TCP Performance in Wireless Mobile Multi-hop Ad Hoc Networks

OLA WESTIN

KTH Microelectronics
and Information Technology

Master of Science Thesis
Stockholm, Sweden 2003

IMIT/LCN 2003-11

TCP Performance in Wireless Mobile Multi-hop Ad Hoc Networks

Ola Westin
Swedish Institute of Computer Science
owe@sics.se

4th December 2003

Abstract

There are many issues that limit the performance of wireless mobile multi-hop ad hoc networks (MANETs). One of them is that TCP is not well adapted to networks where routes can change or disappear often. In this paper the behaviour of a standard TCP implementation is studied in situations typical for MANETs and compared to the behaviour of a partial implementation of a ATCP, a TCP modification that is intended to increase performance in MANETs.

Simulations with simple scenarios show that TCP easily creates a full network load which causes send failures and decreased throughput performance. In some cases the partial ATCP implementation increases throughput but more often it causes an increased amount of duplicate retransmissions. In these scenarios it is unlikely that even a complete ATCP implementation would increase throughput performance.

A few modifications to ATCP and TCP are analysed. Especially a limit of the congestion window size shows a large throughput increase.

The results are inconclusive, the simulations are too simple to show if the results are applicable in more complex scenarios. It is not clear if ATCP actually is useful in a MANET.

Sammanfattning

Många faktorer begränsar prestandan i trådlösa mobila multi-hopp ad hoc-nätverk (MANET:er). En av dem är att TCP inte är anpassat till nätverk där rutter ofta kan förändras eller försvinna. I den här rapporten studeras hur en vanlig TCP-implementation uppför sig i typiska MANET-situationer. Detta beteende jämförs mot en partiell implementation av ATCP, en TCP-modifiering som är tänkt att öka prestanda i MANET:er.

Simuleringar med enkla scenarier visar att TCP lätt genererar en full nätverkslast vilket orsakar misslyckade sändningar och en minskad genomströmningsprestanda. I vissa fall ökar den partiella ATCP-implementationen genomströmningen, men oftare ger den en ökad mängd onödiga omsändningar. I dessa scenarier är det inte troligt att ens en komplett ATCP-implementation skulle öka genomströmningsprestanda.

Några mindre förändringar av ATCP och TCP analyseras. Särskilt ger en begränsning av stockningsfönstret en stor ökning av genomströmningen.

Resultaten är ofullständiga. Simuleringarna är för enkla för att kunna visa om om resultaten är tillämpliga i mer komplexa scenarier. Det är inte klarlagt ifall ATCP verkligen är användbart i ett MANET.

Contents

1 Introduction

Wireless mobile multi-hop ad hoc networks, or *MANETs*, offers network connectivity without using any fixed infrastructure. This can be useful in a wide variety of situations, for example in disaster areas or sensor networks. Since MANETs have a slightly different behaviour than traditional wired networks, using the standard TCP congestion control can lead to suboptimal performance.

To avoid the greatest sources of decreased throughput performance a TCP modification called ATCP [25] has been proposed. It uses a few relatively simple mechanisms to detect different causes of packet loss and tries to deal with them in appropriate ways.

In this paper a partial implementation of ATCP for the network simulator *ns-2* is studied to see how it behaves in a MANET environment compared to the standard TCP implementations. The results show that the partial implementation causes a performance decrease due to it misinterpreting the network state and therefore deals with it in the wrong way.

The background section contains an introduction to MANETs and the factors that affect their performance. In the method section the modifications to the TCP stack are presented and the results of the simulations are shown in the analysis section. Finally, in the conclusions section, the results are summarised and a few suggestions for future work are given.

2 Background

2.1 Introduction to ad hoc networks

An ad hoc network is a network where the topology is unplanned and possibly not even fixed. This is a quite loose definition that can be more tightly specified by adding the words *wireless*, meaning that the nodes use wireless interfaces to communicate, *mobile*, meaning that the nodes can move around, and *multi-hop*, meaning that the nodes can communicate with other nodes indirectly by using closer nodes as relays.

Often the shorter term *mobile ad hoc network*, or *MANET*, is used to specify a wireless mobile multi-hop ad hoc network. The term MANET will be used throughout this report with this meaning.

2.2 Overview

Several different parts, or layers, are needed for an ad hoc network to work. There must be a way to physically transport data between nodes and there must be one or more protocols to ensure that the data arrives at the correct node and the correct application.

These layers interact with each other in a well-defined manner. A layer can pass data to another layer, which in turn can add or remove its own layer specific information and pass the result on to the next layer.

The details of how each layer works are usually ignored by the other layers. However, characteristics like delays and added packet overhead can affect the performance of other layers in more or less subtle ways.

- Link layer

  To build a network there must be some way to transmit data between the nodes. For MANETs this is usually done with wireless interfaces. These interfaces can logically be divided into two parts: the physical layer, which converts data into physical signals (such as radio waves) and back again, and the link protocol layer, which handles details like who the source and destination are and when each of the nodes are allowed to send. It is also possible for the link layer to detect transmission errors and do retransmissions of link frames.

- Ad hoc routing

  In a multi-hop environment there must be some way of finding a route between two nodes. This is done with an ad hoc routing protocol. Often the routing protocol operates below the network layer, but still has knowledge about it.

- Network layer

  The network layer is responsible for delivering packets based on the nodes' network addresses and for knowing which nodes are on the same subnet. Since compatibility with Internet applications is often desired, most MANETs are designed with the TCP/IP stack [40] in mind.

  A MANET is usually treated as a single IP subnet. This means that on the IP level all nodes are considered to be directly connected to each other. The ad hoc routing is hidden by the ad hoc routing protocol and normally normally can not be seen by higher layer protocols.

- Transport layer

  While the network layer is mainly concerned with getting data from one node to another, the responsibility for sorting incoming data and passing it on to the correct application falls on the transport layer. The transport layer can also add features like detection of transmission errors, requesting retransmission of missing or bad packets, and sorting incoming data so that it is passed on to the application in the same order as it was sent. Note that the transport layer error detection and retransmission is usually unaware of link layer error detection and retransmission mechanisms.

- Application

  Applications that use the network usually have no idea about what the network's topology is. They are usually aware of who they are communicating with, but the details of how the data is transmitted are generally invisible to them. This is good, since it allows programmers to write applications without worrying about all the different lower layers that could or do exist. On the other hand, it means that applications can not easily adapt their behaviour based on lower layer conditions.

Also, there are some other factors that can affect the application layer user's perceived performance in different ways. Even though they are not directly involved in the data transmission they must be considered when studying MANETs, because of their effects on the data transmission.

- Mobility

  One of the most prominent useful features of MANETs is that the nodes can move around freely. On the other hand, this also generates a lot of problems. The nodes can move in and out of range of each other and in order to maintain connectivity to specific nodes the ad hoc routing protocol must find new routes through the network. Even if the nodes that are communicating with each other are stationary they can still be affected by the movement of *other* nodes.

  In some cases a node or a group of nodes can move such that there is no possible route between certain other nodes in the network. In this case the network is said to be partitioned and no communication is possible between nodes in the different partitions.

- Topology and environment

  The physical location of the nodes can also affect their performance. If a lot of nodes are located in a small area, there will be a greater amount of contention for the available transmission capacity in this region, but there is a decrease in the probability of the network partitioning.

  The surrounding environment, for example metal walls or strong electromagnetic fields, can affect the link layer performance by hindering radio waves or by otherwise interfering with the transmissions.

- Network traffic

  A network with a high traffic load will naturally have have a higher probability of longer delays before a node can transmit and, with link layer protocols without collision avoidance, a higher probability of collisions. The characteristics of the traffic, such as how well it is distributed over the network, or if it has a constant data rate or comes in shorter bursts, can also affect the throughput rate. Collision probability is dependent on MAC protocol details; there are some protocols which are collision free.

2.3 Wireless networks

The difference between wired networks and a wireless network is, as the later name implies, that the nodes communicate without a wired infrastructure. There are several different technologies that are widely used for wireless communication, e.g. radio, infrared, and laser links. In MANETs radio interfaces are usually used.

What can make wireless networks more difficult to use is that they both can cause more interference and are more vulnerable to interference than traditional wired networks. In a wired network the transmitted data is confined to cables or fibres, often in point-to-point connections. Even if there can be problems with crosstalk between wires, the problems are not as prominent as in wireless networks. In the case of radio communication by mobile nodes the data is often transmitted with non-directional antennas and can easily be overheard by others within range. Even nodes outside of the effective communication range can be affected, since radio transmissions can interfere farther away than they can be correctly received. Furthermore, for the receiver the effects of various sources of background noise must be added to the problem of interference by other nodes.

Because of this, wireless frame transmissions can not be considered to be very reliable. In wired networks the major cause of missing packets is that routers are congested. In the wireless case the reason can more often be link layer transmission errors.

There are several different radio technologies that can be used for data communication. The most common today is IEEE 802.11 wireless LAN, but there are also others such as Bluetooth, GSM (for mobile phones), and a multitude of experimental technologies designed for MANETs.

2.3.1 IEEE 802.11

IEEE 802.11 wireless LAN networks are by far the most common radios used in MANET studies. This is because the hardware is widely available and inexpensive. Both the original IEEE 802.11 standard [17] and the supplement for the higher bit rate IEEE 802.11b standard [18] are available for free from IEEE [19].

In addition to the higher probability for low link quality that is common to all wireless communication, there are some characteristics of IEEE 802.11 networks that can cause problems in MANETs. Xu and Saadawi discuss the following in [44]:

- No multi-hop awareness

  IEEE 802.11 is designed for networks where traffic is routed through access points or single-hop ad hoc networks. These single-hop networks can be combined into a multi-hop network by using an ad hoc routing protocol, but since the link layer is not aware of the traffic outside its own single-hop network, it can not take it into consideration when deciding when to send data. This, in combination with the fact that transmissions cause interference farther away than they can be successfully received, can cause an increased probability of incorrect reception.

- Hidden station

  There is a collision risk if two nodes are far enough apart that they can't hear each others transmissions and both want to sent data to a receiver that can hear both of them, since the senders don't know when the other one is transmitting. This is known as the hidden station problem. In IEEE 802.11 networks this problem is avoided by using request to send (RTS) and clear to send (CTS) packets. A node that wants to send a packet starts by broadcasting an RTS packet. The receiver responds by sending back a CTS packet. In the RTS and CTS packets there is information about how long the sender needs the medium. All nodes that can hear either of the packets will avoid transmitting any data for that time. Figure 1, 2, 3, and 4 illustrate this process. There is, of course, a probability that two nodes send RTS packets at exactly the same time but, since these packets are shorter, the probability of collision is much smaller than for data packets. The performance impact of such a collision is also much lower than a collision with data packets.

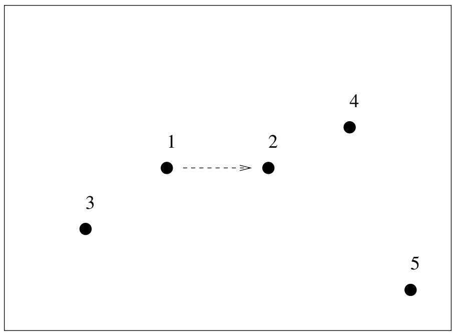

Figure 1: Node 1 intends to send a data packet to node 2.

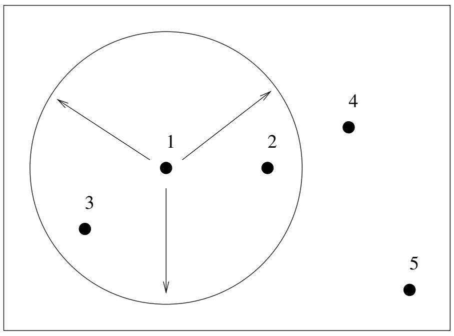

Figure 2: Node 1 sends an RTS packet that is seen by nodes 2 and 3.

- Exposed station and instability

  While the CTS/RTS mechanism decreases the problem of hidden stations, it can not do anything about the related problem known as “exposed station”. In some cases a node may be blocked from sending even though it can't hear any other traffic. If the intended receiver is within interference range of other transmissions, it may not be able to receive the packets correctly or even send CTS packets. If this continues for long enough, the sender will give up and assume that a route failure has occurred, which will result in delays while new routes are unnecessarily discovered.

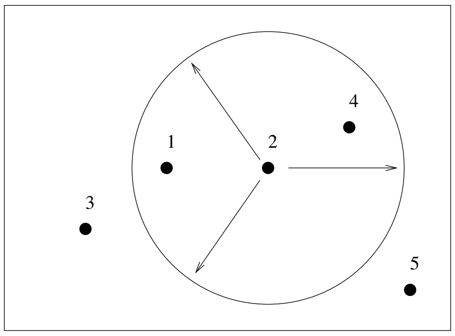

Figure 3: Node 2 responds with a CTS packet that is seen by nodes 1 and 4.

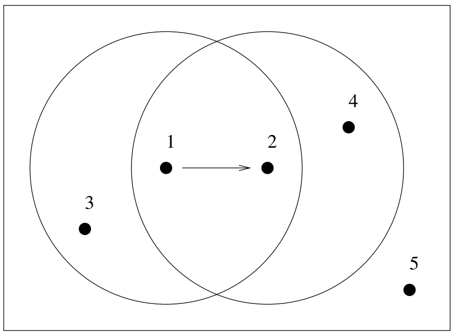

Figure 4: Node 1 can start sending data. All nodes within communication range of either node 1 or 2 are silent.

- Unfairness

  Another related problem is that of unfairness. If a node fails to send a packet because the channel is busy, it will wait for increasingly long periods of time before trying again. This means that in a network with many competing nodes the node with the last successful transmission will have an advantage over the others if it tries to send again. This unfairness can also be observed in wired networks such as Ethernet, which use the CSMA/CD protocol for channel allocation, even though it does not pose as large a problem in modern switched networks where most of the traffic is passed through dedicated point-to-point connections.

2.3.2 Other network technologies

In addition to IEEE 802.11, there are several other network technologies that can be used for MANETs.

