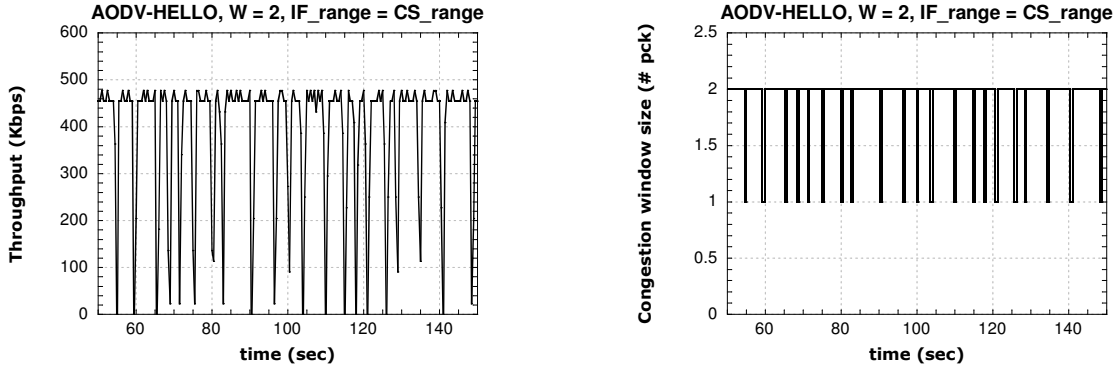


**Figure 9. Throughput (left) and congestion window size (right) vs. time when using AODV-LL in the 3-hop scenario with maximum cwnd of size 2. The interference range (IF_Range) is assumed equal to the Carrier Sensing Range (CS_Range).**

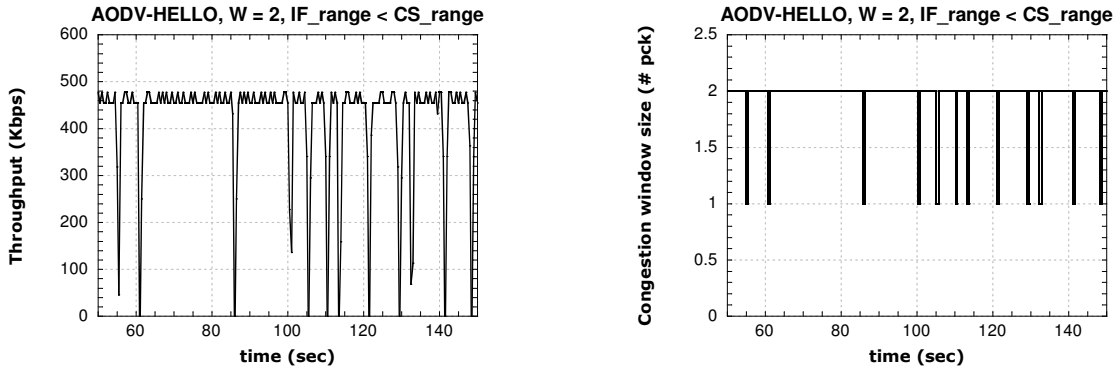


**Figure 10. Throughput (left) and congestion window size (right) vs. time when using AODV-LL in the 3-hop scenario with maximum cwnd of size 2. The interference range (IF_Range) is less than the Carrier Sensing Range (CS_Range).**

When using AODV-LL there is no link failure notification from the data link layer below and, hence, the TCP *cwnd* size and throughput remain constant, as shown in Figure 8. The difference in detections between the two methods can be explained as follows. In the AODV-HELLO case it's sufficient to loose two *broadcast* packets to detect a link failure. In the AODV-LL case a link

failure is detected when a *unicast* packet is lost. Since unicast packets are re-transmitted up to 7 times, while broadcast packets are transmitted just once, the AODV-HELLO mechanism proves to detect link failures quite more frequently than AODV-LL. In conclusion, the TCP throughput whit AODV-HELLO is significantly lower than that with AODV-LL. In addition, frequent false link-failure detections make the TCP throughput with clamped *cwnd* size not so different from the throughput with unclamped congestion window. Actually, the TCP throughput is limited by false link-failures rather than the *cwnd* size.

We also did some simulations runs by assuming IF_Range<CS_Range, which is more realistic. As expected, we observed no difference when using AODV-LL, and a reduced number of false link-failure detections when using AODV-HELLO (see Figure 10).

