Master thesis performed at
Research Department
Ericsson Eurolab Deutschland GmbH
Herzogenrath, Germany
August 4, 1999

TCP Reaction to Rapid Changes of the Link Characteristics due to Handover in a Mobile Environment

Mattias Ronquist
School of Electrical Engineering and Information Technology
Royal Institute of Technology (KTH)
Stockholm, Sweden

Supervisor: Dipl. Inform. Stephan Baucke
Examiner: Prof. Gerald Q. Maguire Jr.

Abstract

The Transmission Control Protocol (TCP), used in the Internet, was not developed for a mobile, wireless environment. One reason why TCP might encounter problems in such an environment is rapid changes of the link characteristics. These rapid changes can occur due to handover between two subnetworks (macro handover), e.g., when a mobile node switches between different access networks. A possible and realistic handover scenario could be when a mobile node is roaming between a high bandwidth local area network (LAN) with limited coverage and a low bandwidth radio link with wide area coverage.

The goals of this thesis were to set up a realistic environment for measurements of the handover performance of TCP, and to observe TCP behavior when the link characteristics suddenly change. Further objectives were to analyze the results and propose solutions for improving the performance.

The mobility management in the measurement setup is handled by Mobile IP. Handovers are performed between a wireless LAN (WaveLAN) and a PPP link over a GSM circuit switched data connection. Our investigation shows that several spurious TCP timeouts occur after handover from the fast link to the slow link, triggering unnecessary retransmissions and hence resulting in TCP performance degradation. To avoid unnecessary retransmissions, we suggest a resetting of the retransmission timeout value (RTO) at the moment of handover.

In the case of handover from the low bandwidth link (PPP) to the high bandwidth link (WaveLAN), our measurements show that queued packets in the link layer buffer continue to flow over the PPP link even after the handover. The high bandwidth available after the switch is thus poorly utilized before the buffer of the low bandwidth link has been emptied. The IP sending process should delay putting packets in the queue of a slow link, thus avoiding large link layer queues and enabling utilization of the high bandwidth link faster. This could be achieved by flow control between the IP layer and the link layer. After the packets have started flowing over the WaveLAN, the RTO value is unnecessarily high, which could result in extensive delays in the case of packet losses. To alleviate the problems we recommend resetting the RTO value or modifying the algorithm for calculating the RTO value to faster adapt to sudden and significant decrease of the round-trip time (RTT) in the case of handover.

In both handover scenarios mentioned above we have found that a small window size is favorable to mitigate the negative effects due to the rapid changes of the link characteristics. The use of Active Queue Management to avoid large window sizes would be an interesting approach for future investigations. Another interesting approach could be to have flow control between the IP layer and the link layer to avoid a large link layer queue when the handover from

Table of Contents

CHAPTER 1

Introduction

The number of users of the Internet has been growing enormously the last decade. An increasing number of people are using the Internet on a daily basis, browsing the World Wide Web, sending and receiving email, downloading files, etc. The users only have access to the Internet at some fixed points such as in their office, at home or at the university. However, nowadays light weight portable laptops are no longer a rarity, which has created a demand for mobile computing and networking. It has become desirable to have access to the same services with the same functionality and quality, regardless of if the users are at their fixed access point or away from it, and maybe even while on the move.

The set of rules for communication over the Internet, the TCP/IP protocol suit [4] was not created and developed for mobile, wireless communication. To achieve mobility for computers, i.e. to make it possible for computers to maintain connection to the Internet even when moving around, Mobile IP [1, 2, 28] was developed, a standard proposed by the Internet Engineering Task Force. Although Mobile IP enables mobile communication, there are many problems to solve and improvements to make before mobile wireless communication can function as smoothly with the TCP/IP protocol suit as the conventional wired communication with a fixed network topology does.

For a future IP based mobile network, *handover*[1] performance of Mobile IP is considered a critical factor. Improving the handover performance on the network layer is the focus of ongoing work at Ericsson Research. Part of these activities is to study the effects of IP mobility on the Transmission Control Protocol (TCP) [26]. The performance of TCP is very important since TCP is used by many popular applications on the Internet, such as World Wide Web access, electronic mail, file transfer, telnet, etc.

There are three main reasons why problems for TCP might occur in a wireless mobile environment:

- The unreliable characteristics of wireless links
- The handover delay
- Rapid changes of the link characteristics

This thesis is limited to the third reason. It is already known that the unreliable characteristic of wireless links and the handover delay have a negative impact on TCP performance regarding the throughput and link utilization (see Chapter 4). While approaches exist to alleviate the effects of packet losses and delays not much is known about effective ways to make TCP quickly adapt to rapidly changing link characteristics.

A possible and realistic handover scenario is when a mobile node is roaming between a high bandwidth local area network (LAN) with limited coverage and a low bandwidth radio link with wide area coverage. One conceivable future trend is that wireless network access will be available everywhere. Several providers offering different types of access networks will result in more heterogeneous wireless communications systems. This makes it interesting to study the TCP

[1] A handover is a necessary information exchange to redirect the data flow to the mobile node's new location. In Mobile IP this redirection is achieved by the registration procedure (see Section 2.4)

behavior and identifying the problems when roaming between different access networks with different characteristics.

To observe the TCP behavior, measurements will be made. The measurement environment used will comprise a mobility server and a mobile client. The server will be equipped with a wireless LAN (WaveLAN) and a GSM circuit switched data connection for access by the mobile client. The reason for choosing a WaveLAN and a GSM connection is the different characteristics of the two access media regarding bandwidth and round-trip time. GSM provides a reliable wireless link when it uses a reliable link protocol, the *Radio Link Protocol* (RLP). In WaveLAN errors and packet losses are rare provided a relatively small distance between the sender and receiver (see [29] pp.20-22). Mobility management on the server and client will be handled by Mobile IP.

The goals of this thesis are to:

- Set up a suitable and realistic measurement environment for measurements of the handover performance of TCP.
- Perform measurements monitoring the behavior of TCP when rapid changes to the link characteristics occur due to handover between two different access media.
- Evaluate the results and identify the problems.
- Suggest possible solutions to the problems.

CHAPTER 2

IETF Mobile IP

The Internet Protocol (IP) [19] is the protocol that determines how packets are routed through today's Internet to their destinations according to their IP addresses. An IP address is divided into two parts; the network ID and the host ID. The network ID informs the routers in the Internet of which network a node (a host or a router) is attached to. If a node is not located on the network indicated by its IP address, datagrams destined to the node, will be undeliverable. Hence, if a node changes its point of attachment to another network, the node has to change its IP address. However, changing the IP address makes it impossible to maintain transport and higher-layer connections (as mentioned in Section 3.1, a TCP connection is uniquely identified by the port numbers and the IP addresses of the two connection endpoints).

Mobile IP [1] is a protocol proposed by the Internet Engineering Task Force (IETF). Mobile IP allows a node to keep the same IP address wherever it is connected to the network, making it possible to maintain transport and higher-layer connections while moving around.

This section intends to give an introduction to the IETF Mobile IP protocol by explaining the main features of the protocol. [1, 2, 28] provides further information and details about the protocol.

2.1 Overview

Mobile IP requires special mobility functionality only in the three entities listed below. The definitions of the entities are taken from [2].

- **Mobile node** – A mobile node is a host or a router that changes its point of attachment from one network or subnetwork to another. A mobile host may change its location without changing its IP address. It may continue to communicate with other Internet nodes at any location using its (constant) IP address, assuming link-layer connectivity to a point of attachment is available.
- **Home agent** – A home agent is a router on a mobile node's home network that tunnels datagrams for delivery to the mobile node when it is away from home and maintains current location information for the mobile node.
- **Foreign agent** – A foreign agent is a router on a mobile node's visited network that provides routing services to the mobile node while registered. The foreign agent detunnels and delivers datagrams to the mobile node that were tunneled by the mobile node's home agent. The foreign agent may always be selected as a default router by the registered mobile nodes.

Each mobile node has a static IP address, called the *home address* which remains unchanged regardless of where the node is attached to the Internet. The network ID of the home address indicates its *home network*. The packets destined to the mobile node are always routed to its home network. When the mobile node is attached to a *foreign network*, i.e. any network other than the mobile node's home network, the *home agent* intercepts all the packets destined to the mobile host and arranges to deliver them to the mobile node's *care-of address*. The care-of address changes at each new point of attachment and is associated with the current foreign network where the mobile node can be reached.

To be able to forward packets to the mobile node, the home agent needs to know the location of the mobile node, i.e., the care-of address, so whenever the mobile node moves, it registers its new care-of address with the home agent. The forwarding of a packet from the home agent to the mobile host requires modification to the packet so that it can be delivered at the care-of address. The home agent constructs a new IP header with the care-of address as the destination IP address, encapsulates the original packet with the new IP header and forwards the packet to the mobile node. When the encapsulated packet arrives at the care-of address it is decapsulated and delivered to the mobile node. This use of encapsulation and decapsulation is often called *tunneling,* with the home agent and the care-of address as the to endpoints of the tunnel.