Bluetooth is a communication technology designed for small mobile devices. The Bluetooth standard covers both the hardware and a communication protocol stack. Groten and Schmidt [13] claim that Bluetooth shows promise, but still has some issues that limit its usefulness in MANETs. Primarily this is because multi-hop support in the standard is incomplete. The Bluetooth standards are available at the Bluetooth Special Interest Group web site [3].

TDMA is a scheduling algorithm where each node is assigned a transmission schedule indicating in which of a series of time slots it is allowed to transmit. In the related protocol Spatial TDMA (STDMA) [28] geographic information is added so that nodes that are located far enough apart from each other may be assigned the same time slots.

There are also many experimental protocols designed with MANETs in mind. Examples are ADAPT [7] and CATS [43] that use different methods for allocating transmission slots. Other protocols like PAMAS [38] are designed to use less energy, which is important for battery powered nodes.

2.4 Ad hoc routing

Several routing protocols for MANETs have been designed, each with its own set of strengths and weaknesses. Feeney [8] has written an overview of a number of these and classified them into groups.

2.4.1 DSR and AODV

Two of the more popular routing protocols are DSR [21] and AODV [33].

Both of these are reactive protocols, which means that they wait until the node wants to send data to another node before trying to find a route to the destination.

The largest difference between them is that DSR finds the complete path to the destination and includes it in each data packet while AODV nodes only know which neighbour to forward packets to for each destination. This has the effect that DSR requires more overhead per packet when sending data and AODV requires more overhead for discovering routes.

Despite the differences several studies show that DSR and AODV perform similarly. Broch et al. [4] found that both DSR and AODV worked well, but that AODV was slightly more expensive in situations with higher mobility. Johansson et al. [20] found that DSR usually performs better than AODV in situations with low traffic loads and AODV performs better under high traffic load. Östergren [31] found that even though AODV causes a higher packet loss following a topology change there is not a significant difference in throughput for TCP.

2.4.2 Other routing protocols

Proactive protocols like DSDV [34] and GSR [6] are similar to AODV and DSR, but try to keep correct routing information to other nodes at all times to eliminate the delay that is otherwise needed before a connection can be established. Non-uniform protocols like ZRP [16] and CEDAR [39] differ in that all nodes are not equal with regard to routing. Either the nodes use only some of their neighbours for routing and forwarding or they can negotiate a hierarchical structure for the routing. There are also some protocols like GRID [24] that use geographic information for routing.

Many of the problems in MANETs are caused by the desire to support large networks. LUNAR [26] is a protocol that has been designed for simple operation of networks that are small enough (for LUNAR the network size is limited to 3 hops) to avoid these problems.

2.5 Transport protocols

While the network protocol takes care of getting data packets to the correct nodes, the transport protocol's first responsibility lies in extracting the data from these packets and passing it on to the correct application, i.e. demultiplexing the incoming data.

The most common transport protocols today are UDP and TCP, but other protocols are also used. UDP is simple enough that there is only one version in use but TCP has continuously been modified and there exist many different versions of it.

2.5.1 UDP

The User Datagram Protocol [35] is a very simple transport protocol. All that it provides is multiplexing of application data streams, i.e. the ability to send data to specific applications on the destination node, and an optional weak checksum of the transported data. There is no guarantee that the data is ever delivered to the destination and the data packets can arrive in any order. This may sound bad, but if there is no need for reliable, in-order delivery, then there is no need to waste resources on it. The simplicity makes it faster than more complicated protocols like TCP, since there is very little overhead (only 8 bytes) for each packet.

UDP traffic has been used in many MANET studies, often for sending one or more data streams with a constant bit rate through a network while other parameters are varied.

2.5.2 TCP

The Transmission Control Protocol [36] is used when all data *must* arrive at the destination application and do so in the same order as it was sent.

TCP is a connection oriented protocol, which means that before any data can be sent the two endpoints must establish a connection by sending control packets back and forth. When the connection is not needed any more it may be closed by the applications. TCP also supports several congestion control mechanisms that help the sender to transmit data rapidly while avoiding overloading the network.

As the resulting TCP implementations have shown quite complex behaviour, a lot of effort has been made to understand TCP's behaviour. Padhye et al. [32] have developed a model for estimating the throughput of a TCP flow. Allman and Falk [1] have made some recommendations about what to consider when evaluating TCP modifications.

2.5.3 TCP congestion control

There are a few standard implementations of TCP that are often used in experiments: TCP Tahoe [2], Reno [2], and NewReno [9]. These code names are taken from the UNIX distributions that they first appeared in. The difference between these implementations is which flow control mechanisms have been included in them.

TCP Tahoe works according to RFC793 [36] with the addition of slow start, congestion avoidance, and fast retransmit from RFC2581 [2]. TCP Reno is the same as TCP Tahoe with the addition of fast recovery, also from RFC2581. In TCP NewReno the fast recovery mechanism is modified according to RFC2582 [9].

The slow start mechanism exponentially increases the number of packets simultaneously in transit until a variable limit is reached. After slow start, TCP goes into congestion avoidance mode where the number of packets are increased linearly until a packet is lost. Fast retransmit means that if the sender receives three duplicate acknowledgements for a packet, then that packet is assumed to be lost and that there was no congestion, therefore the sender immediately retransmits it. Fast recovery means that after a fast retransmit has occurred, the sender goes into congestion avoidance instead of starting over with slow start.

There is also an extension to TCP called selective acknowledgement (SACK) [27] that enables the receiver to specify that a certain packet is missing and which packets after this have arrived successfully.

2.5.4 Enhancements of TCP

The standard TCP implementations have over the years been tuned to the characteristics observed in traditional wired networks. Unfortunately, wireless networks and, in particular, MANETs behave differently enough to make it difficult for these implementations to make correct assumptions about the network state.

The main problem is that if a packet is lost in a wired network the cause is usually congestion somewhere in the network. In wireless networks packet loss can also be due to transmission errors or broken routes, in which case the normal reaction of retransmissions and increasing delays may be devastating for performance.

Because of this, there have been several attempts at creating enhanced TCP implementations that work better in wireless environments.

Chandran et al. [5] have proposed a feedback-based variant called TCP-F. It enables the sender to detect route failures by using explicit route failure and re-establishment notification messages.

TCP-BuS by Dongkyun et al. [22] is similar to TCP-F. The main difference is that intermediate nodes can buffer packets when a route failure occurs, so that the packets can be retransmitted faster when a new route is found.

Liu and Singh [25] have proposed an ad hoc variant of TCP called ATCP. It tries to determine if packets are lost due to bad links, congestion, or route errors and then deals with these situations appropriately. Congestion is detected by using explicit congestion notification (ECN) messages and route failures by using normal ICMP Destination Unreachable messages. If the sender times out while waiting for an acknowledgement, and no other reason has been determined, it assumes that the packet was lost due to a bad link.

2.5.5 Other end-to-end transport protocols

Even though TCP and UDP are the protocols that are usually used when writing Internet programs today it can in some cases be better to use other transport protocols.

SCTP [42, 41] is a new protocol that combines the features of UDP and TCP and adds some new functionality of its own. Multihoming makes it possible for a connection to stay alive even if one of the endpoints' interfaces becomes unreachable. If the host is reachable on another IP address, the connection can reroute itself there. SCTP can also divide the data stream into message chunks that can be delivered to the receiver in the order that they arrive.

2.6 Topology

Where the nodes are located and what their local environment is like determines which nodes can contact each other and the amount of interference from other nodes.

If the nodes are located close to each other, there will be a greater chance that the data will not have to make as many hops as in a network where the nodes a farther apart. On the other hand, in a network with a dense concentration of nodes there will be more contention for the available capacity and also more interference. Gupta and Kumar [15] have calculated the maximum capacity of a network with randomly distributed nodes. If traffic is generated between randomly chosen nodes, then the capacity per node approaches zero as the number of nodes is increased. Even if the scenario in their calculations is a bit unrealistic, it still shows that the risk of decreased performance increases with the network size.

The environment affects the performance in a similar way. Walls and other objects can dampen, reflect, or otherwise interfere with radio transmissions, Since these effects usually are unevenly distributed, it is difficult to simulate this. Rappaport has written an introduction to indoor radio propagation [37] where the most common issues are described.

2.7 Mobility

When nodes move around there is a possibility that they will move out of range of old neighbours or into range of new ones. When that happens the ad hoc routing protocol may have to find new routes to be able to maintain communication which involved these nodes.

When measuring mobility it is not the absolute motion of the nodes that is interesting, but rather how the nodes move in relation to each other. Johansson et al. [20] have defined a mobility metric that quantifies this motion.

Usually a broken route results in a throughput penalty, since no data can transported. To minimise this penalty the routing layer should find a new route as quickly as possible. Goff et al. [12] have studied how throughput can be increased by initiating route discovery before a link actually fails.

High mobility is not always a bad thing for MANETs. Both Gupta and Das [14] and Johansson et al. [20] have observed that mobility can increase throughput by distributing traffic more evenly over the network.

2.8 Traffic

The traffic patterns consist of information about which nodes are communicating with each other and how their communication is distributed over time.

Traffic between nodes that are many hops apart will generate more load than those with fewer hops. Li et al. [23] have seen that large MANETs work best if most of the traffic is between nodes that are close to each other. As the amount of non-local traffic grows, the per node capacity decreases.

If the traffic is unevenly distributed over the network, e.g. a single node is the sender or receiver of the traffic, then that part of the network will have a higher load than other parts.

It is difficult to do realistic simulations because of the very different possible situations where MANETs can be used. Johansson et al. [20] simulated some scenarios with different topology, mobility, and traffic patterns. They found that the ad hoc routing protocol had a large impact on the performance.

2.9 Simulations

There are several ways to measure various aspects of network performance. Simulations, lab measurements, and live measurements each have their own advantages and disadvantages [10]. Some advantages of simulations are that they are cheap, easy to control in detail, and provide the ability to experiment with hardware that is not available. Disadvantages are that it can be very difficult to create good simulations and that errors in the simulator can make the results completely useless.

Major network simulation softwares includes the free Network Simulator - ns-2 [29], the partially free GloMoSim [11], and OPNET [30], which is a commercial product. All are fairly complex, which one to use depends on if any of them are already available, their compatibility with other software (e.g. for scenario generation), which ones colleagues have experience with, and the available budget.

3 Method

The goal for this project was to show how a TCP implementation that is enhanced for use in a MANET behaves in comparison to a standard TCP implementation. To do that these steps were needed:

- Select TCP implementations to compare
- Implement enhanced TCP in ns-2
- Run various simulations and compare the results

3.1 Implementation of enhanced TCP

Ns-2 already has most standard TCP types implemented. To be able to compare one of them with an enhanced TCP, this enhanced TCP must be implemented. ATCP was chosen as the enhanced TCP since it has some promising solutions to the issues in MANETs.

ATCP uses four different states for different situations that can occur.

- Normal

  This is the state that ATCP is in when no special situation has been detected. ATCP behaves like the normal TCP implementation. In this state ATCP must be able to detect and count duplicate acknowledgements and retransmission timeouts. All of this is already present in the normal TCP implementations and all that is needed is to add transitions to the other states.

- Loss

  If a segment remains unacknowledged and no other reason can be detected, then the sender assumes that the packet has been lost during transmission. ATCP freezes the TCP state and retransmits the unacknowledged segment until it is acknowledged.

  ATCP must be able to change the TCP retransmission timer behaviour and send segments from TCP's send buffer without affecting TCP's state.

- Congested

  This state also uses normal TCP behaviour. The only reason for its existence is to prohibit state changes to the loss state when congestion is detected.

  If the source node receives a TCP segment with a notification about congestion, it stops trying to detect packet loss. Since TCP already treats congestion well, ATCP will not interfere with its normal operation. ATCP stays in the congested state until a new segment is successfully transmitted.

  To implement this the intermediate nodes must generate Explicit Congestion Notification (ECN) messages and the destination nodes must echo the ECN information in acknowledgements which the source nodes must detect and act upon.

  – Intermediate nodes
    The nodes have to use a queue type which can set the ECN bits in forwarded IP packets. In ns-2 the interface queues are defined in the ad hoc routing protocols. Unfortunately, both AODV and DSR are hard coded to use simple priority queues.
    Ns-2 has a RED queue implementation that might be useful if the ad hoc routing protocol could be persuaded to use it. As an alternative the hard coded priority queues could be modified to set ECN bits when the queues are full.

  - Destination node

    It is simple to enable ECN at the destination node during simulations. No modification of ns-2 is necessary.

  - Source node

    Source node congestion handling is already present in ns-2. ATCP simply needs to intercept the ECN message and change state accordingly. It must also be able to detect when a new segment is transmitted.

- Disconnected

  When the sender has received an *ICMP destination unreachable* message from an intermediate node it responds by freezing the TCP state until a new route to the destination can be found.

  The routing protocol must be able to inform the source node's ATCP stack about broken routes. To follow the ATCP specification would mean implementing ICMP which is currently *not* present in ns-2. Besides this, the ATCP code needs a method that changes the ATCP state when an ICMP message is received.

  - Intermediate nodes

    When the ad hoc routing protocol detects a broken route it must send a message back to the source node. Since ICMP support is practically non-existent in ns-2, either it must be implemented from scratch or some other means of informing the source node must be used. The routing protocol could be modified to send a message back to the source. Alternatively a cheat solution could "magically" inform the source node about the broken route, i.e. by directly setting the state in the source node.

  - Source nodes

    As in the *loss* state, ATCP must be able to stop TCP's timers. Periodically it has to probe for for new routes by retransmitting packets from TCP's send buffer.

3.2 Changes in ns

The TCP implementation of ns is half-duplex. Each connection has two end-points: one is a source node which sends data packets and receives acknowledgements and the other is a destination node which receives data packets and sends acknowledgements. In this case only the source node needs to be modified since the destination already has all the functionality that is needed.

The TCP implementation that was initially selected to be modified and compared to was TCP Tahoe. It is the simplest of the available implementations and should serve well as an initial test. Later an implementation using TCP NewReno was added.

3.2.1 Tahoe implementation

The existing TCP Tahoe class was extended to add ATCP functionality. Most of the functionality was added by piggy-backing on existing methods.

- Variables

  One variable was added to hold the ATCP state and one to hold the TCP persist state. A few other variables were added to enable control over specific parts of the TCP behaviour from the simulation scripts.