### 4.3. Influence of the background traffic

We also investigated the influence of interfering traffic. To this end, we considered the 3-hop scenario described above and added a CBR (Continuous Bit Rate) session to it. This CBR session has N3 as its source node and N2 as it recipient node, and uses UDP as the transport protocol. It inject in the network a periodic traffic pattern with a bit rate equal to 192 Kbps, which correspond to the bit rate of an MP3 stream**.** The results obtained in this scenario (throughout referred to as 3-hop-UDP) are shown in Figure 11. There is no qualitative difference with the results in , except that TCP throughputs are lower and the retransmission indices greater. As in the 3-hop scenario without background traffic, the optimal *cwnd* size is 3. But, as above, there are not significant differences associated with the various maximum *cwnd* sizes.

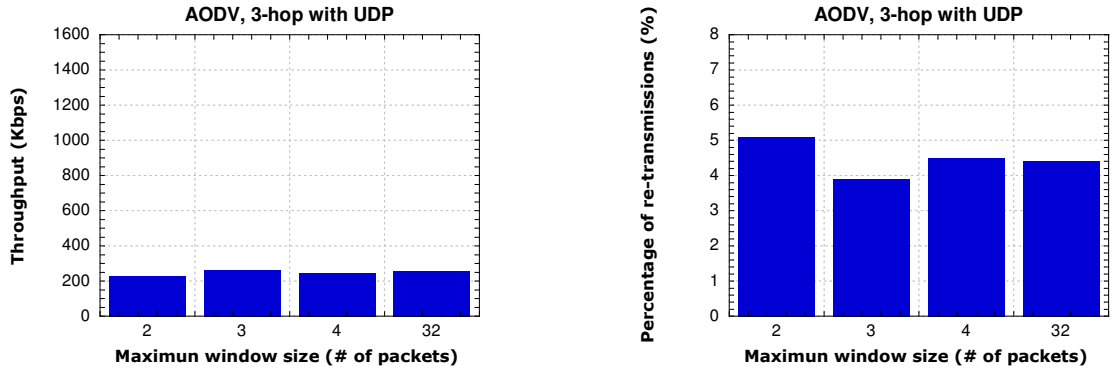


**Figure 11. Throughput (left) and percentage of retransmitted segments (right) vs. maximum *cwnd* size in the 3-hop scenario with background periodic UDP traffic . The routing protocol is AODV.**

The results discussed above are summarized in Table 1 and Table 2, where minimum and maximum values measured in the various experiments are also reported.

### 4.4. Analysis with OLSR routing protocol

To conclude our analysis we also performed some experiments with a routing protocol different from AODV. We used OLSR which is a proactive protocol (while AODV is a reactive protocol). The results obtained, in terms of throughput and percentage of re-transmissions, are summarized in Table 3 and Table 4, respectively.

Experiments with AODV and OLSR were carried out in different days. Therefore, a direct comparison of results in corresponding scenarios does not make sense because of different external conditions. However, we can observe that the results with OLSR are not very different from those with AODV under the same conditions. An important issue is that the retransmission index with OLSR is always significantly lower than that with AODV. We observed that the latter results is confirmed by simulations.

One possible reason for this behavior is the different parameter values used by AODV and OLSR to manage HELLO messages. OLSR sends HELLO messages periodically every `HELLO_INTERVAL` and considers the information provided by a HELLO message valid for `NEIGHB_HOLD_TIME` seconds. Assuming default parameter values, `HELLO_INTERVAL` is set to 2s and `NEIGHB_HOLD_TIME` to 6s. Therefore, when using OLSR a node considers a link as broken if it fails to receive *three* consecutive HELLO messages from its neighbor. Instead, as described above, with AODV a node assumes a link failure when it fails to receive *two* consecutive HELLO messages. Hence, OLSR is more robust to false link failures. In addition, since routing protocols flush out the queue of pending transmissions when they detect a link failure, if AODV detects a larger number of link failures the fraction of segments discarded is larger as well.

**Table 1. Throughput (in Kbps) vs. maximum *cwnd* size with AODV.**

| Scenario | W=2 | | W=3 | | W=4 | | W=32 | |
|---|---|---|---|---|---|---|---|---|
| | Avg | min-max | avg | min-max | avg | min-max | avg | min-max |
| 1-hop | 1387,5 | 1382-1392 | 1455,8 | 1463-1467 | 1479,4 | 1446-1507 | 1508,7 | 1500-1512 |
| 2-hop | 441,2 | 390-533 | 538,2 | 677-432 | 553,9 | 524-626 | 526 | 448-559 |
| 3-hop | 343,6 | 225-412 | 397,8 | 334,9-453 | 377,46 | 274-425 | 363,6 | 261-422 |
| 4-hop | 235,2 | 146-290 | 263,3 | 195-315,4 | 224,14 | 145-274 | 232,9 | 148-262 |
| 3-hop-UDP | 229 | 194-259 | 260,7 | 243-304 | 244 | 152-366 | 258 | 175-375 |