The mobile host can acquire the care-of address in two ways. One way is to receive the care-of address from a foreign *agent advertisement* message. In this case the care-of address is an IP address of the foreign agent, and thus called foreign agent care-of address. The other way is to receive the care-of address from an external assignment mechanism such as DHCP [17]. The care-of address is then called colocated care-of address. Note that the care-of address is the endpoint of the tunnel in both cases. The use of a foreign agent care-of address has the advantage that several mobile nodes can share the same care-of address and therefore save some of the limited address space in IP version 4. The use of a colocated care-of address has the advantage that the mobile node can in some cases act like any other node on the foreign network, e.g., in the case of short lived connections when it can use the care-of address.

Figure 2-1 illustrates the routing of datagrams with mobile IP in an example scenario (the scenario is taken from [2], Section 2.3). The mobile node is away from home and has registered with its home agent. Datagrams are routed to and from the mobile host as described below.

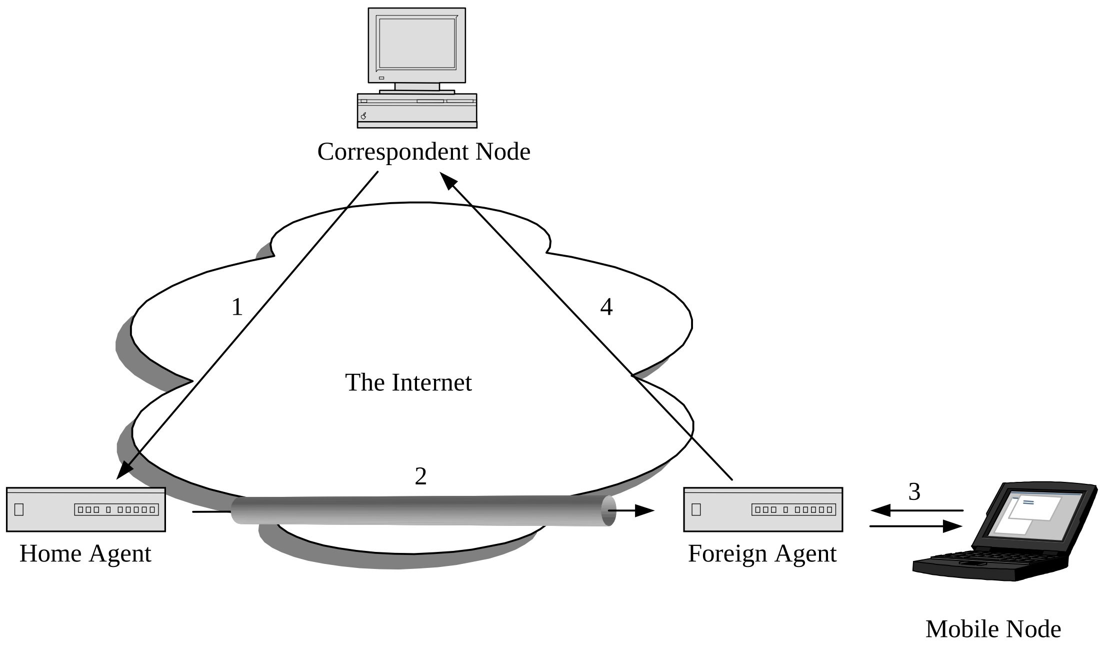

Figure 2-1: Routing of datagrams with Mobile IP to and from the mobile node

1. The correspondent node sends datagrams to the mobile node. The datagrams are routed via standard IP routing to the home network, were the home agent intercepts them.

2. The home agent encapsulates the datagrams and tunnels them to the care-of address, which in this case is a foreign agent care-of address.
3. The foreign agent decapsulates the datagrams and delivers them to the mobile node. The mobile node sends its datagrams to the default router. In Figure 2-1 the foreign agent is the mobile node's default router.
4. The foreign agent sends the datagrams from the mobile node to the correspondent host. The datagrams are routed through the Internet to the correspondent host via standard IP routing.

If a colocated care-of address is used the mobile node will act as its own foreign agent, decapsulating the datagrams by itself. The routing scenario in Figure 2-1 is often called *triangle routing*.

2.2 Tunneling with IP-in-IP Encapsulation

Mobile IP requires each home agent and foreign agent to support tunneling datagrams using *IP-in-IP encapsulation* [18]. Also mobile nodes that uses a colocated care-of address are required to support receiving datagrams tunneled using IP-in-IP encapsulation, i.e., the mobile nodes have to be able to decapsulate the datagrams. Minimal Encapsulation and Generic Record Encapsulation (GRE) are alternative methods that may optionally be supported by the mobility agents and the mobile nodes (see [2] pp 99-108 for further details).

Encapsulation of an IP datagram, using IP-in-IP encapsulation, is done by inserting an outer IP header [19] in front of the original IP header. The IP-in-IP encapsulation procedure is shown in Figure 2-2.The source and destination IP address of the outer header identify the endpoints of the tunnel, i.e., the home agent's IP address and the care-of address. The original IP header is not changed except for the *time-to-live* field (TTL), which is decremented by one.

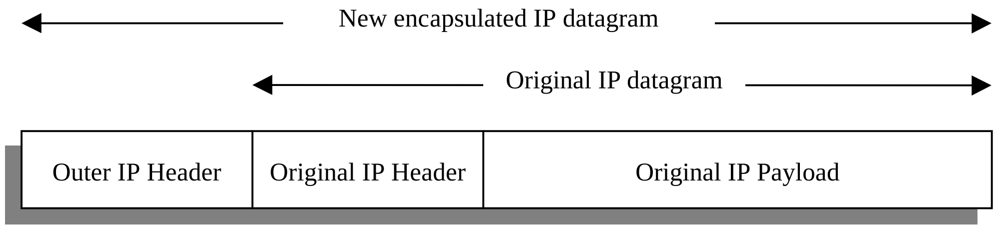

Figure 2-2: IP-in-IP encapsulation

2.3 Agent Discovery

Agent Discovery is an important process in Mobile IP. A mobile node makes use of Agent Discovery to detect if it is connected to its home network or to a foreign network. Movement between different networks is also detected by Agent Discovery, and if the mobile node is connected to a foreign network, Agent Discovery makes it possible for the mobile node to determine the foreign agent care-of address offered by each foreign agent on the network.

Agent Discovery is based upon an existing standard protocol, ICMP Router Discovery [20], and uses two additional messages: *Agent Advertisement* and *Agent Solicitation*.

2.3.1 Agent Advertisement

Agent advertisement messages are sent as periodic multicasts or broadcasts by *mobility agents*[2] on each link the mobility agents are configured for. Mobile nodes listen for these advertisements to locate themselves and register if necessary.

An agent advertisement is basically an ICMP Router Advertisement message with a *Mobility Agent Advertisement Extension*. The structure of this extension is depicted in Figure 2-3. In addition a Prefix-Length Extension and a One-Byte Padding extension can optionally be added. However, these two extra extensions are not important for the basic understanding of Mobile IPv4, and thus not discussed in this paper.

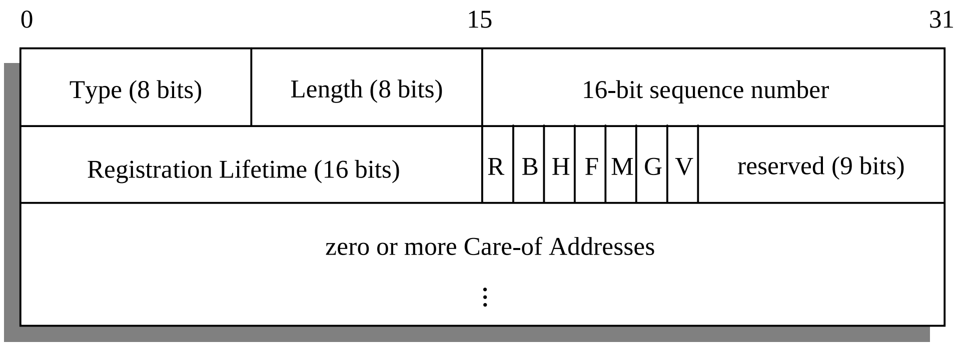

Figure 2-3: Mobility Agent Advertisement Extension

The *type* field is 16 indicating that the message is an agent advertisement. A *length* field is required since the field with the care-of address(es) can vary in size depending on the number of addresses it contains. The *registration lifetime* states the longest lifetime in seconds that the agent is willing to accept in any *registration request* (see Section 2.2.1). When all bits in this field are set, a registration lifetime of infinity is indicated. If the *R* bit is set registration with this foreign agent is required, and a colocated care-of address is not allowed to be used.