- TCP persist mode

  TCP persist mode was piggy-backed on the normal retransmit timer. When the TCP persist flag is set, the timer changes its behaviour to a persist timer. The differences between the retransmit and persist timers are that persist mode sets the next segment to be sent to the first currently unacknowledged segment, and that the receiver's advertised window is assumed to be zero. On each timeout one persist probe is sent to see if the receiver accepts more segments.

- Broken routes

  ICMP was intended to be used for indicating broken routes. The routing protocol was to be changed to send ICMP packets to the source node when it detects a broken route. When the source node receives this packet it should call the method atcp_icmp_received() which puts ATCP in the state Disconnected.

  This part was not completely implemented. No ICMP code was added to ns-2. The only way to enter Disconnected state is to fake ICMP messages with calls to a TCL command at explicit times in the simulation scripts. This requires a time-consuming manual inspection of trace files to find the times at which route breaks occur, and is therefore not used during the majority of the simulations.

- Congestion

  The send queue used by the routing protocol was intended to be changed to set ECN flags on packets when a packet is dropped due to full queue. These flags are echoed back from the receiver to the source which in turn takes appropriate measures in the method ecn(). In the ATCP paper [25] this means entering the state Congested. In this state the normal TCP congestion behaviour is used. Since the reference code does not do this and instead uses the state Normal, the same thing was done in ns-2. All that was needed was to prevent ATCP from entering the state Loss.

  This part was not completely implemented. The changes to TCP were finished, but the routing protocols were *not* modified to generate ECN messages.

- Loss

  If a retransmit timeout occurs, and the persist flag is not set, ATCP changes to the state Loss. The same thing happens when too many duplicate acknowledgements have been received, which causes the method dupack_action() to be called. When loss is detected the congestion window is saved and TCP persist mode is enabled.

  When a new acknowledgement arrives the loss condition is considered to be over and ATCP changes to the Normal state. This causes the congestion window to be restored and transmission continues at the rate that was used before the loss condition.

3.2.2 NewReno implementation

In ns-2, TCP NewReno [9] is implemented as a subclass to the Tahoe implementation. To add ATCP functionality to it only those methods which overrided the methods in the Tahoe class needed to be changed. The implementation layout was the same as in the Tahoe implementation.

3.3 Simulation scenarios

3.3.1 Stationary one hop scenario

The stationary one hop scenario uses the simplest possible layout for a MANET. Two stationary nodes communicate with each other without any outside interference, see Figure 5 for an illustration of the layout. With this scenario a maximum bound for the performance of a given protocol implementation can be determined for comparison to later simulation results.

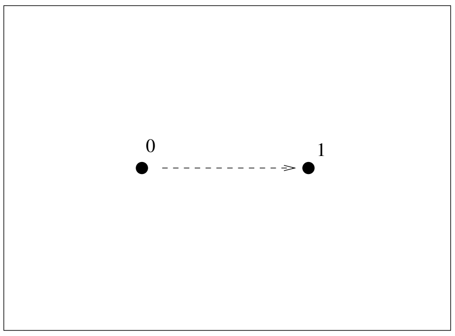

Figure 5: Stationary one hop scenario layout. Both nodes are stationary.

3.3.2 Varying network load

To see how much the network load affects the performance some simulations were run with a time delay between the generation of each data packet from the constant bit rate (CBR) agent. The offered network load can be defined as $1/delay$ packets per second where *delay* is the time between packets. As long as the network can transfer data at this rate this will result in a constant load, but if the network gets fully loaded, then packets will become queued in the interface queues.

This method of limiting the network load depends on the application knowing how high throughput the network can handle, which is not realistic to do for real applications. It is better to let the transport layer handle the capacity estimation, for example as TCP does with its congestion window. Shortening the network interface queues could also let TCP react faster to lost packets since the delays before actual transmission would be shorter. This, however, has not been tested for these simulations.

3.3.3 Varying packet loss

In some simulations the amount of packet loss is varied to see how the different TCP implementations and ad hoc routing protocols cope with it. The node layout is the same as in the stationary one hop test in order to exclude any other factors.

Packet loss is generated by adding a filter to the reception code at the physical layer. The filter is set to drop received packets at a given packet error rate (PER).

3.3.4 Increasing distance scenario

To determine the range of the simulated radio transmitter two nodes were simulated. Figure 6 shows the layout of the nodes. Node 0 is stationary while node 1 moves away at a constant speed of 1 m/s. In this scenario one node sends data to the other in a constant packet rate TCP stream. The coordinates of the moving node are examined at the time of the last successful packet reception. This range could then be used for setting the distance between nodes in the subsequent increasing hop count scenario.

This scenario, as well as the stationary one hop scenario, also works as a test of the raw performance of the combinations of the ad hoc routing protocols DSR and AODV, and normal TCP and ATCP, when no other factors are involved.

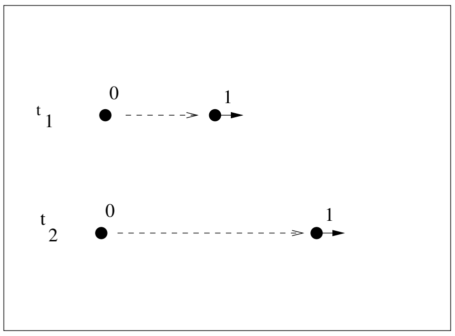

Figure 6: Increasing distance scenario layout.

3.3.5 Increasing hop count scenario

To determine the behaviour of the TCP implementations when hop count is increased another simple scenario was simulated. The simulation is similar to the increasing distance scenario, but the velocity of the moving node has been increased to 10 m/s and extra nodes have been placed along its path at 300 m intervals. The extra nodes form a linear chain where each node can communicate with its closest neighbours on each side, but not with any other nodes farther away. As the moving node moves along this chain more and more nodes are needed to pass packets to it.

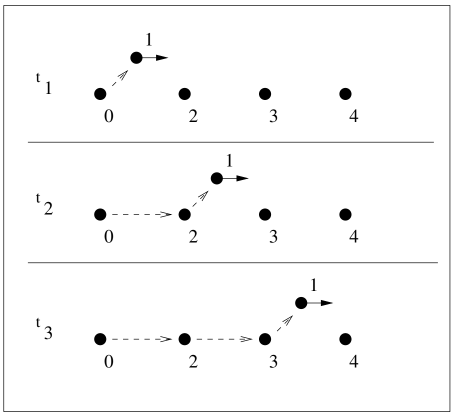

Figure 7: Increasing hop count scenario layout. As time progresses more and more intermediate nodes are needed to maintain connectivity between nodes 0 and 1.

4 Analysis

Most of the simulations were run using the stationary one hop scenario. The reason for choosing this scenario was to study the behaviour of the protocols with a minimum of interference involved. Since these results showed a more complicated behaviour than predicted, the amount of simulations with more complicated scenarios became less than intended.

4.1 Summary of simulation results

4.1.1 General performance

Figure 8 shows the transfer progress with the Tahoe implementation. DSR *without* ATCP gets the best results, while enabling ATCP decreases throughput by 7%. AODV, both with and without ATCP, experiences a throughput decrease of 28% compared to DSR without ATCP.

Figure 9 shows the transfer progress with the NewReno implementation. DSR without ATCP still gives the best results. Enabling ATCP decreases throughput by only 1%. AODV has the same throughput with and without ATCP, but has a throughput decrease of 30% compared to DSR without ATCP.

Note that these figures are only taken from one set of simulations. In different simulations the figures vary, but the general trend is the same for all simulations that used the one hop scenario.

4.1.2 Congestion window behaviour

The difference in congestion window behaviour in the different TCP implementations with and without ATCP can be illustrated by Figures 10, 11, 12, and 13. In Figure 10 we can see a typical TCP Tahoe congestion window behaviour. After a timeout or three duplicate acknowledgements, the window is reset to 1 and slow start is restarted. In Figure 11 this behaviour is modified by ATCP. If the congestion window was decreased because of duplicate acknowledgements, then it is restored after the next new acknowledgement arrives. After timeouts (not shown here) the congestion window is not restored. Figure 12 shows how TCP NewReno uses the Fast Recovery algorithm to avoid resetting the congestion window to 1 after duplicate acknowledgements. Finally, Figure 13 shows how ATCP in the NewReno implementation behaves similar to the ATCP in the Tahoe implementation.

4.2 Effect of full network load

The first thing that can be noted in the results is that creating a network load that is too high is not good for throughput performance. The data that is transferred is generated by a constant bit rate (CBR) agent that transmits data as fast as it is allowed to by the lower network layers. TCP makes estimates about how fast the network can transmit data and limits the rate at which it sends segments to the interface queue based on these estimates. Unfortunately, in these simulations the estimates are a bit too optimistic which results in growing interface queues.

For the simulations an IEEE 802.11 network is used which brings problems such as those described in Section 2.3.1. When more than one node wants to transmit at the same time one of them will have to wait. If this node has to wait for too long it will give up and report a send failure to the ad hoc routing protocol. Also, if a transmission fails because of errors caused, for example, by interference, a send failure will be reported.

Note that this is not congestion in the usual sense, which is that the interface queues are filled so that they must drop incoming packets. Send failures caused by contention for the transmission media can be seen as a sign of congestion at the link layer. However, it is difficult for the link layer to detect if a send failure is caused by such congestion or if it is caused by other reasons.

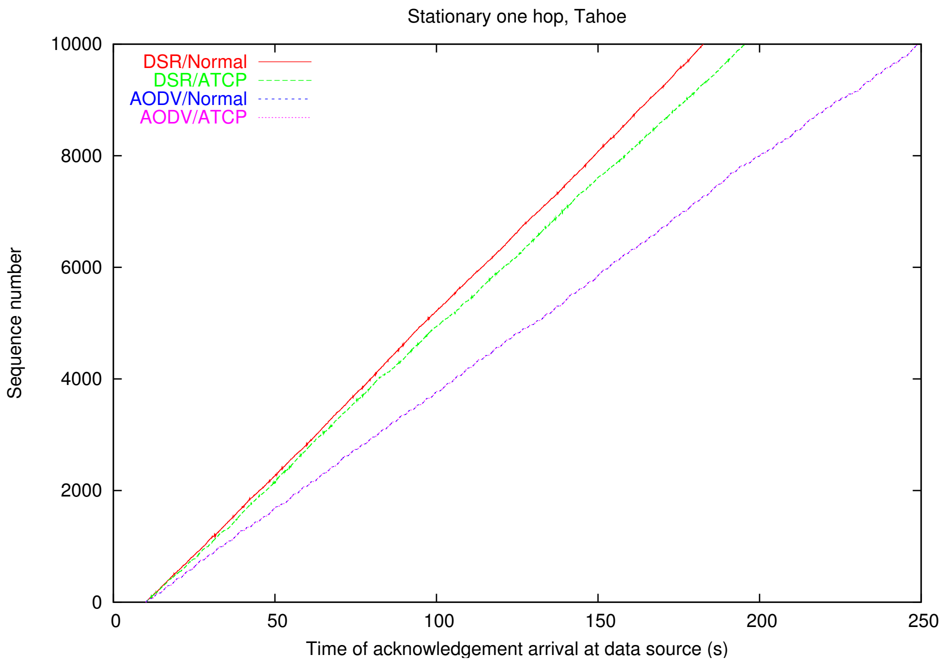

Figure 8: Stationary one hop. DSR and Tahoe with ATCP.

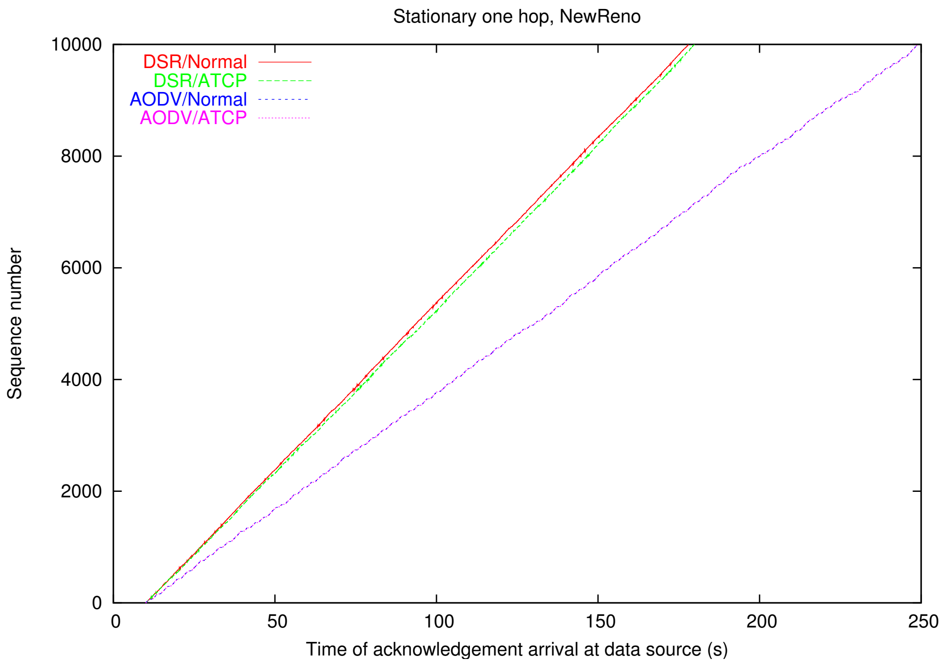

Figure 9: Stationary one hop. DSR and NewReno with ATCP.

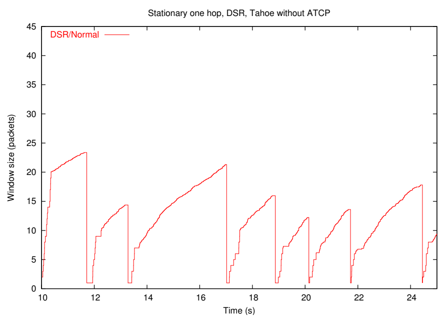

Figure 10: Stationary one hop. DSR and Tahoe 2 without ATCP.