**Table 2. Percentage of re-transmissions vs. maximum *cwnd* size with AODV.**

| Scenario | W=2 | | W=3 | | W=4 | | W=32 | |
|---|---|---|---|---|---|---|---|---|
| | Avg | min-max | avg | min-max | avg | min-max | avg | min-max |
| 1-hop | 0 | 0-0 | 0 | 0-0 | 0 | 0-0 | 0 | 0-0 |
| 2-hop | 2,3 | 0,98-3,4 | 1,3 | 0,25-2,58 | 1,84 | 0,81-2,33 | 2,16 | 1,25-3,62 |
| 3-hop | 1,57 | 0,65-2,85 | 1,12 | 0,5-1,7 | 1,8 | 1,25-2,9 | 4,58 | 3,3-6,4 |
| 4-hop | 4 | 2,2-6,9 | 3,6 | 2,2-6,3 | 3,9 | 3,4-4,5 | 7,1 | 4,7-8,4 |
| 3-hop-UDP | 5,1 | 3,9-6,3 | 3,9 | 2,2-4,7 | 4,5 | 1,3-6,7 | 4,4 | 2,1-7 |

**Table 3. Throughput (in Kbps) vs. maximum *cwnd* size with OLSR.**

| Scenario | W=2 | | W=3 | | W=4 | | W=32 | |
|---|---|---|---|---|---|---|---|---|
| | avg | min-max | avg | min-max | avg | min-max | avg | min-max |
| 1-hop | 1369,13 | 1331-1396 | 1456 | 1428-1471 | 1473,6 | 1439-1503 | 1523,9 | 1518-1531 |
| 2-hop | 627,3 | 581-650 | 676,3 | 616-705 | 698,4 | 668-719 | 696,5 | 638-723 |
| 3-hop | 213,4 | 126-330 | 282,1 | 130-390 | 229,9 | 83-351 | 275,5 | 131-438 |
| 4-hop | 175,1 | 139-214 | 172,8 | 155-195 | 151,8 | 87-189 | 162,55 | 121-213 |
| 3-hop-UDP | 238,9 | 201-276 | 259,3 | 193-302 | 218,7 | 202-258 | 233,3 | 174-262 |

**Table 4. Percentage of re-transmissions vs. maximum *cwnd* size with OLSR.**

| Scenario | W=2 | | W=3 | | W=4 | | W=32 | |
|---|---|---|---|---|---|---|---|---|
| | avg | min-max | avg | min-max | avg | min-max | avg | min-max |
| 1-hop | 0 | 0-0 | 0 | 0-0 | 0 | 0-0 | 0 | 0-0 |
| 2-hop | 0 | 0-0 | 0,07 | 0-0,22 | 0 | 0-0 | 0 | 0-0 |
| 3-hop | 1,09 | 0,56-2,33 | 1,12 | 0,19-2,47 | 1,45 | 0,72-2,86 | 1,87 | 0,61-2,53 |
| 4-hop | 3,55 | 1,4-7,22 | 3,2 | 2,3-5 | 3,85 | 2,5-5,25 | 4,8 | 2,8-7,44 |
| 3-hop-UDP | 1,5 | 1,3-1,9 | 1,36 | 0,8-2,25 | 1,35 | 0,33-2,44 | 1,8 | 1,14-2,25 |

## 5. Conclusions

TCP performance over multi-hop ad hoc networks (MANETs) have been extensively analyzed in many previous studies. However, most of them are based on simulation results, and some of then takes simplistic assumptions, e.g., they do not consider the effect of the routing protocol. On the

other hand, several previous studies have shown the importance of an experimental analysis when dealing with MANETs. In this chapter we have used an experimental testbed based on WiFi technology, and measured the TCP performance in an indoor environment by considering two different routing protocols (i.e, AODV and OLSR).

For the sake of simplicity and, also, for better comparison of experimental and simulation results, we have limited our analysis to static networks with a chain topology and a limited number of hops. We have found some interesting results contrasting with simulations. In particular, we have found that in our testbed with a chain topology of four hops or less, the optimal performance is achieved with a maximum *cwnd* size equal to 3 (instead of 2, as suggested by simulation). In addition, the TCP performance with limited congestion window size is not so different from that achievable with an unclamped congestion window. We have shown that these discrepancies are due to the different protocols -- or different protocol implementations -- used in practice with respect to simulation tools.

In conclusion, by showing cases in which real TCP implementations and popular simulation implementations behave pretty differently, we believe that this work can further motivate to take a *experimental* approach in further investigating the TCP behavior over MANETs.