The agent advertisement can be sent by either a home agent or a foreign agent, which is indicated by the *H* bit and the *F* bit respectively. Either the *H* bit or the *F* bit has to be set in an agent advertisement message. If the mobility agent is a foreign agent, the *B* bit can be set, to inform the mobile node that it is busy and cannot accept registrations from mobile nodes at the moment. A foreign agent has to continue sending agent advertisements, even when it is busy, to inform the already registered mobile nodes that they are still in the range of the foreign agent and that the foreign agent has not crashed. A home agent must always be prepared to serve its own mobile nodes and never claim to be too busy by setting the *B* bit.

The *M* and the *G* bit indicate that the mobility agent supports minimal encapsulation and generic record encapsulation (GRE) respectively. These encapsulation methods are optional since Mobile IPv4 only requires that IP-in-IP encapsulation is supported. If the *V* bit is set the mobility agent supports use of Van Jacobson header compression [21].

If the mobility agent is a foreign agent at least one care-of address must be included in the agent advertisement message. However if the *H* bit is set, indicating a home agent the care-of address field can be empty.

[2] A mobility agent is either a home agent or a foreign agent.

2.3.2 Agent Solicitation

If the mobile node is searching for a mobility agent but does not receive any agent advertisements, it can send an agent solicitation. When a mobility agent receives an agent solicitation it should respond with sending an agent advertisement.

An agent solicitation message is identical to a ICMP Router Solicitation message, with the exception that the TTL field of the IP header must be set to one. This limits the agent solicitation messages to reach outside the originating network. The rate of which both agent advertisements and agent solicitations are sent, have to be limited to a certain maximum rate.

2.4 Registration

When the mobile node has received a care-of address, the home agent has to be notified of where the mobile node currently is. The home agent needs to know the mobile node’s care-of address to be able to tunnel datagrams to the mobile node. This is accomplished by exchanging *Registration Request* and *Registration Reply* messages. The registration procedure results in a *mobility binding* for the mobile node being created (or modified if one already existed) by the home agent. A mobility binding contains the mobile node’s home address, the corresponding care-of address and the lifetime of the registration. The home agent has a mobility binding table with an entry for each registered mobile node, to be able to map between the mobile node’s home address and its current care-of address. The mobile node also makes use of the registration procedure to renew a binding that is due to expire and to deregister when returning to its home network.

There are two variations in the registration procedure in Mobile IP, depending on if the mobile node wants to register a foreign agent care-of address or a colocated care-of address. In the case of registering a foreign agent care-of address the exchange of the registration messages is as follows: The mobile node sends a registration request to the prospective foreign agent. The foreign agent processes the registration request and then relays it to the home agent, who sends a registration reply to the foreign agent to grant or deny the request. After the foreign agent has processed the registration reply it relays the reply to the mobile node.

If the mobile node wants to register a colocated care-of address the mobile node sends the registration request directly to the home agent, who grants or denies the request by sending a registration reply. Also if the mobile node has returned to its home network and is deregistering, the mobile node naturally exchanges the registration messages directly with the home agent.

2.4.1 Registration Request

A mobile node should initiate registration by sending a registration request whenever it detects a change in its network connectivity. If the mobile node is on a foreign network, the registration request makes it possible for the home agent to create or modify a mobility binding for the mobile node. When the mobile node returns home and deregisters, the home agent deletes all mobility bindings for that mobile node and the mobile node acts as a normal node without any mobility functions. A registration request should also be sent when the mobile host needs to reregister, for example when the current registration is about to expire or if the foreign agent has rebooted.

Figure 2-4 shows the format of the registration request message. The *type* field is set to one, indicating a registration request. *S* stands for simultaneous bindings and by setting this bit the mobile node is requesting that the home agent retain its prior mobility binding. The mobile node will in this case receive one copy of every datagram for each mobility binding. If this bit is not set, the home agent will delete the previous bindings. By setting the *B* bit, the mobile node requests broadcast datagrams sent on the home network to be tunneled to it. If the *D* bit is set, it means that the mobile node will decapsulate the datagrams itself, i.e., the mobile node is using a colocated care-of address. The *M*, *G* and *V* bits are the same as in the mobility agent advertisement extension (see Section 2.3.1) and the *rsv* bits are reserved bits sent as 0.

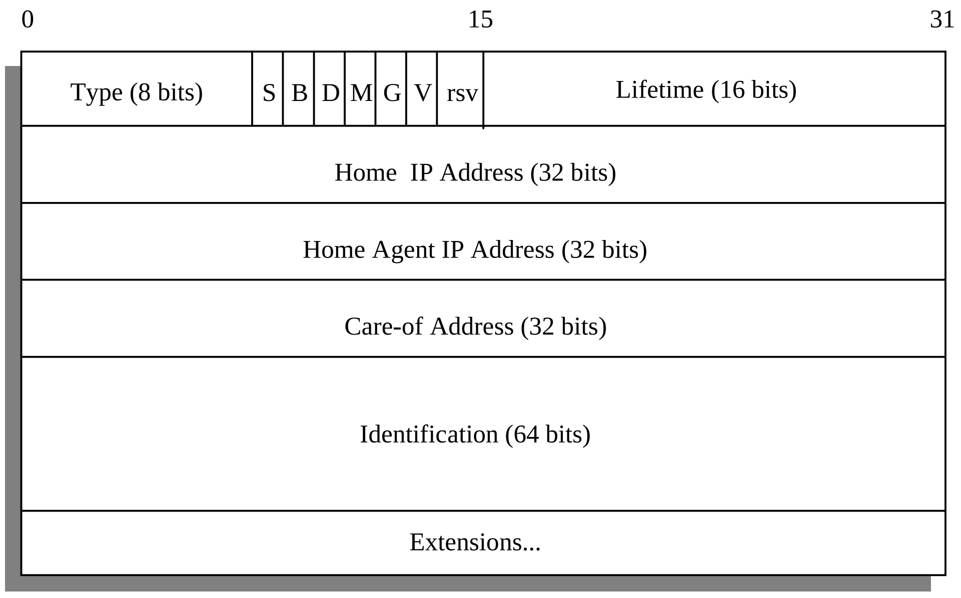

Figure 2-4: Registration Request

The *lifetime* field contains the number of seconds the registration is considered valid. The *identification* field contains a 64-bit number chosen by the mobile node to be unique for each attempted registration. The identification number allows the mobile node to match the received registration reply with the corresponding registration request. It also protects against nodes that might have saved a copy of the registration request and try to send it to the home agent at a later occasion. The *extensions* field can contain different extensions. Every registration must be authenticated to prevent potential attacks from malicious nodes, hence every registration request and reply must contain one *mobile-home authentication extension* (see [2] Section 4.9 for details).

2.4.2 Registration Reply

Before a registration reply can be sent both the foreign agent and the home agent performs certain validity checks of the registration request. The registration request is hopefully accepted, but it can also be denied by either the foreign agent or the home agent if the request for some reason fails the validity checks. In all cases a registration reply is generated and sent to the mobile node. The format of the registration reply message is shown in Figure 2-5.

The *type* field is set to 3 to indicate a registration reply. The registration reply contains a *code* field, indicating if the registration is accepted or denied. If the registration is denied the code specifies the reason for denial. The rest of the fields where explained in the previous section. A mobility agent can however decrease the lifetime if it is not willing to accept the lifetime requested by the mobile node.

If the registration request is accepted the home agent updates its mobility binding table, sends a registration reply to the mobile node via the foreign agent (provided one is used) to indicate that the registration was successful, and then starts to tunnel datagrams to the mobile node’s care-of address. The foreign agent keeps a list of visiting mobile nodes. When it receives the registration reply from the home agent it updates this list and forwards the reply to the mobile node.