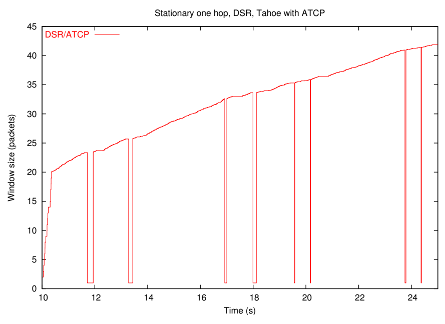

Figure 11: Stationary one hop. DSR and Tahoe with ATCP.

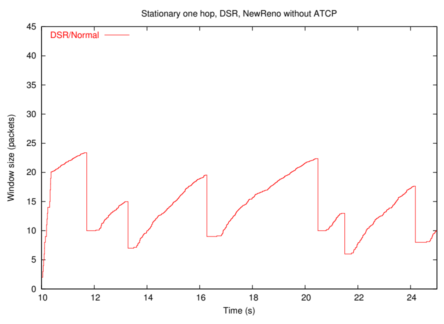

Figure 12: Stationary one hop. DSR and NewReno without ATCP.

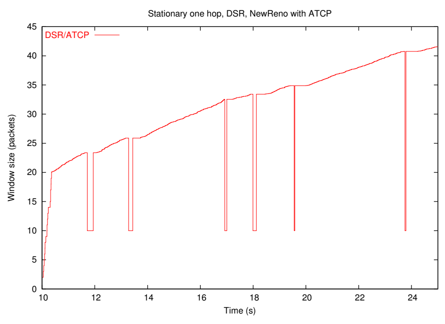

Figure 13: Stationary one hop. DSR and NewReno with ATCP.

4.3 Behaviour after send failures

When DSR or AODV receives a send failure report they both react by invalidating the route used for the transmission. A new route discovery process has to be finished before the route can be reestablished and used again.

When a node finds that a packet that it tries to transmit is associated with a broken route, the node has to decide what to do with the packet. Either it can “recover” the packet by requeueing it, usually at the end of the interface queue, or it can simply drop it.

Requeueing gives a risk of packet reordering and subsequent unnecessary retransmissions while dropping gives a risk of unnecessary retransmission timeouts. In different situations both DSR and AODV can be affected by any of these problems, but here a closer at the one hop scenario will be made.

4.3.1 DSR

DSR recovers packets that has originated at same node. Packets from other nodes are dropped.

In the one hop case this means that DSR will always try to recover the packets. If there are many packets in the queue, then all of them that are scheduled for transmission before the route can be reestablished will be requeued. This carries a risk of packet reordering if one set of packets are requeued so that they are transmitted later than another set of packets that were originally enqueued later.

Figure 14 shows how TCP Tahoe behaves after DSR has encountered a send failure. Each plus corresponds to a data packet enqueued at the source by TCP. The crosses show when the data packets arrive at the destination and the stars when the acknowledgements arrive at the source. Since the interface is always kept filled as much as the send window allows, it usually takes a while (queue delays of up to 0.2 s are not uncommon) between the enqueueing and the actual transmission (and reception), and yet some more time before the acknowledgement can travel back.

At 11.59 s a route break is caused by a send failure. During the broken route handling DSR requeues all packets destined through the route, which reorders the packets in the interface queue. The resulting packet reordering triggers the Fast Retransmit algorithm which decreases the congestion window size and enqueues one packet in the hope that it is the only one missing. Unfortunately, since there are still a lot of packets in the interface queue, it takes a while until this packet arrives and by that time the original packet, that was still in the queue, should already have arrived. As new acknowledgements start to arrive at 11.94 s the congestion window slowly starts to increase again. The net result is that a route break causes DSR to mix up the interface queue which leads to unnecessary retransmissions which is a suboptimal use of network resources.

Figure 15 shows the same events with ATCP enabled. The behaviour is the same as without ATCP up to 11.94 s when the first new acknowledgement arrives. Here the congestion window is restored to the value it had before the duplicate acknowledgements which allows the sender to instantly requeue a lot of packets that have timed out. The problem with this is that TCP can not react to the large jump in acknowledgements at 12.06 s, since TCP can not remove redundant packets from the interface queue. The conclusion is that restoring the congestion window too quickly, before the network problems are over and any following confusing is sorted out, can cause TCP to waste network resources.

Figures 16 and 17 show how TCP NewReno uses Fast Recovery by only resending one of the missing packets and waiting for the result before taking any further action. This allows TCP to react faster to the jump in acknowledgements at 12.06 s, and reduces the penalty of the ATCP implementation.

4.3.2 AODV

When AODV is used, a node recovers packets if the destination is closer in hop count, according to the broken route, than the source. Other packets are dropped. This is based on the assumption that is the packet has passed a majority of its path, then the trouble of recovering the packet will lower the total network load.

The effect in the one hop case is that AODV drops all packets that experience route breaks. This causes TCP to wait until the retransmission timer times out which gives a pause in the transmissions.

Figure 18 shows TCP Tahoe behaviour after an AODV route failure. At 11.59 s a route break occurs which causes AODV at node 0 to drop both packets and AODV at node 1 to drop acknowledgements. This requires TCP to wait for a retransmission timeout before it can decide that a segment has been lost and make a retransmission. After the timeout TCP sets its congestion window to 1 and enters slow start. TCP NewReno has the same behaviour as Tahoe after a retransmission timeout. ATCP enters the loss state after the timeout but at the first new acknowledgement it returns to normal state and the result is identical to when ATCP is not used.

4.4 Decreasing network load

Since a fully loaded network results in send failures and subsequent broken routes, a set of simulations using a slightly lower network load was run.

Figure 19 shows how a lower load results in interface queues that are emptied as fast as they are filled up. This gives transmission delays around 0.014 s which is much less than for the fully loaded network where the delays can sometimes pass 0.2 s. There are no problems with sending packets and no routes break so all TCP variants get exactly the same performance.

In this scenario it is the application layer that decreases the offered network load by setting a limit on how fast data is generated at the application layer. This means that the maximum throughput is limited by a layer that doesn't know any details of the true network capacity. This task is normally done by the transport layer,

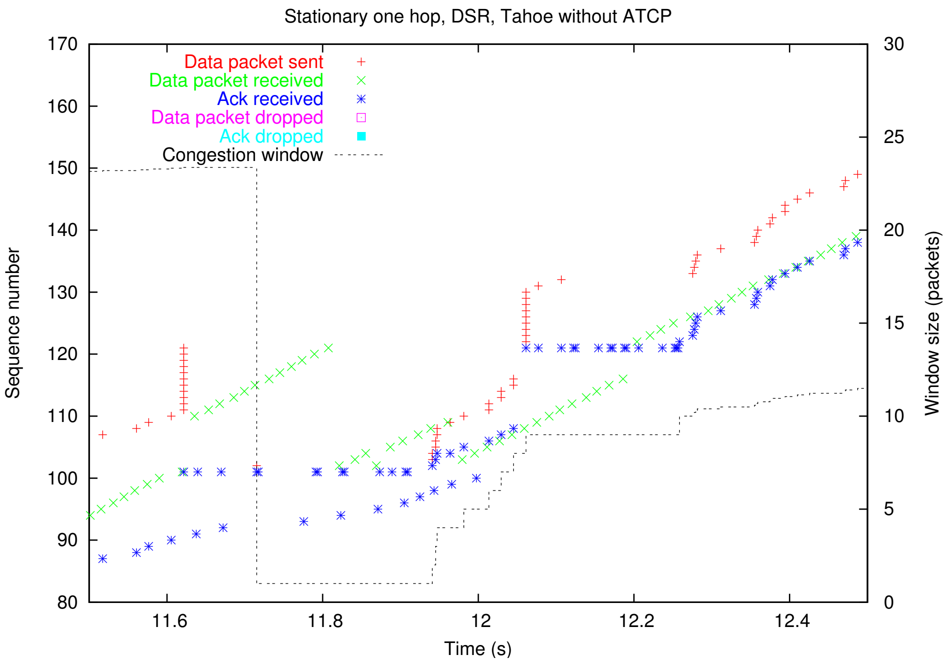

Figure 14: Stationary one hop. DSR and Tahoe without ATCP.

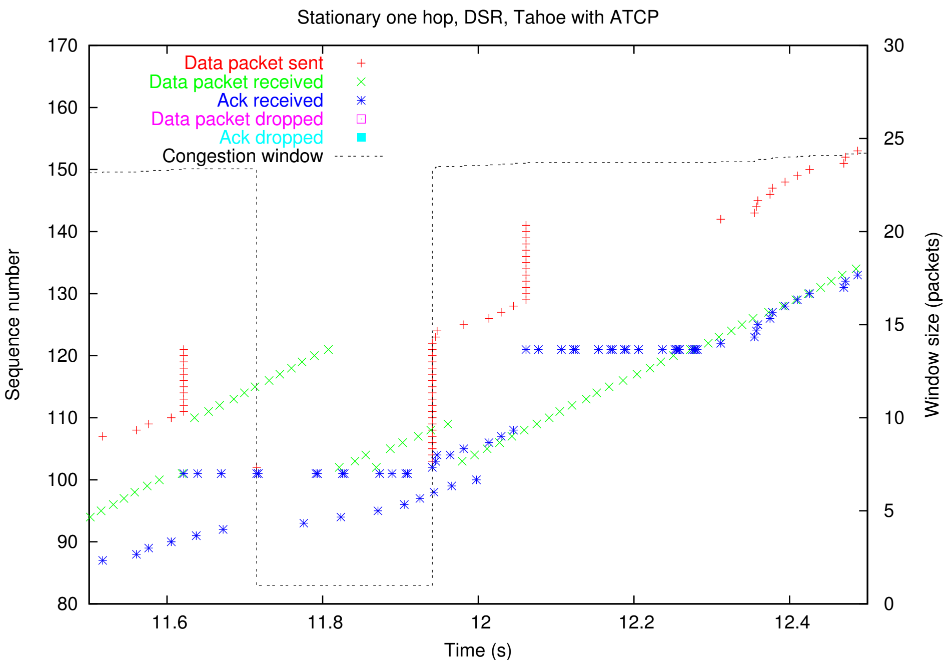

Figure 15: Stationary one hop. DSR and Tahoe with ATCP.

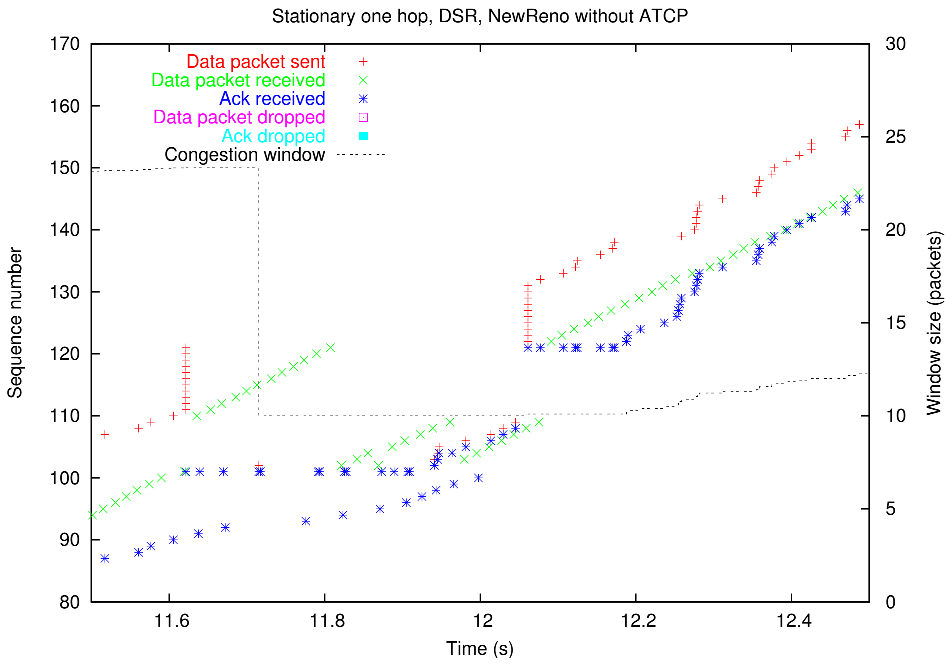

Figure 16: Stationary one hop. DSR and NewReno without ATCP.

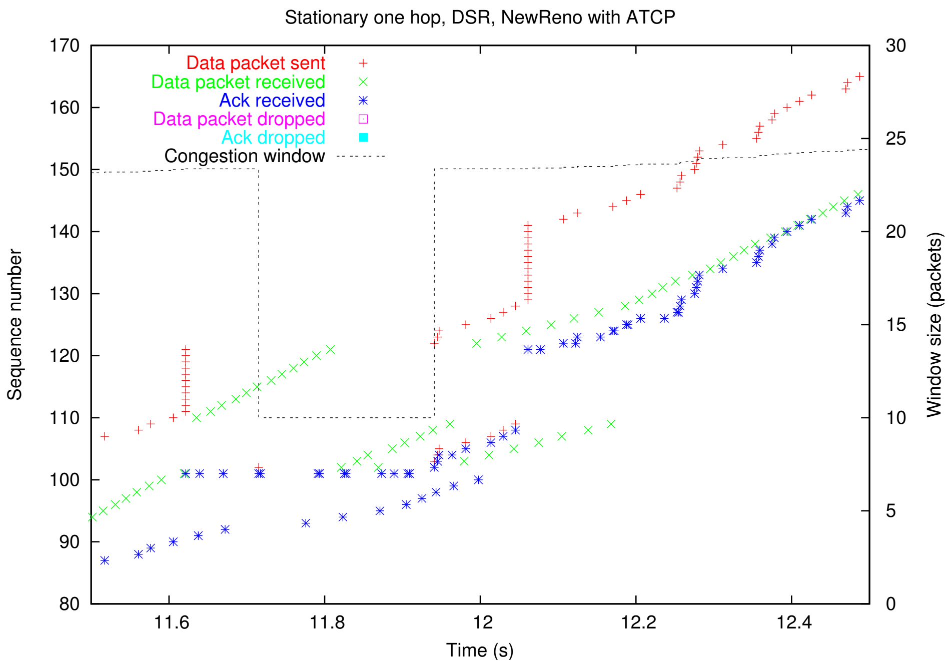

Figure 17: Stationary one hop. DSR and NewReno with ATCP.