## References


[AodvUU] AODV-UU, AODV Linux Implementation, University of Uppsala. Available at: http://core.it.uu.se/AdHoc/AodvUUImpl.

[Agg00] A. Aggarwal, S. Savage, and T. Anderson, “Understanding the Performance of TCP Pacing”, Proceedings of IEEE INFOCOM 2000, Tel Aviv, Israel, March 2000

[Alt03] E. Altman and T. Jimenez, “Novel Delayed ACK Techniques for improving TCP Performance in Multihop Wireless Networks," Proceedings of the *IFIP International Conference on Personal Wireless Communications (PWC 2003)*, Venice, Italy, September 23-25, 2003,. *Lecture Notes in Computer Science*, N. 2775*, pp. 237-250.*

[Ana04] G. Anastasi, E. Borgia, M. Conti, E. Gregori, “Wi-Fi in Ad Hoc Mode: A Measurement Study, Proceedings of the *IEEE International Conference on Pervasive Computing and Communications (PerCom 2004),* Orlando (Florida), March 14-17, 2004.

[Ana05] G. Anastasi, E. Ancillotti, M. Conti, A. Passarella, “TPA: A Transport Protocol for Ad hoc Networks”, Proceedings of the *IEEE Symposium on Computers and Communications (ISCC 2005),* Cartagena (Spain), June 27-30, 2005.

[Anc06] E. Ancillotti, R. Bruno, M. Conti, E. Gregori, and A. Pinizzotto, “A Layer-2 Architecture for Interconnecting Multi-hop Hybrid Ad Hoc Networks to the Internet,” *in Proceedings of WONS 2006,* Les Menuires, France, January, 18–20 2006, pp. 87-96.

[Ahu00] A. Ahuja, S. Agarwal, J. Sing, R. Shorey, “Performance of TCP over Different Routing Protocols in Mobile Ad Hoc Networks”, *Proceedings of the IEEE Vehicular Technology Conference (VTC 2000)*, pp. 2315-2319, May 2000.

[Bop01] R. Boppana, S. Konduru, “An Adaptive Distance Vector Routing Algorithm for Mobile Ad Hoc Networks”, *Proceedings of IEEE Infocom* 2001, Vol. 3. pp. 1753-1762, April 2001.

[Cha01] K. Chandran, S. Raghunathan, S. Venkatesan, R. Prakash, “A Feedback Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks”, *IEEE Personal Communication Magazine*, Special Issue on Ad Hoc Networks, Vol. 8, N. 1, pp. 34-39, February 2001.

[Che03] K. Chen, Y. Xue, K. Nahrstedt, "On setting TCP's congestion window limit in mobile ad hoc networks", Proceedings of the *IEEE International Conference on Communications (ICC 2003)*, Vol 2, pp. 1080-1084, USA, May 11-15, 2003.

[Che04] K. Chen, Y. Xue, S. Shah, K. Nahrstedt, "Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks", *Computer Communications*, Vol. 27, pp. 923-934, 2004.

[Cla03] T. Clausen and P. Jaquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, October 2003. Available: http://www.ietf.org/rfc/rfc3626.txt.

[Dub97] R. Dube, C. Rais, K. Wang, S. Tripathi, "Signal Stability-based Adaptive (SSA) Routing for Ad Hoc Mobile Networks", *IEEE Personal Communications Magazine*, pp. 36-45.

[Dye01] T.D. Dyer, R.V. Boppana "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks", *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, October 2001.

[Fu02] Z. Fu, X. Meng, S. Lu, "How Bad TCP Can Perform in Mobile Ad Hoc Networks", Proceedings of the *IEEE Symposium on Computers and Communications (ISCC 2002)*, Taormina-Giardini Naxos (Italy), July 2002, pp. 298-303.

[Fu03] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", *Proceedings of IEEE INFOCOM 2003*, San Francisco (California), March 30.April 3, 2003.

[Gu04] A. Gupta, I. Wormsbecker, C. Williamson, "Experimental Evaluation of TCP Performance in Multi-hop Wireless Ad Hoc Networks", Proceedings of *IEEE/ACM MASCOTS*, pp. 3-11, Volendam, Netherlands, October 2004

[Hol02] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks", *Wireless Networks*, Vol.8, pp. 275-288, 2002.

[Joh04] D. B. Johnson, D. A. Maltz and Y.-C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", *Internet Draft,* July 19 2004. Available: http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt.

[Kaw05] V. Kawadia, P. Kumar, "Experimental investigation into TCP Performance over Wireless Multihop Networks", *Proc. of ACM SigCom 2005 Workshops*, Philadelphia (PA), August 22-25, 2005.