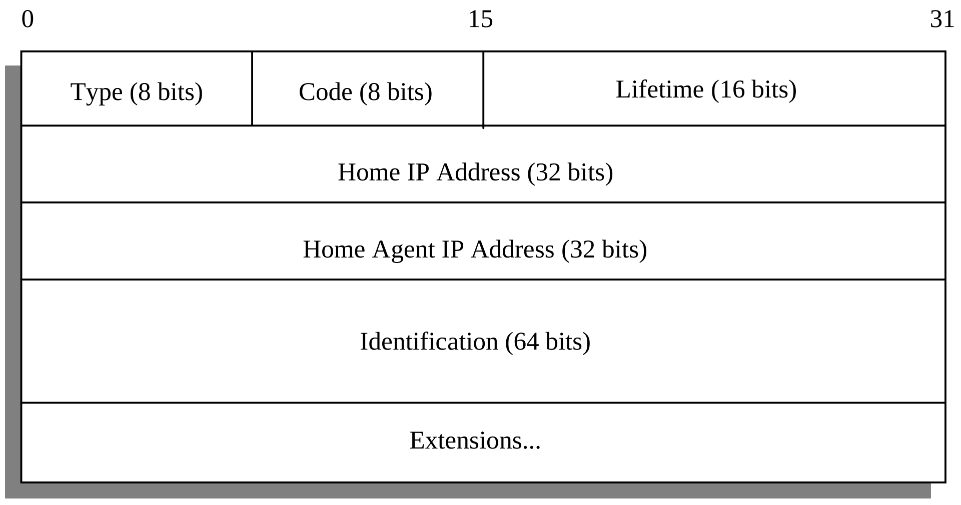

Figure 2-5: Registration Reply

2.5 Reverse Tunneling

Some routers at the border of an administrative domain, called *border routers*, perform filtering based on the source IP address. If a border router receives an IP datagram from inside of its domain with a source IP address indicating that the datagram originated from outside of the domain, the router can discard the datagram. This is called *ingress filtering*, shown in Figure 2-6a, and is used to prevent malicious Internet users from using fictitious source IP addresses in “attack datagrams”.

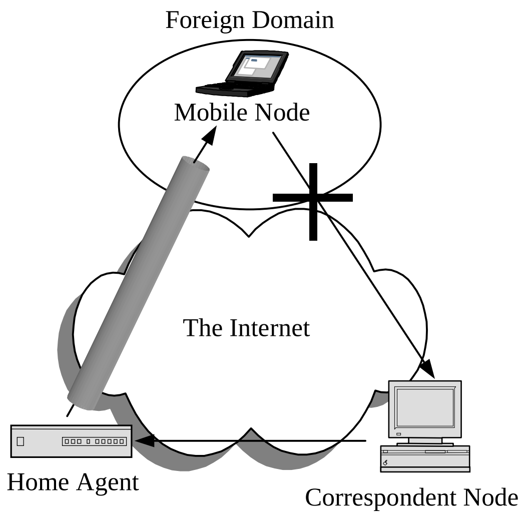

Figure 2-6a: Ingress filtering

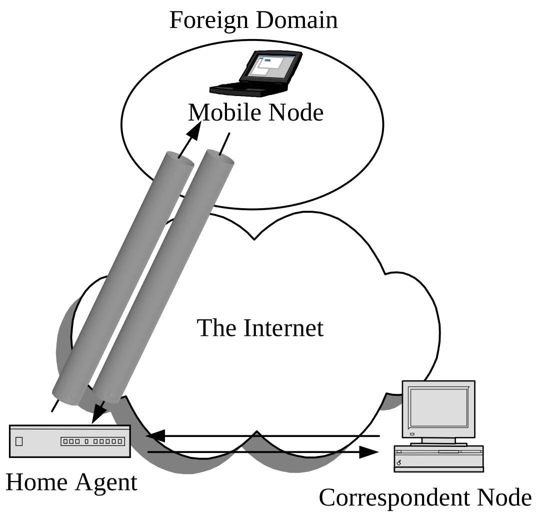

Figure 2-6b: Reverse tunneling

Ingress filtering, however makes it impossible for a mobile node to use its own IP address as the source IP address, when sending datagrams to a correspondent node from a foreign network. To solve this problem *reverse tunneling* also called *bi-directional tunneling* has been proposed. Reverse tunneling means that the datagrams sent from a mobile node to a correspondent node are encapsulated and tunneled to the home agent, who decapsulates the datagrams and forwards them to the correspondent node. This is illustrated in Figure 2-6b. The encapsulation can be performed by either the foreign agent or the mobile node. This solution is very simple, but it has an obvious drawback. Since the datagrams sent from the mobile node are tunneled to the home agent and not directly sent to the correspondent node, the routing path length for the datagrams will increase.

CHAPTER 3

TCP — Transmission Control Protocol

The Transmission control protocol (TCP) [26] is a transport layer protocol, used by many popular applications. TCP provides a reliable, connection-oriented, byte stream service, and usually runs on top of the Internet Protocol (IP). IP provides an unreliable datagram service, which means that IP datagrams can be received out of order, be duplicated or even fail to reach the receiver. IP leaves the end to end reliability issues to the upper layer protocols like TCP. TCP provides a reliable connection oriented service.

This section intends to be a short introduction to TCP and an explanation of the main features. For further information and details of TCP we recommend [4], from where most of the information below is taken.

Section 3.1 describes the TCP encapsulation. The connection establishment and termination are described in Section 3.2. Section 3.3 explains the acknowledgement scheme. Section 3.4 explains slow start. Section 3.5 describes the two ways TCP detects packet losses and Section 3.6 describes how TCP reacts to packet losses.

3.1 TCP Encapsulation

TCP receives data from an application, breaks the data into what TCP considers to be best size chunks and encapsulates the data with a TCP header. The unit of information passed by TCP to the underlying IP layer is called a *segment*. IP encapsulates the TCP segment with an IP header, resulting in an IP datagram. This is visualized in Figure 3-1.

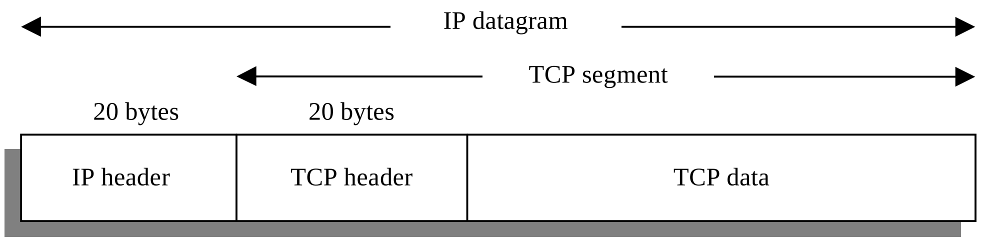

Figure 3-1: Encapsulation of TCP data in an IP datagram

TCP must be able to recover from data that is damaged, lost, duplicated, or delivered out of order. To provide this reliability, each transmitted octet of data (byte) is assigned a *sequence number*. The sequence number of the first byte in a segment is the sequence number of that segment, which is included in the TCP header, illustrated in Figure 3-2. The TCP header also contains an *acknowledgement number*, which is the sequence number of the next expected data byte. To make sure that the other end has received the data correctly, the sending TCP requires a positive acknowledgement (ACK) from the receiving TCP. When a segment containing data is sent, TCP puts a copy on a retransmission queue and starts a timer. When the ACK for the data in that segment is received, the copy is deleted from the queue. If the ACK is not received before the timer expires, the segment is retransmitted.

The sequence numbers make it possible for the receiver to order the segments that are received out of order and to discard any duplicate segments.

To be able to detect if the received segments have been damaged during the transmission over the network, TCP utilizes a checksum, which covers both its header and data. If a received segment has an invalid checksum, TCP discards it without sending any acknowledgement to the sender.

The TCP header also contains the *source port number* and the *destination port number* to identify the sending and receiving application. The two port numbers together with the source and destination IP addresses in the IP header, uniquely identify a TCP connection.

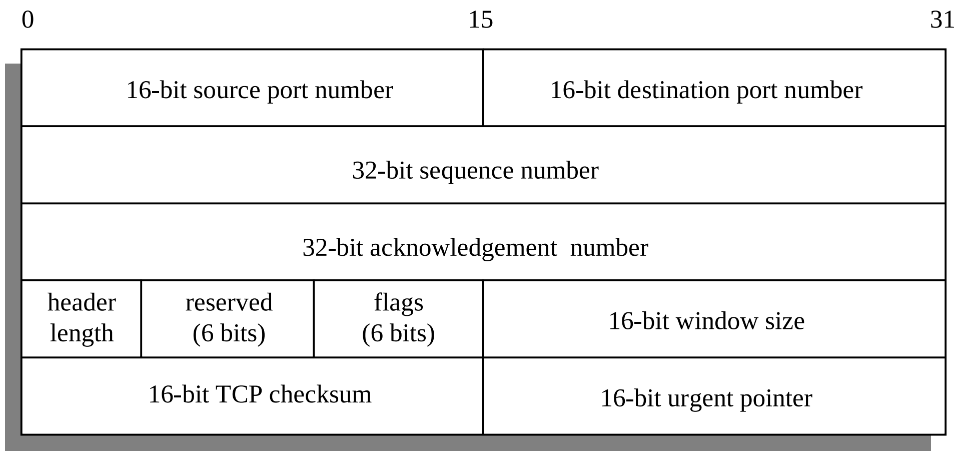

Figure 3-2: TCP header

In addition to the fields in the TCP header shown in Figure 3-2, the header can also contain some options. A *header length* field is included since the length of the header can vary depending on the options field. The normal size of a TCP header is 20 bytes, but with options the header can be up to 60 bytes.