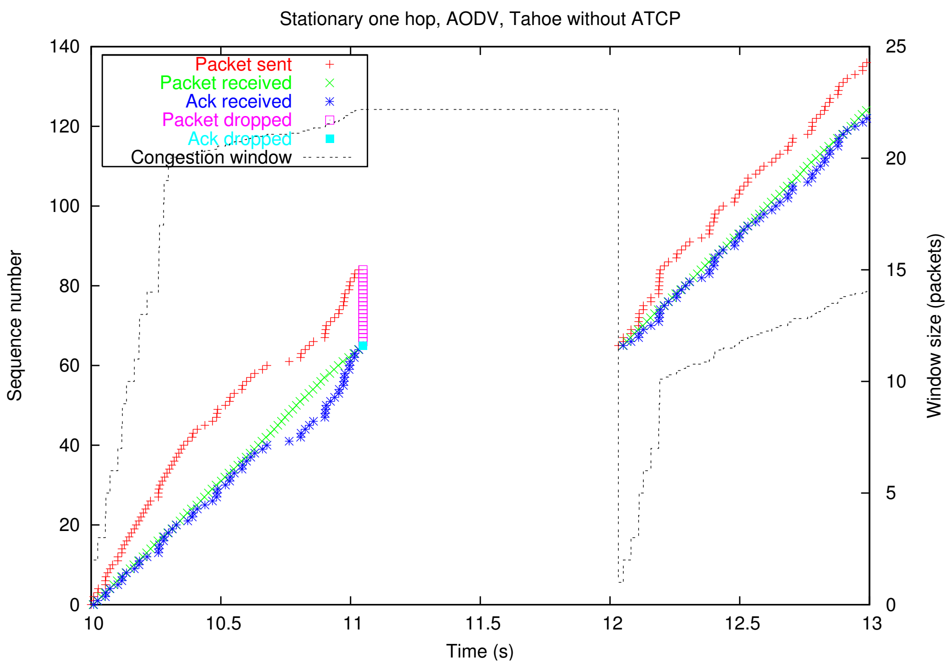

Figure 18: Stationary one hop. AODV and Tahoe without ATCP.

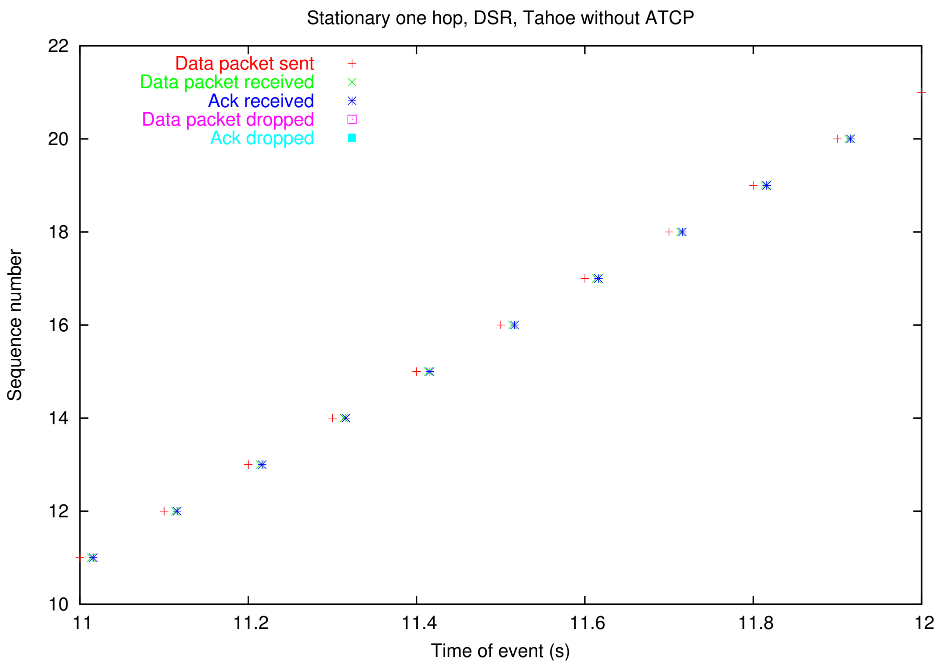

Figure 19: Stationary one hop. DSR and Tahoe without ATCP. Segment generation delay is 0.1 s.

i.e. TCP, which only accepts data from the application at a rate that it calculates that the network can manage.

Trying different delay values showed that a packet generation delay of 0.0161 s increased throughput by about 4% compared to simulations without a limit. Shorter delays resulted in interface queues that were filled up and longer delays resulted in the network being idle for short periods between the packets.

In these simulations this performance increase comes at the cost of having to estimate the network capacity at the application layer. Also, this scenario is very simple, in more complex scenarios the capacity may change over time, requiring the application to update its estimates. If instead TCP can do the rate limiting satisfactorily, then all applications using TCP will automatically benefit from it.

4.5 Varying packet error rate

When the packet error rate, PER, increases, throughput performance is naturally expected to drop. At full network load this is also the case in these simulations, but at lower load the performance decrease is very dependent on how the ad hoc routing protocol deals with the errors. If there is enough idle transmission time, then throughput can be increased by the routing protocol attempting to recover the packets.

Figures 20, 21, 22, and 23 show how throughput performance degrades under low network load (10 packets/s.), but with an increasing error rate. DSR, which in this case recovers the packets, can maintain the transfer rate even with up to 20% PER. AODV, which simply drops the packets, instead sees a huge decrease in throughput.

4.6 Increasing distance

The increasing distance scenario is very similar to the stationary one hop scenario. The only thing that lowers performance besides the protocol implementations is mobility. When the nodes move out of range all communication halts.

In all simulations the transmissions show a relatively even transfer rate until the nodes move more than 333 meters from each other. No successful reception is done at longer distances. Since the simulations use an ideal propagation model (the two-ray ground reflection model) and an ideal world with flat ground and no obstacles, this is probably a case of the reception power dropping below a certain threshold. The abrupt halt to communications is unrealistic; normally the probability of correct reception decreases gradually as the nodes move out of range of each other.

4.7 Increasing hop count

In general the transfer rate decreases as the destination node moves away from the source, thus forcing the packets to traverse more hops to reach it. Path lengths

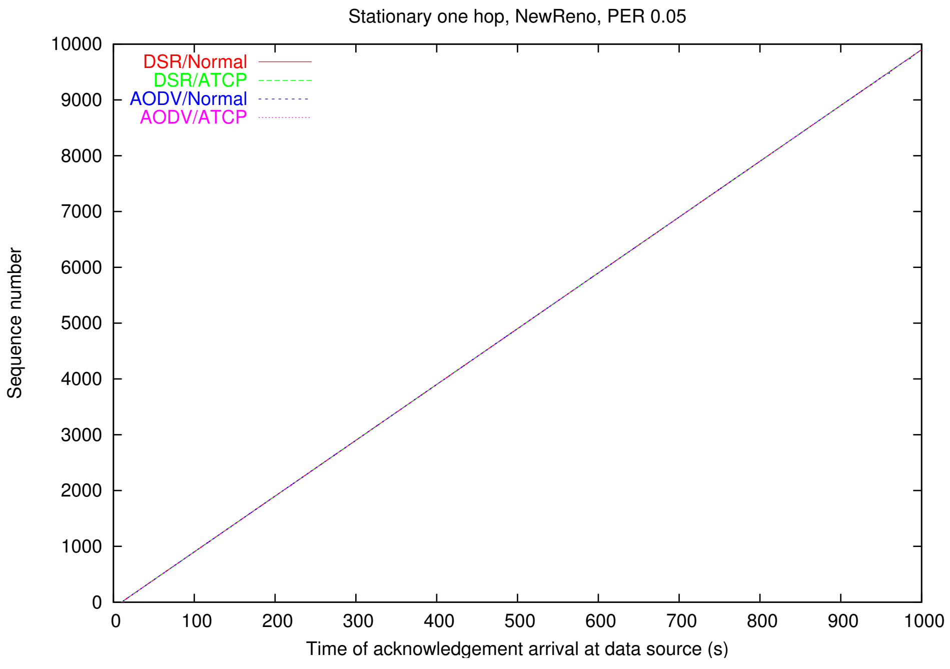

Figure 20: Stationary one hop. Load is 10 packets/s and PER is 0.05.

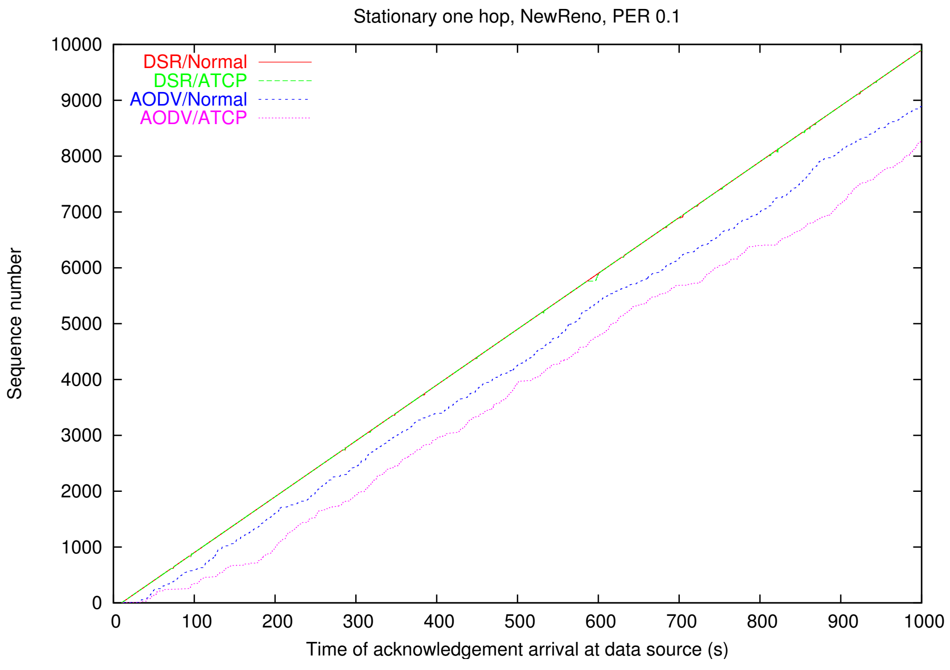

Figure 21: Stationary one hop. Load is 10 packets/s and PER is 0.1.

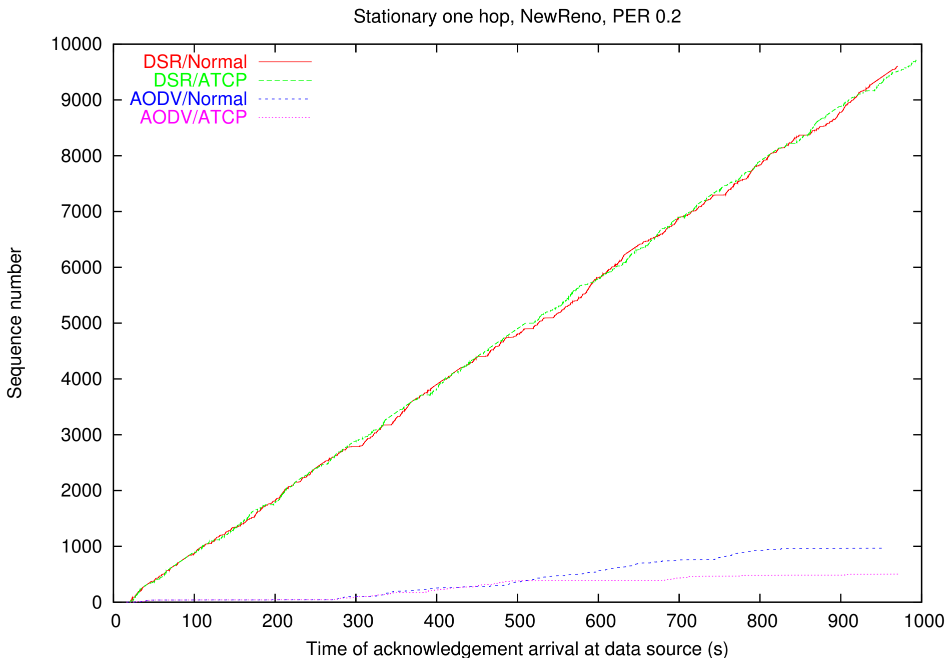

Figure 22: Stationary one hop. Load is 10 packets/s and PER is 0.2.

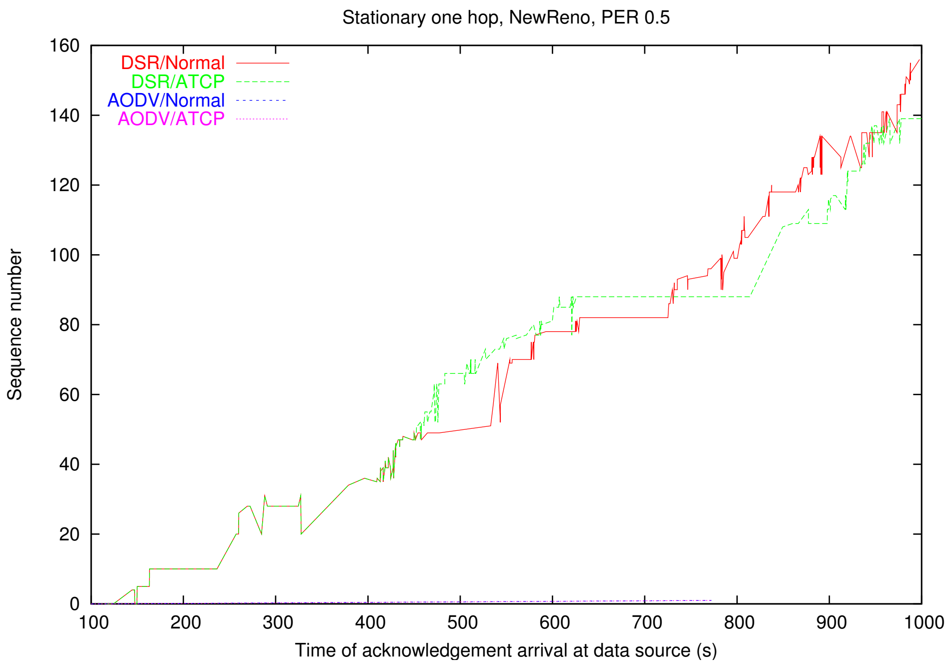

Figure 23: Stationary one hop. Load is 10 packets/s and PER is 0.5.