[Kim00] D. Kim, C. Toh, Y. Choi, "TCP-Bus: Improving TCP Performance in Wireless Ad-Hoc Networks", *Journal of Communications and Networks*, vol.3, no.2, pp. 1-12 June 2001.

[Kim05] D. Kim, H. Bae, J. Song, J.-C. Cano, "Analysis of the Interaction between TCP Variants and Routing Protocols in MANETs", Proceedings of the *IEEE International Conference on Parallel Processing Workshops (ICPPW'05),* pp. 380-386, June 14-17, 2005,.

[Li01] J. Li, C. Blake, D. De Couto, H. Lee, R. Morris, "Capacity of Ad Hoc Wireless Networks", Proc. ACM/IEEE International Conference in Mobile Computing and Networking (MobiCom 2001), Rome, Italy, July 2001.

[Liu01] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 19, N. 7, pp. 1300-1315, July 2001.

[Ns-2] The Network Simulator - ns-2 (version 2.28). http://www.isi.edu/nsnam/ns/index.html.

[Oli05] R. de Oliveira, T. Braun, "A Dynamic Adaptive Acknowledgment Strategy for TCP over Multihop Wireless Networks", Proceedings of *IEEE Infocom 2005*, Vol. 3, pp. 1863-1874, Miami, USA, March 13-17, 2005,.

[Osi06] E. Osipov, C. Tschudin, "Evaluating the Effect of Ad Hoc Routing on TCP Performance in IEEE 802.11 Based MANETs", Proc. *6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN)*, March 2006.

[Pap04] S. Papanastasiou, M. Ould-Khaoua, "TCP Congestion Window Evolution and Spatial Reuse in MANETs", *Journal of Wireless Communications and Mobile Computing*, Vol. 4, N. 6, pp 669-682, Sept. 2004.

[Pap05] S. Papanastasiou, M. Ould-Khaoua, L. MacKenzie, "TCP Developments in Mobile Ad Hoc Networks", Chapter 30 in Handbook of Algorithms and Wireless Networking and Mobile Computing (A. Bouchercke editor).

[Par97] V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", *Proceedings of IEEE INFOCOM '97*, Kobe (Japan), 1997, Vol. 3, pp. 1405-1413.

[Per94] C. Perkins, P. Bhagwat, "Higly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers", Proc. of the *Conference on Communications Architectures, Protocols and Applications*, pp. 234-244, New York, 1994.

[Per03] C. Perkins, E. Belding-Royer, S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003. Available at: http://www.ietf.org/rfc/rfc3561.txt

[Rak05] S. El Rakabawy, C. Lindemann, M. Vernon, "Improving TCP Performance for Multihop Wireless Networks", *Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN 2005)*, 2005.

[Sun03] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad hoc Networks," *Proc. ACM Symposium on Mobile Ad Hoc Network and Computing (MobiHoc 2003)*, Annapolis (Maryland), June 2003.

[Sun01] D. Sun and H. Man, "ENIC - An Improved Reliable Transport Scheme for Mobile Ad Hoc Networks", *IEEE Globecom Conference*, 2001 (San Antonio, TX), Nov. 2001, Vol. 5, pp. 2852-2856.

[Tan99] K. Tang, M. Gerla, "Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks", Proceedings of IEEE MMT'99, Venice (I), October 1999.

[Ton04] A. Tønnesen, "Implementation of the OLSR specification (OLSR UniK)", Version 0.4.10, University of Oslo. Available at: http://www.olsr.org/.

[Wan02] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response", *Proceedings of the third ACM International Symposium on Mobile Ad Hoc Networking &Computing* (MobiHoc 2002), pp. 217-225, Lausanne, Switzerland, 2002.

[Xu01] S. Xu, T. Saadawi, "Performance Evaluation of TCP Algorithms in Multi-hop Wireless Packet Networks", *Wireless Communications and Mobile Computing*, Vol. 2 (2001), N. 1, pp.85-100.

[XuS02] S. Xu, T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks", *Computer Networks*, Vol. 38 (2002), pp. 531.548.

[XuB02] K. Xu, S. Bae, S. Lee, M. Gerla, "TCP Behavior across Multihop Wireless Networks and the Wired Networks", Proceedings of the *ACM Workshop on Mobile Multimedia (WoWMoM 2002),* Atlanta (GA), September 28, 2002, pp. 41-48.

[Xu05] K. Xu, M. Gerla, L. Qi, Y. Shu, “TCP Unfairness in Ad Hoc Wireless Networks and a Neighborhood RED Solution”, *Wireless Networks, Vol. 11, pp. 383-399.*