There are six different flags in the TCP header, consuming one bit each. One or more of them can be turned on at the same time. Below follows a short description of the flags.

URG	The urgent pointer is valid. The urgent pointer is however not discussed in this paper. [4, 5] provides more detailed information.
ACK	The acknowledgment number is valid. This flag is normally always set, since there is no extra cost for including an ACK.
PSH	The receiver should pass the data in the segment as soon as possible to the application.
RST	Reset the connection.
SYN	Synchronize the sequence numbers to initiate a connection. See Section 3.2.
FIN	The sender is finished sending data. See Section 3.2

TCP also provides flow control using a sliding window mechanism. Each end of a TCP connection has a finite amount of buffer space. A receiving TCP only allows a sender TCP to send no more than the receiver has room for in its buffer. This is to prevent a fast sender from overwhelming a slow host with data. The TCP buffer size is indicated in the window size field in the TCP header. The window size informs the sender of how many bytes the receiver is willing to accept.

3.2 Connection Establishment and Termination

TCP offers a connection oriented service, which means that before two hosts can start communicating with each other, using TCP, a connection must be established between them. When the two hosts have exchanged the data they wanted to, the connection must be terminated. Figure 3-3 illustrates the connection establishment and the connection termination with a time line diagram. Time increases down the page.

The connection establishment, often called *the three-way handshake,* consists of three segments that have to be sent between the two hosts (usually referred to as the client and the server). The connection establishment is necessary to make sure that the other end is present and ready to receive data. To be certain that no packets are lost in the beginning of a data transfer, the two end hosts have to synchronize their sequence numbers. The three-way handshake is explained below.

1. To initiate a connection establishment and synchronize the sequence numbers the client sends a segment header (segment 1 in the figure) with the SYN flag set in the TCP header. The segment specifies the initial sequence number (ISN) and the port number of the server that the client wants to connect to. RFC 793 [26] recommends a change of the ISN over time, to ensure different ISN for each connection. The purpose is to prevent delayed segments from an old connection to be misinterpreted as part of an existing connection.
2. In response to the client's connection request the server sends its own SYN segment specifying its own ISN (segment 2 in the figure). In this segment, the server also ACKs the client's SYN segment.
3. To complete the connection establishment the client has to ACK the SYN segment from the server (segment 3 in the figure).

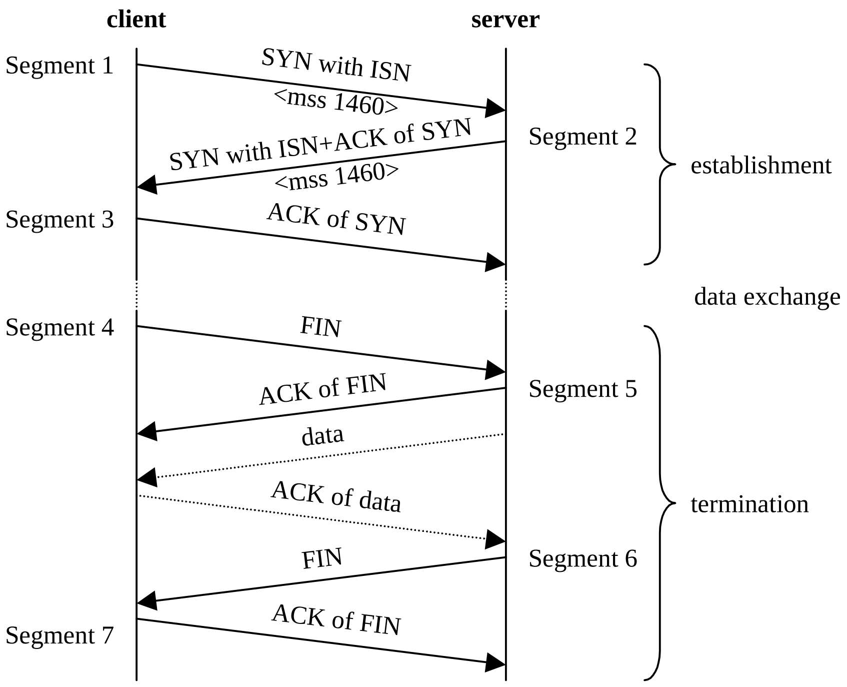

Figure 3-3: Connection establishment and termination

The additional information under the arrows representing the two first segments in Figure 3-3 shows the *maximum segment size* (MSS), which is an option that can be set in the TCP header of the SYN segments and is normally used to avoid fragmentation. An MSS value of 1460 states that the sender does not want to receive any segments larger than 1460 bytes. The MSS value must not be larger than *the maximum transmission unit*[3] (MTU) of the outgoing interface, minus the TCP and IP headers. In general it is good to have an MSS as large as possible since it will reduce the overhead relative to the data. If the destination IP address indicates a different subnet than the sender's, a default MSS value of 536 bytes is normally used.

TCP allows one end of a connection to terminate its output, but still be able to receive data from the other end. This is called a *half-close*. To be able to support the half-close, TCP requires four messages to terminate a connection.

When an application does not have any more data to send, it issues a close which causes TCP to send a segment with the FIN flag set to inform the other end that there will be no more data flowing in that direction. The receiving TCP end of the FIN segment must notify the application that the other end has initiated connection termination. When the FIN segment has been acknowledged the connection is terminated in one direction, but data can still flow in the other direction.

The exchange of the four different segments in the connection termination procedure, and the possible data transfer during the half-close, are shown in Figure 3-3 and explained below.

1. When one side has finished sending data, in this case the client, the application sends a FIN segment to the other end (segment 4 in the figure). The other end of the TCP connection, the server, notifies the application upon receiving this FIN segment.
2. The server then sends an acknowledgement of the FIN to the client (segment 5 in the figure). The TCP connection is now terminated in one direction, i.e. the data flow from the client to the server is terminated, but data can still flow from the server to the client.
3. When the server has no more data to send, it closes the application and sends a FIN segment to the client (segment 6 in the figure). The client notifies the application of this.
4. To finish the connection termination, the client sends an acknowledgement of the FIN (segment 7 in the figure). The TCP connection is now terminated.

3.3 Acknowledgement Scheme

TCP uses a cumulative acknowledgement scheme, which means that an acknowledgement specifies the sequence number of the next byte that the receiver expects, thus confirming reception of all previous bytes. If for example two segments with data bytes 1460-2919 and 2920-4379 respectively arrive at the receiver, it is sufficient for the receiver to send one ACK, confirming reception of both segments. This means that there normally is not a one-to-one correspondence between data segments and acknowledgements. However, RFC 1122 recommends an acknowledgement for at least every second segment.

An ACK can be carried by a data segment or a segment that has been created only for the purpose of acknowledging received data. Since the TCP header contains the acknowledgement number field, there is no extra cost for sending an ACK with a data segment.

[3] Most types of networks have an upper limit of the packet size (including the TCP and IP headers) in bytes the link layer can accept from the network layer. If IP datagrams are larger than the MTU, IP will perform fragmentation. This limit is called the maximum transmission unit (MTU) of the link.

In order to reduce the number of "pure" acknowledgements, TCP normally does not send an ACK the instant it receives data. Instead TCP delays the ACK, hoping to have data going in the same direction as the ACK, so the ACK can be sent along with the data. Most implementations delay an ACK up to 200 ms. However, if a segment is received out of order, TCP does not use the delayed-ACK scheme, since generating an ACK instantly enables a packet loss to be detected faster (see Section 3.5).

3.4 Slow Start

Older TCP implementations started a connection with the sender injecting multiple segments into the network, up to the window size advertised by the receiver. This works well when the two communicating hosts are on the same local area network (LAN), but if there are routers and slower links between the to hosts, packets might be lost. If some intermediate router has to queue packets, but does not have enough buffer space, packets will be dropped. This naïve approach can cause congestion and reduce the throughput significantly and ultimately lead to congestion collapse as shown in [15].

To avoid the drastic reduction of throughput TCP uses an algorithm called *slow start*. It operates by injecting new packets into the network at the same rate as it receives acknowledgements. Slow start adds a new window to the sender's TCP, so called congestion window or *cwnd*. When a new connection is established with a host, the congestion window is initialized to one segment. Each time a segment is acknowledged, TCP increases the congestion window by one segment. The sender is allowed to transmit up to the minimum of the congestion window and the window advertised by the receiver. The congestion window is flow control imposed by the sender and the advertised window is flow control imposed by the receiver.

After a connection is setup, the sender starts by transmitting one segment. When the acknowledgement of that segment arrives, the congestion window is increased to two segments, allowing the sender to transmit two segments. Receiving ACKs for both of those segments will increase the congestion window to four segments and so on. This provides an exponential increase, which continues until the rate of ACKs does not increase.