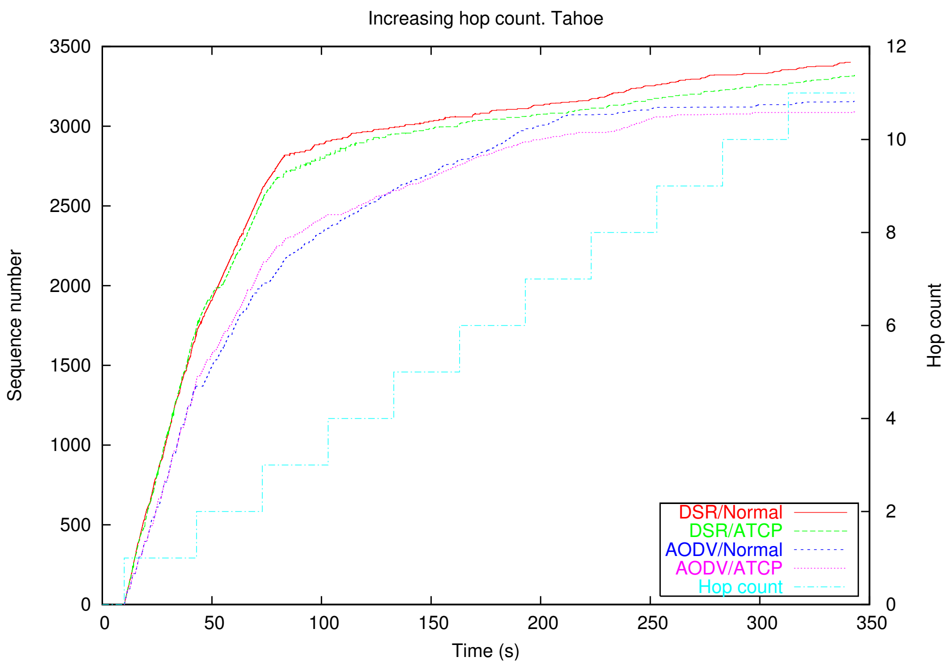

Figure 24: Increasing hop count, data transfer progress. Tahoe.

above 3 do not decrease the throughput much more because several transmissions can then be made simultaneously at different places along the path.

Since, as in the stationary one hop scenario, the network is loaded as much as possible, there is a high probability of send failures. When the number of hops increase, the probability for a send failure somewhere along the route increases. Since the ad hoc routing protocols often drops the packets in the interface queue when the route breaks, an increased hop count, especially in the presence of an increased packet error rate, gives a large throughput penalty.

In Figure 24 it can be noted that AODV performs better than DSR at path lengths over 2. This is because AODV recovers packets after route breaks if the destination was closer than the source, while DSR only recovers packets that originated at the same node.

4.8 Faking ICMP messages

Since ICMP messages were not implemented in ns-2, an alternative way to evaluate the effect of using the disconnected state was implemented. At explicit times in the simulation script, ICMP messages were faked which put ATCP into disconnected state. The times for the faked ICMP messages were found by looking at the trace files and locating the first sign of a route break. A faked ICMP message was added for that time and the simulation was rerun to find the next route break. This was repeated until all route breaks for the first 10 seconds were found.

4.8.1 DSR

Figures 25 and 26 shows the difference in behaviour with DSR. In Figure 25 the broken route is not detected by ATCP which leads to the normal behaviour. After three duplicate acknowledgements, loss state is entered and when a new acknowledgement arrives, ATCP returns to normal state.

Figure 26 shows what happens instead when a faked ICMP message is generated after the route break. At 12.77 s an ICMP message is faked which puts ATCP in disconnected state. This puts TCP in persist mode which sets the next sequence number to be sent to be the first unacknowledged segment. The sender then waits for a persist timeout to occur or for a new packet to arrive. When a new packet arrives at 12.80 s ATCP returns to normal state but it now starts transmitting at the first unacknowledged segment which causes a lot of unnecessary transmissions.

A side effect is the sequence of duplicate acknowledgements that arrive between 13.24 s and 13.49 s. These acknowledgements are not intended to be duplicates, instead they are messages telling that the destination node has already received the segments that were recently retransmitted. Since the transmission delay is so long these acknowledgements do not arrive until the sender has started transmitting higher sequence numbers and therefore they are misinterpreted as duplicate acknowledgements, causing even more unnecessary retransmissions. As a result, the throughput drops with 10%.

4.8.2 AODV

With AODV the faked ICMP messages do not affect the throughput much. Figure 27 shows that without an ICMP message, TCP will wait until a retransmission timeout before retransmitting. In Figure 28, ATCP receives a faked ICMP message and enters disconnected state. After a persist timeout it transmits a persist probe which, when the receiver acknowledges the segment, returns ATCP to normal state. Since the persist timer is almost the same as the retransmit timer, the result is that the throughput is almost the same as without the faked ICMP messages.

4.8.3 Modifications to increase throughput

Based on the results of the faked ICMP message simulations, two further modifications were tested. Neither of the modifications behave as either ATCP or normal TCP normally would. They both improve throughput in the stationary one hop scenario but they must be tested in more situations before they can be suggested for wider use. Also, they either depend on recovery of packets by the routing protocol or that the route is reestablished quickly after the route break.

On entry to disconnected state the variable that controls which segment to transmit next is decreased to the first unacknowledged sequence number. Since this decrease is unnecessary when DSR recovers packets after a route break, a simulation was run without it. Figure 29 show how ATCP enters disconnected state after the route break. This decreases the congestion window to 1 and when

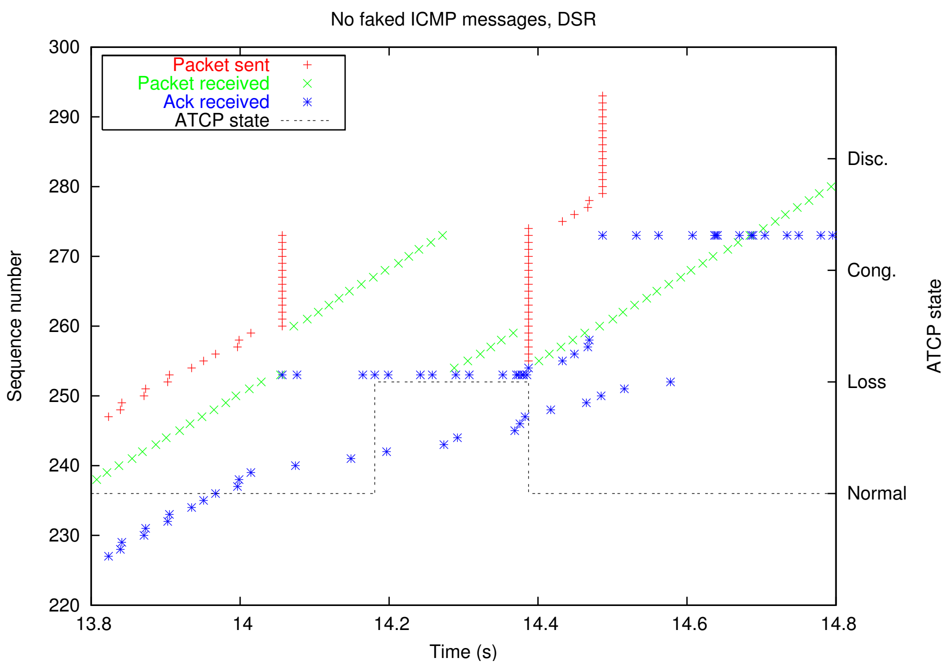

Figure 25: No faked ICMP messages. DSR and Tahoe with ATCP.

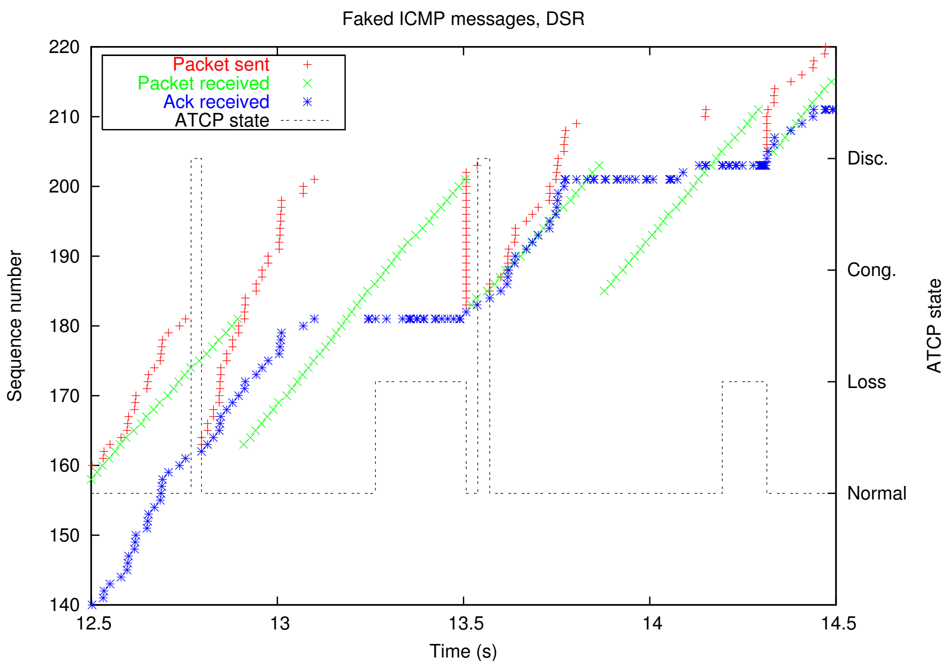

Figure 26: Effect of faked ICMP messages. DSR and Tahoe with ATCP.

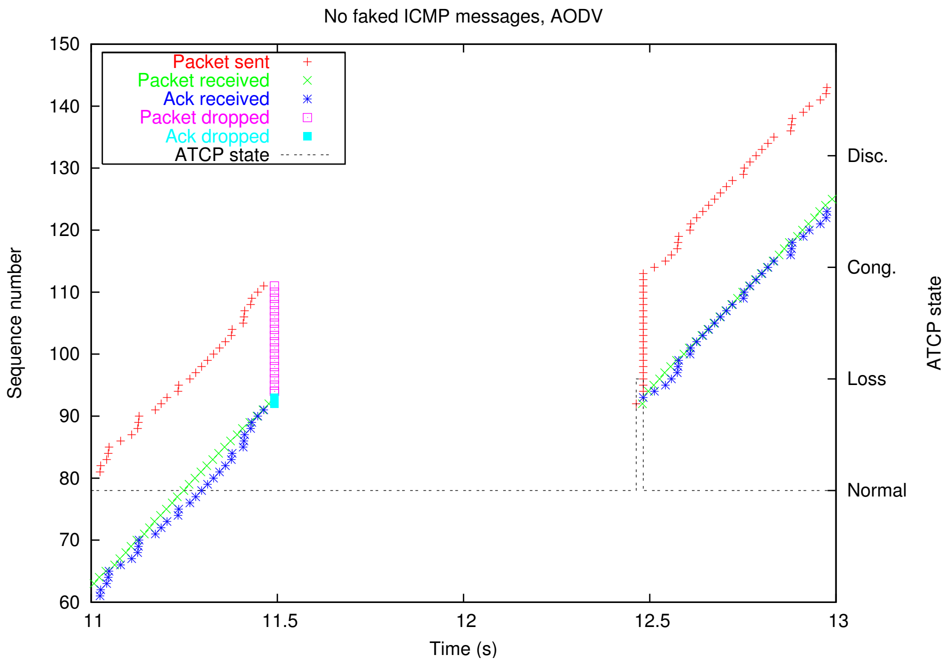

Figure 27: No faked ICMP messages. AODV and Tahoe with ATCP.

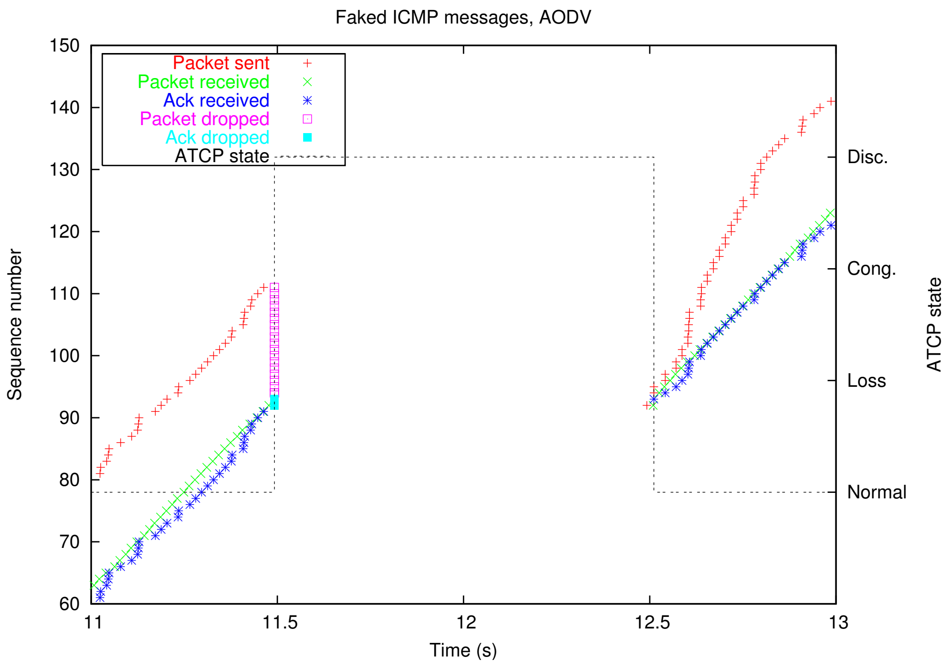

Figure 28: Effect of faked ICMP messages. AODV and Tahoe with ATCP.