3.5 Recognition of Packet Losses

TCP recognizes packet losses by two different methods. The first method makes use of the retransmission timer. If an acknowledgment is not received before the timer expires, TCP assumes that the segment has been lost, and retransmits the segment. The second method is the reception of duplicate acknowledgments. Whenever a segment gets lost or delayed, the next arriving segment will be out of sequence. This causes receiving TCP to immediately generate and send an ACK, telling the sender which segment the receiver expects to get. Upon reception of further out-of-sequence segments, the receiver repeats the procedure, i.e., one duplicate ACK will be generated for each segment that arrives out of sequence. Since packets can be delayed and reordered in the Internet, TCP waits for a small number of duplicate ACKs (often three) before it assumes that a segment has been lost, and immediately retransmits the segment.

3.5.1 Retransmission Timeout

The second method mentioned only requires the decision of how many duplicate ACKs should be received before a packet loss is assumed. The first method (retransmission timeout) is more complicated since packets can take different routes through the internet and round-trip times can vary, making it difficult to anticipate the delay characteristics of the networks. It is however very important to have a reasonable value of the retransmission timeout (RTO). If the timer expires too quickly, unnecessary retransmissions will be triggered if the packet only is delayed, thus increasing the network load and reducing the TCP throughput (see Section 3.6). If the timer value on the other hand is too large, the sender will wait too long before it starts to retransmit packets, hence wasting network resources.

Fundamental to TCP's timeout and retransmission is the measurement of the round-trip time (RTT) experienced on a given connection. Since the RTT can change over time, TCP should notice these changes and modify its timeout accordingly. First TCP must measure the time between sending a byte with a particular sequence number and receiving an acknowledgement that covers that sequence number. Recall from Section 1.3 that normally there is not a one-to-one correspondence between data segments and ACKs. The original algorithm for calculating the RTO was based on a smoothed RTT estimator updated by TCP. Van Jacobson and Karels show in [15] that this approach can not keep up with wide fluctuations in the RTT, causing unnecessary retransmissions. They suggest a new approach, which calculates the RTO based on both the mean and the variance in the RTT measurements. Further they show that the new approach provides much better response to wide fluctuations in the round-trip times. The following equations are applied to each RTT measurement M (see [4] page 300).

$$Err = M - A$$

$$A \leftarrow A + g \bullet Err$$

$$D \leftarrow D + h(|Err| - D)$$

$$RTO = A + 4D$$

A is the smoothed RTT (an estimator of the average) and D is the smoothed mean deviation. Err is the difference between the measured value just obtained and the current RTT estimator. Both A and D are used to calculate the next retransmission timeout. The gain g is for the average and is set to 1/8. The gain h is for the deviation and is set to 1/4. The larger gain for the deviation makes the RTO go up faster when the RTT changes.

The granularity of the timer, which varies with different operating systems, should also be considered. The timer granularity for BSD is 500 ms, while it is only 10 ms for Linux. This makes the RTO value much more precise for Linux implementations of TCP compared to BSD implementations. However, a finer granularity of the timer could cause more timeouts in the case of a sudden increase to the RTT.

3.6 Reaction to Packet Losses

In the previous section the two methods that TCP uses to detect packet losses were described. TCP reacts differently to a packet loss depending on how the packet loss was detected. If the retransmission timer expires, TCP will go into slow start and *congestion avoidance* mode and use an *exponential backoff* scheme. What the exponential backoff scheme basically does is double the RTO value every time the retransmission timer expires, but only after the lost segment has been retransmitted. If TCP detects a packet loss by receiving duplicate ACKs, the *fast retransmit and fast recovery* algorithms will be used.

3.6.1 Slow Start and the Congestion Avoidance Algorithm

The assumption of the congestion avoidance algorithm is that packet loss caused by damaged data is very rare. Therefore the loss of a packet indicates congestion somewhere in the network between the sender and receiver. When congestion occurs it is desirable to slow down the transmission rate of packets into the network, and then invoke slow start to get things running again. Slow start and congestion avoidance are two independent algorithms, but they are implemented together when congestion is indicated by a timeout.

Congestion avoidance and slow start require that two variables be maintained for each connection: a congestion window, *cwnd*, and a slow start threshold size, *ssthresh*. The combined algorithm is illustrated in Figure 3-4 and operates as follows (see [4, 5]):

1. Initialization for a given connection sets cwnd to one segment and ssthresh to 65535 bytes (the maximum window size since the window size field in the TCP header is limited to a representation of 16 bits)
2. The TCP output routine never sends more than the minimum of cwnd and the receiver's advertised window (unless a Push occurs, at which time it sends what it has).
3. When congestion occurs (indicated by a timeout or the reception of duplicate ACKs), one-half of the current window size (the minimum of cwnd and the receiver's advertised window, but at least two segments) is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment (i.e., slow start).
4. When new data is acknowledged by the other end, increase cwnd, but the way it increases depends on whether TCP is performing slow start or congestion avoidance.

   If cwnd is less than or equal to ssthresh, TCP is in slow start; otherwise TCP is performing congestion avoidance. Slow start continues until TCP is half way to where it was when congestion occurred and then congestion avoidance takes over.

   Slow start begins with cwnd set to one segment, and is incremented by one segment every time an ACK is received. Hence it opens up the window exponentially: send one segment, then two, then four and so on. Congestion avoidance dictates that cwnd be incremented by 1/cwnd each time an ACK is received. This is a linear growth of cwnd, compared to slow start's exponential growth. The increase in cwnd should be at most one segment each round-trip time (regardless how many ACKs are received in that RTT), whereas slow start increments cwnd by the number of ACKs received in a round-trip time.

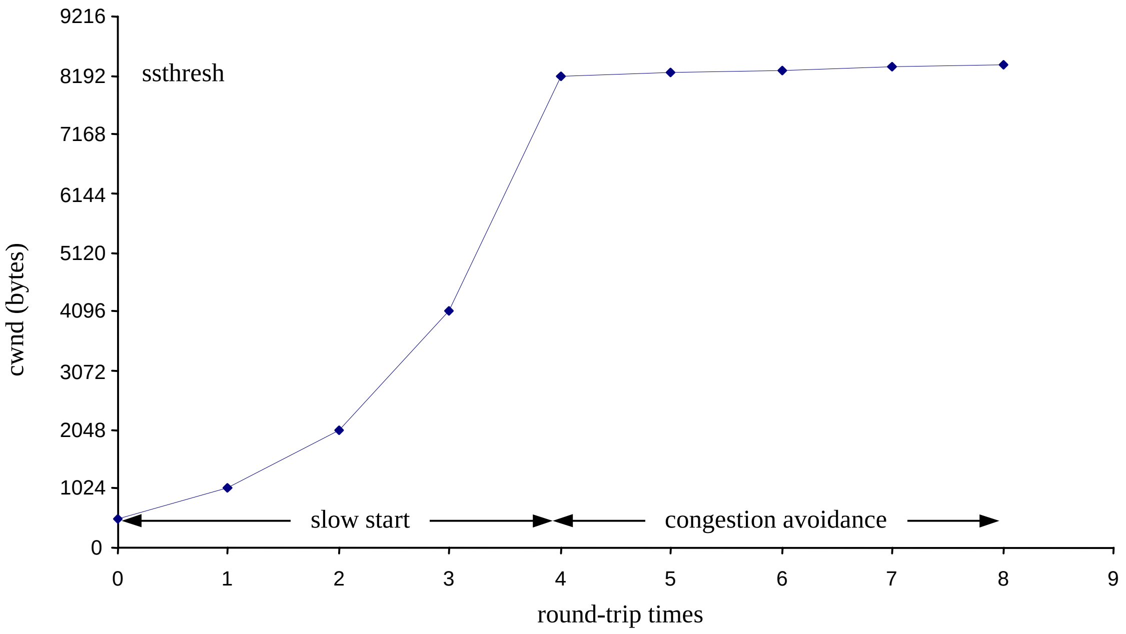

Figure 3-4: Slow start and congestion avoidance after timeout

In Figure 3-4 it is assumed that congestion occurred, detected by timeout when cwnd had a value of 16384 bytes. TCP sets the ssthresh to 8192 bytes and the cwnd to the size of one segment (512 bytes), entering slow start. cwnd increases exponentially until cwnd reaches the value of 8192 bytes and equals ssthresh, which happens after four round-trip times. After this TCP enters congestion avoidance state with a linear increase of cwnd.

3.6.2 Fast Retransmit and Fast Recovery Algorithms

The fast retransmission and the fast recovery algorithms are modifications to the congestion avoidance algorithm. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost and that there is no congestion. Then TCP retransmits the segment that seems to be missing at the receiver, without waiting for the retransmission timer to expire. This is the *fast retransmit* algorithm.