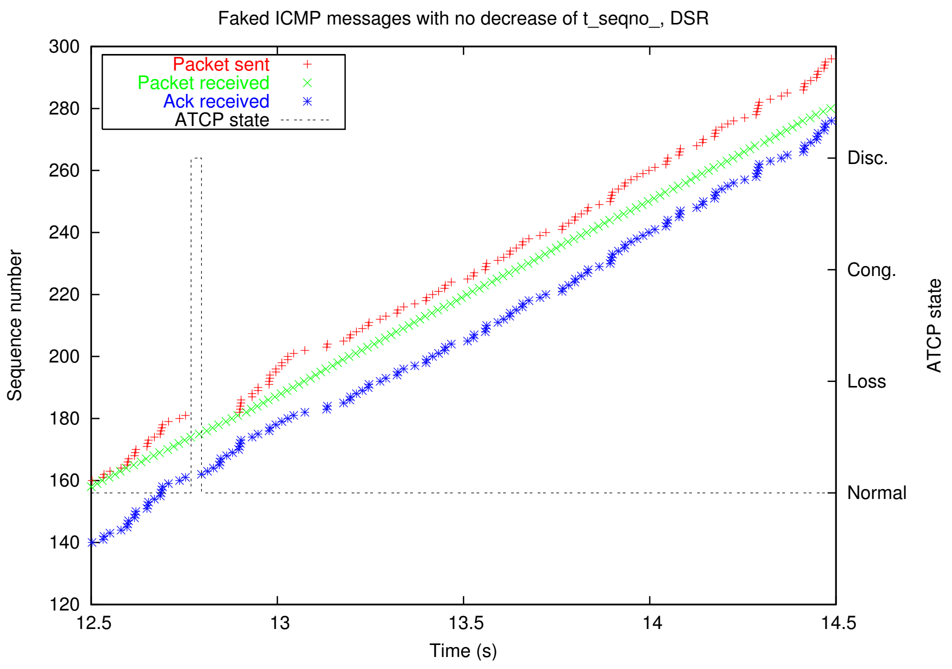

Figure 29: Faked ICMP messages with no decrease of sequence number. DSR and Tahoe with ATCP.

normal state is reentered, slow start is used. The effect is that the sender is briefly paused so that the link layer can catch up with the interface queue. After enough acknowledgements have arrived, the congestion window has grown enough to allow the TCP source to start sending more segments. This resulted in a throughput increase of around 9% compared to the simulation without faked ICMP messages.

For AODV, the problem was instead that TCP waited for a timeout until it could start sending again. To avoid this, ATCP was changed to do a retransmission immediately on entry to disconnected state. Figure 30 shows how this allows TCP to continue sending data with only a minimal interruption for route reestablishment. This gave a throughput increase of 48% compared to the simulation without ICMP.

4.9 Limiting the congestion window

An entirely different way, which is much simpler to implement than ATCP, of increasing throughput is to limit the congestion window size. Since a transmission in an IEEE 802.11 network can interfere with transmissions from other nodes at least two hops away, the network must be relatively large before more than one transmission can be made successfully at the same time. Because of this, using large congestion windows and queueing many packets is overly optimistic and will only result in longer delays.

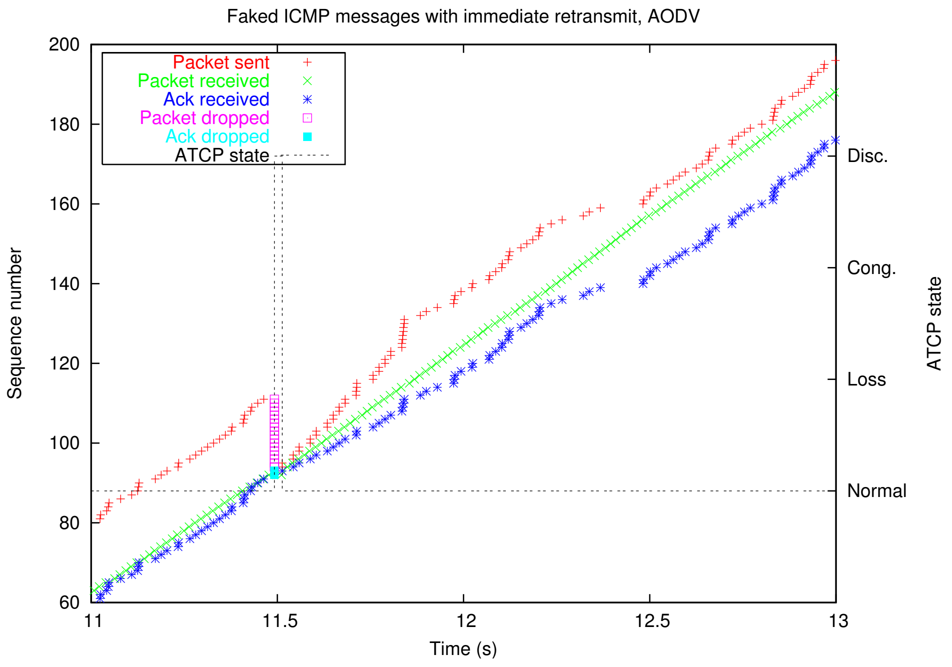

Figure 30: Faked ICMP messages and immediate retransmission. AODV and Tahoe with ATCP.

In Figure 31 an 8% increase in throughput can be seen with DSR when the congestion window is limited to values between 1 and 3 packets. A limit of 5 packets gives an increase of 7% and higher limits give smaller throughput increases.

Figure 32 shows how AODV reacts more to the congestion window size. A limit of 1 packet gives a 52% increase in throughput compared to when the size is unlimited. A 2 packet limit only gives an 18% increase and as the limit increases the throughput quickly approaches the level of the unlimited case.

The cause of the huge throughput increase for AODV is that in the one hop scenario AODV suffers much by every send failure that occurs. With a congestion window of 1, TCP has to wait for an acknowledgement after each transmitted segment, which effectively removes all contention for the transmission media between transmissions of data and acknowledgements. DSR shows a similar behaviour, but since it does not give as large performance decrease as AODV after route breaks, the effect is not as large.

If there had been competing traffic in the network or there had been more hops between the communicating nodes, then the results would have been different. However, this clearly shows that using larger congestion windows does not necessarily give higher throughput. Besides giving higher throughput rates, using a small congestion window leads to shorter round trip times since the packets do not have to wait in the interface queues for as long as with an unlimited window.

A problem with limiting the congestion window is that if there are other nodes

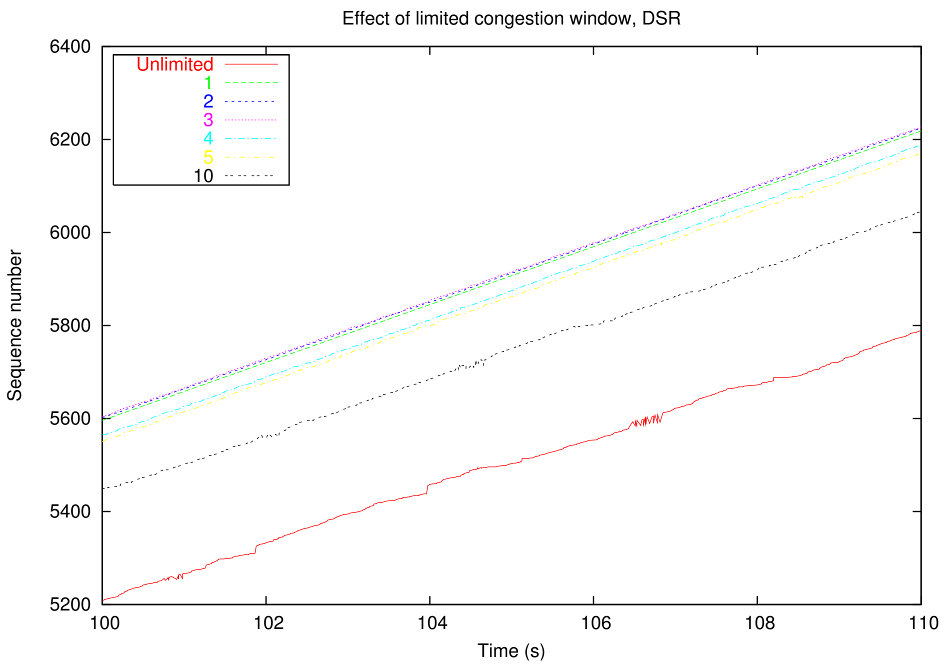

Figure 31: Throughput with different limits of the congestion window. DSR and Tahoe without ATCP.

present which do not limit their congestion windows, then the node with a limit will be at a disadvantage against unlimited nodes.

4.10 Comparison to experiments by Liu and Singh

The results from these simulations differ quite much from the results presented by Liu and Singh [25]. Therefore a comparison of the different conditions are presented here. The conditions are not described in detail by Liu and Singh so the comparison is based on what can be assumed from their paper.

4.10.1 Scenarios

The experiments by Liu and Singh include scenarios with bit error, congestion, partitioning, and packet reordering. In all cases they found a throughput increase when ATCP was used. Of these scenarios, only packet error was explicitly simulated for this paper. Packet reordering was experienced as a side effect in many simulations but with opposite results than for Liu and Singh.

Congestion and partitioning would have been interesting to simulate but this was prevented by lack of time. It is difficult to say how ATCP behaves in those situations without further simulations.

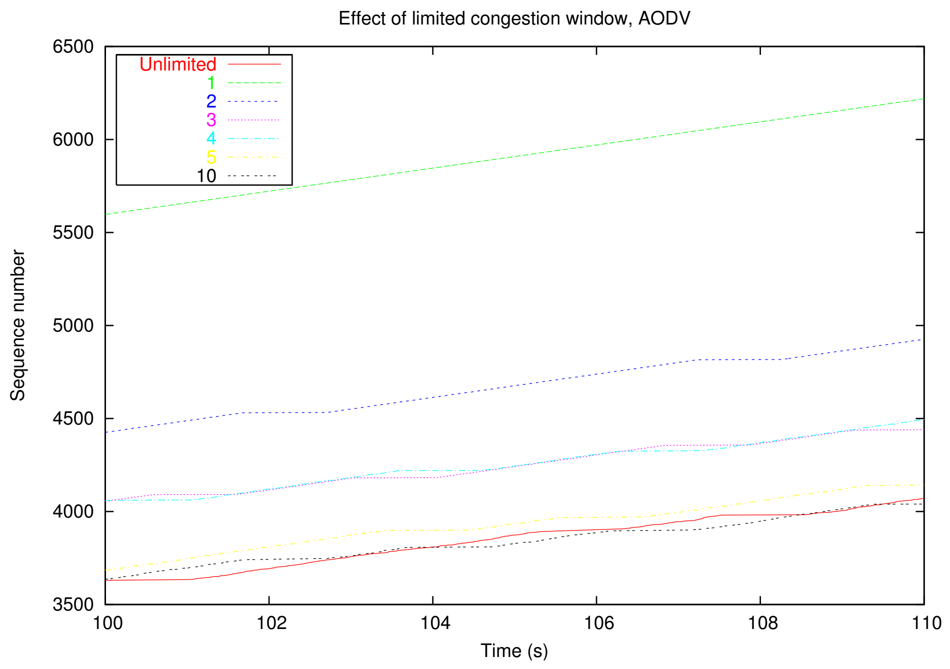

Figure 32: Throughput with different limits of the congestion window. AODV and Tahoe without ATCP.

4.10.2 FreeBSD implementation vs. Simulations

Liu and Singh conducted their experiments on a set of real computers running FreeBSD with their ATCP patches. Compared to ns-2 simulations where many parts are simplified this will show a more realistic behaviour. Generally, using real computers for experiments will give a more realistic result, but it is still important to know exactly which parts are involved and how they affect each other.

4.10.3 Network technology

Liu and Singh did not use real wireless interfaces for communication, but instead emulated them using normal Ethernet cards. The transmission delays and error rates were controlled by filters at the link layer. However, the behaviour of a real IEEE 802.11 network is more complicated than what they seem to have modelled. The simulated IEEE 802.11 does not necessarily show a real behaviour either, but it should still be closer than the emulation used by Liu and Singh.

- Simultaneous transmissions

  In an IEEE 802.11 network a radio transmission can interfere with other transmissions at least two hops away. The total transmission capacity over the entire network, as defined by how many packets can be successfully transmitted at the same time, will therefore be very low. Liu and Singh do not describe their setup so it can only be assumed that they have not taken transmission ranges into consideration. This will give their experiments a total network capacity of at least one packet per link. Also, if they have not disabled full duplex capability for the Ethernet cards, they will be able to both transmit and receive at the same time on a link which is impossible with radio interfaces.

The simulated IEEE 802.11 network uses a radio model to to determine the success of each transmission. The radio model may not be entirely correct, but it is still better than not taking interference into account at all.

- Loss and delay modelling

  Liu and Singh write that they modelled packet loss and delay by intercepting packets in the IP code when they are sent and received on the links. The delays are set by packet size and link speed and the loss is set to a constant bit error rate of 10^{-5}. They do not take into consideration the effects of interference by other nodes.

  In the simulations, the radio model determines if the transmission succeeds based on node distance, other transmissions, and surrounding environment.

- Effect of send failures

  Liu and Singh do not write about if the act of dropping a packet has any effects at the link layer in their experiments. In the ns-2 simulations, a transmission error will first cause retransmission attempts by IEEE 802.11. If IEEE 802.11 fails for too long, then it reports a send failure to the ad hoc routing protocol which causes the route to be invalidated.

  IEEE 802.11 has no method to determine if the send failure is caused by temporary problems, which should put ATCP in the loss state, or by true unreachability, which should put ATCP in the disconnected state. Liu and Singh tested the disconnected state by explicitly introducing route breaks and have not presented any method of discerning between loss and disconnection that can be used in wireless networks. This is a problem that must be solved before ATCP can be used.

4.11 Possible error sources

There are a number of factors that can affect the validity of the results.

4.11.1 Incomplete ns-2 implementation

Since ICMP messages and ECN marking have not been implemented there are some effects where the ATCP parts misinterpret the network state and therefore acts incorrectly.

On many occasions during the simulations there were send failures that caused route breaks. According to the ATCP specification the source should then enter disconnected state, but since it cannot detect this it will assume packet loss. The simulation results described in Section 4.8 shows that *not* entering disconnected state can actually give a better result than entering it, although this may not always be the case in other scenarios.