Since duplicate ACKs can only be generated upon reception of out-of-order segments, the sender knows that packets are still reaching the receiver. This indicates that a minor reduction of the flow can be enough. Thus congestion avoidance is invoked instead of slow start to avoid reducing the flow abruptly. This is the *fast recovery* algorithm. The two algorithms are usually implemented together as follows (see [4, 5]):

1. When the third duplicate ACK is received, set ssthresh to one-half of the minimum of the current congestion window and the receiver's advertised window. Retransmit the missing segment. Set cwnd to ssthresh plus three times the segment size.
2. Each time another duplicate ACK arrives, increment cwnd by the segment size and transmit a packet (if allowed by the new value of cwnd).
3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This should be the ACK of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the third duplicate ACK. This step is congestion avoidance, since TCP is down to one-half the rate it was when the packet was lost.

CHAPTER 4

Reasons for TCP Problems in a Wireless Mobile Environment

The Transmission Control Protocol (TCP) was developed for traditional networks with wired links and stationary hosts. Packets on the Internet get lost either because of network congestion or corruption by the underlying physical link. Wired links are often characterized by low bit-error rates, and are thus considered quite reliable. TCP assumes that packet losses are due to congestion, which in most cases is correct in these kinds of networks. Losses are detected either by timeouts or multiple duplicate acknowledgements (the fast retransmit algorithm described in Section 3.6.2). By invoking congestion control and slow start (Section 3.6.1) in the case of packet losses or large delays, TCP performs very well in wired networks.

4.1 Unreliable Characteristics of Wireless Links

Communication over wireless links is characterized by high bit-error rates due to channel fading, noise, or interference, and temporary disconnections due to handovers, which is something that has to be considered by network protocols and applications. However, TCP reacts no differently to packet losses in a wireless environment even though they may not be due to congestion. Since packet losses in a wireless network often occur because of the unreliability of the link and not because of congestion, there is an unnecessary reduction of the link utilization, when TCP makes the transmission window size smaller, initiates congestion control and slow start, and resets its retransmission timer. This of course results in significant performance degradation in the form of poor throughput and increased delay. The negative effects of unnecessary slow starts are extensive in Long Thin Networks [6], i.e., networks containing links with high round-trip times (RTT) and low bandwidth. If the RTT is high, it will take a long time for the link to recover, i.e. to be fully used after an unnecessary slow start.

There are several different approaches to solve the problems with TCP performance in wireless networks. One way would be to modify TCP to make it mobile aware, but because of the sheer number of hosts using TCP, it would be unfeasible to modify all hosts. Another possibility would be to build a separate transport layer protocol for mobile hosts. However, this solution causes interoperability problems with the stationary hosts using TCP.

The optimal solution would hence be to improve TCP performance without requiring any modifications to the stationary hosts or causing any interoperability problems between the mobile and stationary hosts. We summarize some different approaches below and try to point out some advantages and disadvantages of each method.

4.1.1 Split Connection Approach

As the name implies the split connection approach tries to alleviate the problems associated with TCP performance over wireless links by dividing a TCP connection into two separate connections; one connection over the wired link and one connection over the wireless link. A regular TCP connection is set up between the stationary host and the base station. For the connection between the base station and the mobile host some mobile aware transport protocol is used, e.g., a modified TCP such as the Indirect TCP (I-TCP) protocol [9]. Another split-connection approach is the Wireless Socket Protocol (WSP), developed at EED/R[4].

[4] Research Department, Ericsson Eurolab Deutschland GmbH

The advantage of the split connection approach is that it separates the flow and congestion control of the wireless link from that of the fixed network. If a well suited transport protocol is used over the wireless link, the performance could improve significantly. A. Bakre and B. R. Badrinath show in [10] that I-TCP performs up to four times better than regular TCP regarding the end to end throughput.

However, there are some weaknesses in this approach. Since the TCP connection is split in two distinct connections, segments are not acknowledged end-to-end. An acknowledgement of a TCP packet could in fact reach the sender before the packet reaches the receiver.

4.1.2 Snoop

In [11] the authors describe a method to improve end-to-end throughput in a TCP connection between a fixed host and a mobile host. The authors address both the problem with the high bit-error rate on the wireless link and the handover problem. Protocol changes are made to the network layer software at the base stations[5] and the mobile hosts. Measurement results of throughput speedups of up to 20 times compared to regular TCP are presented in [11].

The base station routing code is modified by adding a module, called *snoop,* that monitors every packet that passes through the connection in either direction. The snoop module keeps track of all the packets sent by the fixed host and all the acknowledgments from the mobile host. When a packet loss on the wireless link is detected by the base station (duplicate acknowledgements or timeout), the base station retransmits the lost packet to the mobile host. This local retransmission is possible since the base station has the packet cached. The packet loss is hence invisible to the fixed host and unnecessary slow starts can be avoided.

Packets sent by the mobile host can of course also be dropped on the wireless link. There is no way for the mobile sender to know if the packet loss occurred on the wireless link or elsewhere in the network due to congestion. Since TCP retransmission timeout is based on round trip time estimates for the connection, sender timeouts for lost packets on the wireless link will happen much later than they should. To avoid this, the snoop module generates negative acknowledgments for packets that have been lost on the wireless link.

4.1.3 Explicit Feedback

As shown in the Snoop solution, local retransmissions from the base station to the mobile host improve the performance of TCP. However, timeouts can still occur at the source while the base station is performing local retransmissions. Bakshi, et al. propose a solution in [12] using local retransmissions combined with an explicit feedback mechanism to eliminate timeouts at the source during local retransmissions. The results in [12] show an improvement of up to 100% of the TCP throughput compared to basic TCP.

To simulate the characteristics of the wireless link with some deep fades, a two-state Markov model was used. The simulations started with the wireless link in the good state. After ten seconds the wireless link entered the bad state, remaining there for four seconds, and then reentered the good state. This cycle continued for the duration of the connection. During the good state no packets were lost and during the bad state all packets were lost.

Since all packets are lost during the bad state, the base station cannot get the buffered packets through to the mobile host. To prevent the TCP sender from invoking congestion control

[5] A wireless network is connected to the wired Internet via a router, which must have at least one interface to support the wireless communication and one interface to support the wired communication. Such a router is often referred to as a Mobile Support Router (MSR) or a Base Station (BS). In this survey we will call such routers Base Stations.

mechanism during the bad state, the base station sends an Explicit Bad State Notification (EBSN) to the sender. The EBSN message causes the sender to reset its retransmission timer. The base station keeps sending EBSN messages to the sender, while trying to retransmit unacknowledged packets to the mobile host, until the first acknowledgement sent by the mobile host after the bad state is detected. The scenario is illustrated in Figure 4-1.

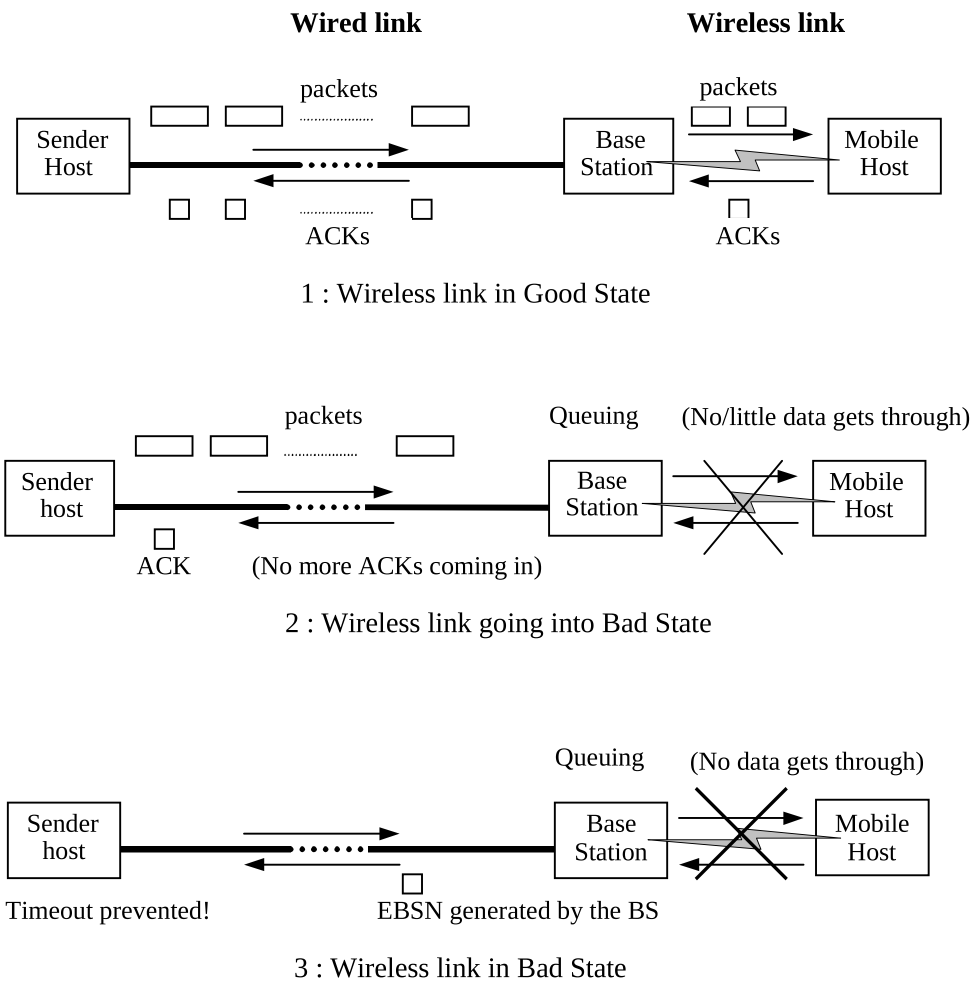

Figure 4-1: Illustration of how the explicit feedback works with EBSN when a link enters bad state. Timeout of the retransmission timer is prevented by the EBSN sent by the BS.