All packet drops that occurred were done by the ad hoc routing protocols because of route breaks. Since there were no congestion caused by full interface queues, the effect of not using ECN should be nonexistent. If there had been congestion the situation might have been different.

4.11.2 Implementation errors

There is always the possibility of errors in the implementation. Simulations run during the implementation of ATCP sometimes showed very different results. This is an indication of that TCP performance is very dependent on small details in the implementations.

Several of the results shown in this report are from simulations that were run with different snapshots of the code. However, when the code was changed so that it might have affected the results the simulations were rerun.

4.11.3 Simulation errors

The simulation specification scripts and data analysis scripts can contain errors.

The radio model and many other options that are used are the same as in most other wireless examples that can be found, so the results should be comparable to other ns-2 simulations.

4.11.4 Errors in ns-2

Ns-2 is by no means polished software. In each version old bugs have been fixed and new may have been introduced. There is no guarantee that there are no bugs in the current version (2.26) that affects the performance estimates.

4.11.5 Simplifications in ns-2

Besides any bugs that may be present in ns-2, there are a number of simplifications that are made in the simulator. Most notable for these simulations is that all commonly used TCP implementations in ns-2 are half-duplex and also use some other simplifications. Because of this, the TCP behaviour in ns-2 may differ from real TCP implementations.

5 Discussion

5.1 Problems encountered in the simulations

The problems encountered in these simulations are mainly caused by the interaction between TCP, the ad hoc routing protocol, and the interface queues. The characteristics of the IEEE 802.11 links also play a part in creating the problems.

- Many levels of error handling.

  IEEE 802.11 checks that each link layer frame is received properly and performs retransmissions when problems occur. If the retransmissions fail the link layer reports a send failure to the ad hoc routing protocol which in turn can discard the packet or requeue it for transmission with a new route. Besides this, TCP keeps track of packets at the transport layer level and performs retransmissions when it thinks it is necessary.

  The combination of using all these error handling mechanisms at the same time leads to less predictable results.

- Send failures caused by high network load.

  When there is a high network load in an IEEE 802.11 network, there is also a high probability of send failures. With DSR and AODV, each send failure causes the route it is using to be invalidated and forces the ad hoc routing protocol to restart the route discovery process.

- Packet reordering caused by route failures.

  When a route is invalidated, then transmissions of other packets in the interface queue that are using that route will also fail until the route discovery process has finished successfully. The failed transmissions will cause the packets to be either dropped or requeued for transmission with a new route. If the packets are requeued, there is a large risk of them arriving in the wring order at the destination. This can in turn lead to packets being retransmitted by TCP.

5.2 Possible solutions

Here are a few ideas for avoiding or decreasing the effect of send failures:

- Decrease the amount of send failures.

  - Use another link layer technology.
    Since an overloaded network using IEEE 802.11 makes send failures more common, it could be wise to examine other link layer technologies that behave better under high load. Technologies that use transmission schedules, e.g. those based on TDMA, suffers much less from collisions and subsequent send failures.

- Make TCP more aware of network capacity.

  If we still want to use IEEE 802.11, it would be good if TCP could be made more aware of how fast the network can send packets. By knowing the capacity and trying not to exceed it, a better behaviour could be achieved. This can be a difficult task since it is desired that TCP can be used with many different link layer technologies with different characteristics. Also, the gain would only be partial if only one of the connection endpoints use an improved TCP.

- Limit the congestion window.

  Since large congestion windows caused a high network load, the congestion window can be limited. If all TCP implementations in the network use smaller congestion windows, then a smaller number of packets will be present in the network at the same time which will give shorter delays where packets wait in interface queues. Shorter round trip times will also make retransmission timeouts faster which will make TCP react faster to missing acknowledgements. Note that there is a risk of unfairness where nodes with a limited congestion window is at an disadvantage against nodes without a limit.

• Decrease the effect of send failures.

  - Use sorted interface queues.

    In these simulations, both AODV and DSR use simple FIFO queues for the interface queues. Instead of this, the queues could be designed so that they try to transmit packets in the correct order. This would remove most packet reordering that occurred in these simulations.

  - Discard retransmissions of packets still in queue.

    Some of the observed performance decreases are caused by unnecessary retransmissions. When a packet in enqueued at the link layer a check could be made to see if an identical packet is already enqueued. If so, the new packet can safely be discarded. This would remove some of the performance decrease caused by reordered packets.

  - Prioritise route discovery packets.

    Some of the requeueing observed in these simulations could be avoided by giving priority to route discovery packets after a route break. This would avoid trying to send packets that precedes the route discovery packets in the interface queue. Trying to send a packet before a new route is established will only result in the packet being dropped or requeued.

5.3 Insights from the work

5.3.1 TCP behaviour changes

Changing the behaviour of TCP is tricky. Changing something to increase performance in certain cases can cause performance penalties in other cases. For example it could be assumed that larger congestion windows would cause a throughput closer to the maximum network capacity. However, in an error-prone network like the MANETs, the increased congestion window instead reduces the protocols ability to react quickly to errors like lost packets or packet reordering. Those who want to implement changes should be both familiar with the details of the protocol and with the source code of the implementation that they want to change.

5.3.2 Ns-2

Ns-2 is a large and complicated piece of work. Many network researchers have contributed to create a program that can simulate many different aspects of networks. Unfortunately, even though the entire source code is available, it is quite difficult to make more than small changes to it. Many parts of the program are poorly documented and it is hard to get good overview of its overall design.

5.3.3 Simulation results

Even though these simulations seldom showed increased throughput with ATCP, increased knowledge of the network state can still be used to increase performance by reacting to different situations in proper ways. Even though the transport layer is often said to have little knowledge about the lower layers, the current TCP implementations do probe the network and they make assumptions about its state; for example by changing the congestion window. While the TCP implementations of today are usually tuned for wired, low-loss networks, future implementations could be more adapted to other network profiles.

5.3.4 Network state

To what extent should network state information be explicitly passed around the network? This is a difficult question. Even though it can be used by the nodes to make more intelligent decisions, it takes resources from the real data transfers. The benefit of knowing more the network state must be weighed against the cost of learning about it. In a world of finite resources, smarter does not always mean faster.

5.4 Future work

Here are some suggestions for future work:

- Implementing solutions to problems encountered in the simulations.

  One or more of the ideas suggested in section 5.2 could be implemented and evaluated. This could be done independent of other TCP modifications.

- Implementing ATCP fully in ns-2.

  It would be interesting to redo these simulations and compare them to a complete ATCP implementation in ns-2. In that way the effects of skipping parts of ATCP could be observed.

- Doing more extensive simulations of different scenarios.

  The simulations made for this paper are very simple and can mostly be seen as a proof that the implementations are runnable without crashing the simulator. To be able to draw any real conclusions about the effects of ATCP more complex situations must be simulated. The next step in complexity would be adding competing traffic parallel to or crossing the observed path.

- Studying other TCP modifications.

  ATCP is not necessarily the best choice for MANETs. Other TCP modifications like TCP-F [5] or TCP-Bus [22] could be implemented and studied.

- Studying alternatives to TCP.

  Even if TCP is the most commonly used protocol, there are other protocols, e.g. SCTP [42], that could be used in MANETs. One of these could be implemented and compared to TCP.

6 Conclusions

In this report, TCP and ATCP behaviour has been studied in a MANET environment. Even though the ATCP implementation is not complete and the simulation scenarios are simple, the results still show that there can be a large throughput decrease when TCP behaves in ways that are inappropriate for different MANET conditions.

In these simulations, the largest problems were caused by the interaction between TCP, the ad hoc routing protocol, and the IEEE 802.11 network. When the IEEE 802.11 is at full load, the probability of send failures increases. These send failures cause route breaks, which in turn can cause either packet reordering or packet loss. This causes TCP to either make unnecessary retransmissions or unnecessarily wait for a retransmission timeout, which in both cases decreases throughput.

Some ideas on how to make the throughput decrease smaller have been presented.

- Use a link layer technology that performs better under full load than IEEE 802.11.

- Set a maximum size limit on the TCP congestion window.
- Improve the link layer queue behaviour.

In general, when the partial ATCP implementation was used, throughput performance decreased. This runs opposite to the results presented by Liu and Singh [25] where ATCP increased throughput drastically. The simulations for this report were too simple to draw any definite conclusions about the suitability of ATCP, but there are also some questions about the validity of Liu and Singh's results.

References

[1] M. Allman and A. Falk. On the Effective Evaluation of TCP. *ACM Computer Communication Review*, 29(5):59–70, October 1999.

[2] M. Allman, V. Paxson, and W. Stevens. RFC 2581: TCP Congestion Control. http://www.ietf.org/rfc/rfc2581.txt, April 1999.

[3] Bluetooth. The Official Bluetooth Wireless Info Site. Web page: http://www.bluetooth.com/, September 2002.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, 1998.

[5] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback-based scheme for improving TCP performance in ad hoc wireless networks. *IEEE Personal Communications*, 8(1):34–39, February 2001.

[6] T.-W. Chen and M. Gerla. Global state routing: a new routing scheme for ad-hoc wireless networks. In *Proceedings of IEEE International Conference on Communications, ICC 98*, pages 171–175, June 1998.

[7] I. Chlamtac, A. Faragó, A. D. Myers, V. R. Syrotiuk, and G. Záruba. ADAPT: a dynamically self-adjusting media access control protocol for ad hoc-networks. In *Global Telecommunications Conference, 1999. GLOBECOM '99*, pages 11–15, December 1999.

[8] L. M. Feeney. A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks. Technical Report T1999:07, Swedish Institute of Computer Science, October 1999.

[9] S. Floyd and T. Henderson. RFC 2582: The NewReno Modification to TCP's Fast Recovery Algorithm. http://www.ietf.org/rfc/rfc2582.txt, April 1999.

[10] S. Floyd and V. Paxson. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, February 2001.

[11] GloMoSim. Web page: http://pcl.cs.ucla.edu/projects/glomosim/, August 2002.

[12] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu. Preemptive routing in Ad Hoc networks. In *Proceedings of the seventh annual international conference on Mobile computing and networking*, pages 43–52, 2001.

[13] D. Groten and J. R. Schmidt. Bluetooth-based mobile ad hoc networks: opportunities and challenges for a telecommunications operator. In *IEEE VTS 53rd Vehicular Technology Conference, VTC 2001 Spring*, pages 1134–1138, May 2001.

[14] N. Gupta and S. R. Das. A capacity and utilization study of mobile ad hoc networks. In *26th Annual IEEE Conference on Local Computer Networks, LCN 2001*, pages 576–583, November 2001.

[15] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[16] Z. J. Haas, M. R. Pearlman, and P. Samar. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-zrp-04.txt, July 2002. Work in progress.

[17] IEEE. IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. http://standards.ieee.org/getieee802/, 1999.

[18] IEEE. Supplement to 802.11-1999, Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band. http://standards.ieee.org/getieee802/, 1999.

[19] IEEE. Get IEEE 802 home page. Web page: http://standards.ieee.org/getieee802/, July 2002.

[20] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206, August 1999.

[21] D. B. Johnson, D. A. Maltz, Y.-C. Hu, and J. G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, February 2002. Work in progress.

[22] D. Kim, C.-K. Toh, and Y. Choi. TCP-BuS: improving TCP performance in wireless ad hoc networks. In *2000 IEEE International Conference on Communications, ICC 2000*, pages 1707–1713, June 2000.

[23] J. Li, C. Blake, D. S. D. Couto, H. I. Lee, and R. Morris. Capacity of Ad Hoc wireless networks. In *Proceedings of the seventh annual international conference on Mobile computing and networking*, pages 61–69, 2001.

[24] W.-H. Liao, J.-P. Sheu, and Y.-C. Tseng. GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks. *Telecommunication Systems*, 18(1-3):37–60, 2001.

[25] J. Liu and S. Singh. ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1300–1315, July 2001.

[26] LUNAR - Lightweight Underlay Network Ad hoc Routing. Web page: http://www.docs.uu.se/docs/research/projects/selnet/lunar/, July 2002.

[27] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018: TCP Selective Acknowledgment Options. http://www.ietf.org/rfc/rfc2018.txt, October 1996.

[28] R. Nelson and L. Kleinrock. Spatial TDMA: A collision-free multihop channel access protocol. *IEEE Trasactions on Communication*, 33(9):934–944, September 1985.

[29] The Network Simulator - ns-2. Web page: http://www.isi.edu/nsnam/ns/, July 2002.

[30] OPNET Technologies. Web page: http://www.opnet.com/, July 2002.

[31] M. Östergren. TCP Performance in Ad Hoc Networks. Technical Report T2000:14, Swedish Institute of Computer Science, November 2000.

[32] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[33] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-10.txt, January 2002. Work in progress.

[34] C. E. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proceedings of the conference on Communications architectures, protocols and applications, SIGCOMM '94*, pages 234–244, August 1994.

[35] J. Postel. RFC 768: User Datagram Protocol. http://www.ietf.org/rfc/rfc768.txt, August 1980.

[36] J. Postel. RFC 793: Transmission Control Protocol. http://www.ietf.org/rfc/rfc793.txt, September 1981.

[37] T. S. Rappaport. An Introduction to Indoor Radio Propagation. Web page: http://www.sss-mag.com/indoor.html, January 2001.

[38] S. Singh and C. S. Raghavendra. PAMAS: Power Aware Multi-Access protocol with Signaling for Ad Hoc Networks. *Computer Communication Review*, 28(3), July 1998.

[39] P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '99*, pages 202–209, March 1999.

[40] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1998.

[41] R. Stewart and C. Metz. SCTP: New Transport Protocol for TCP/IP. *IEEE Internet Computing*, 5(6):64–69, November 2001.

[42] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauerand, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. RFC 2960: Stream Control Transmission Protocol, October 2000.

[43] Z. Tang and J. J. Garcia-Luna-Aceves. Collision-avoidance transmission scheduling for ad-hoc networks. In *2000 IEEE International Conference on Communications, ICC 2000*, pages 1788–1794, June 2000.

[44] S. Xu and T. Saadawi. Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks. *Computer Networks*, 38(4):531–548, March 2002.

IMIT/LCN 2003-11