Bakshi, et al. show that their solution improves the throughput of TCP traffic compared to basic TCP. However, to be able to prevent timeouts during local retransmissions the TCP sender needs to understand the EBSN messages, which requires modification to the TCP code at the sender. To make it possible for the mobile host to communicate with any host on the Internet this approach would hence require modification to TCP at all existing hosts. As mentioned earlier this is not feasible.

4.2 Handover Delay

Packet losses can occur due to the handover delay, causing TCP to invoke its congestion control mechanisms. As mentioned before, packet losses result in performance degradation. Hence it is desirable to minimize the handover latency.

When a mobile node is moving within a subnet, it can keep its care-of address. Thus the handover in this case does not have to involve registration with the home agent. Instead the handover should be performed locally between the base stations as suggested in [22]. For handover between different subnets, but within the same administrative domain the authors of [22] suggest a hierarchical foreign agent strategy. The mobile node is registered at the home agent with a *domain foreign agent* care-of address, which remains the same as long as the mobile node stays within that domain. The subnet foreign agents include the domain foreign agent care-of address in their agent advertisement messages to make it known to the mobile nodes. The domain foreign agent maintains per-mobile node routing entries, which it updates when a mobile node moves across subnets.

In [11] a solution to achieve low-latency handovers without any data loss by using multicast and intelligent buffering in the nearby base stations, is presented. The home agent multicasts the packets destined to the mobile host to the base stations that is close to the mobile host. When the mobile host changes cell, the new base station has the most recent packets cached and also receives all new packets by multicast, the handover can be fast and no *in flight* packets[6] will get lost.

The disadvantage of this scheme is the increase of the traffic load in the backbone and the increase of the processing load in the routers. Another possible problem could be that retransmission timeouts still occur at the TCP sender during local retransmissions, causing unnecessary invocation of the congestion avoidance algorithm.

[23] investigates the effects of using a colocated care-of address on *macro handover*[7] latency with Mobile IP. The focus is on the retrieval of a colocated care-of address using the Dynamic Host Configuration Protocol (DHCP) [17]. The authors suggest that the address conflict checking mechanism recommended for DHCP should be left out of the care-of address retrieval process and performed as a background process instead. They show that by doing so, the address retrieval delay is reduced by 96 %.

4.3 Rapid Changes to the Link Characteristics

Rapid changes of the link characteristics in the case of roaming across different access networks, will probably also cause performance degradation. The sudden changes could cause spurious

[6] Packets sent to the old care-of address, that do not arrive before the mobile node has changed its care-of address are called *in flight* packets.

[7] Macro handover is a commonly used term for the handover necessary when a mobile node switches from one subnetwork to another. When the mobile node roams between two cells within the same subnetwork the handover required is called *micro handover*.

timeouts in the case of handover from the faster link to the slower link due to differences of the round-trip times. In the case of switching from a slower link to a faster link it could a take long time for TCP to adapt to the changes, leaving available bandwidth on the faster link unutilized for an unnecessarily long time. However, not much is known about the TCP reaction to sudden changes of the link characteristics. As mentioned in the introduction the goal of this thesis is to find out the possible TCP problems.

CHAPTER 5

MosquitoNet Mobile IPv4 Implementation

The Mobile IP implementation we have chosen to use is the MosquitoNet Mobile IPv4 implementation [7], which is an implementation under Linux developed at the Computer Science Department of Stanford University. It was chosen because the required software with installation information and other relevant documentation could easily be retrieved from the Internet. Also, a reduction of hardware is possible since this implementation does not provide any foreign agents. Further, some people at Ericsson had previous experience with it.

5.1 Design Overview

The MosquitoNet Mobile IP is an implementation based on the IETF Mobile IP proposed standard (RFC 2002) and developed for Linux kernel 2.0.33 or 2.0.36. The MosquitoNet Mobile IP implementation provides all the required capabilities of both the mobile host and the home agent. The encapsulation used is IP-in-IP encapsulation which is a Mobile IP requirement, but minimal encapsulation and GRE are not supported. A foreign agent is not provided by the implementation. However, the MoquitoNet mobile node can interoperate with a foreign agent if desirable, when there is one available at the foreign network. Since a foreign agent is not provided, this implementation focus on having a mobile host with a colocated care-of address, and tries to take advantage of the extra flexibility made possible by this running mode.

The main advantage of excluding foreign agents is of course that the foreign networks do not need to provide a foreign agent. If we would like to include more foreign networks in our measurement environment, see Chapter 6, we do not need to set up a foreign agent for each new foreign network. Another advantage is that standard IP routing can be used for short lived connections where mobile support might not be needed. The mobile mode, using its colocated care-of adderss can act as any other node on the subnet.

One disadvantage of leaving out the foreign agents is the additional packet losses that can occur during handovers. Packets sent by the home agent to the mobile host, when the mobile host is changing network, will be lost in the case of having a colocated care-of address. If instead foreign agents are being used the old foreign agent can forward these packets to the new foreign agent to avoid packet losses. This disadvantage is mitigated in this implementation by allowing multiple active interfaces simultaneously. Before the handover is completed, i.e., before the mobile node has registered successfully with its home agent the mobile node can still receive packets over the interface associated with the old care-of address and hence no packets are lost. Such a handover when no packets are lost is often referred to as a *soft handover* and requires that the cells[8] overlap sufficiently with respect to the handover latency and the mobile node's velocity [23].

In the MosquitoNet Mobile IP implementation some core functions in Mobile IP such as packet encapsulation and forwarding are implemented in the kernel. Other functions such as the transmission and reception of registration messages are implemented in user space daemons.

[8] The limited area surrounding a base station, within which mobile nodes can establish connectivity with the base station is called a cell.

5.2 Supporting Multiple Delivery Methods

The MosquitoNet Mobile IP supports different delivery methods for different flows of traffic, resulting in significantly improved throughput for some traffic flows. This is an enhancement of the IETF Mobile IP RFC 2002. The three different delivery methods are:

- Regular IP — Routing is performed as usual, i.e., without mobile IP support.
- Mobile IP with triangle routing — Packets sent from the correspondent host (the host which is communicating with the mobile host) to the mobile host, are routed via the home agent which tunnels the packets to the mobile host's care-of address. Packets from the mobile host are routed with regular IP.
- Mobile IP with reverse tunneling — Packets sent to and from the mobile host are sent via the home agent. The tunnel with the home agent and the mobile host's care-of address as endpoints is used in both directions.

Some Internet traffic has no need for Mobile IP features. For instance, when downloading a Web page, it usually makes no difference what IP address you have. The mobile host will get the correct Web page regardless of whether it is using its care-of address or its home address, provided that the mobile host does not move to another foreign network during this download. In this case, the normal IP routing mechanism could be used to avoid unnecessary Mobile IP overhead, improving the performance significantly if the mobile host is far from its home network but near the correspondent host.

Some domains require tunneling of the datagrams not only from the home agent to the mobile host but also from the mobile host to the correspondent host, provided that the mobile node is located within such a domain and the correspondent host outside the domain. This tunneling in both directions is called reverse or bi-directional tunneling and is required when border routers do filtering based on packet source address, so called ingress filtering (Section 2.5).

Some flows of traffic can use triangle routing. The round trip times could be reduced significantly by using triangle routing compared to reverse tunneling, when the correspondent host and the mobile host are "close" to each other and "far away" from the home agent.

5.3 How it Works[9]

To be able to support different delivery methods for different flows of traffic based on the characteristics of the traffic, the MosquitoNet Mobile IP implementation uses a Mobile Policy Table (MPT). The Mobile Policy Table determines which policy to use based on the destination IP address and the destination port number. The IP address is useful to be able to treat flows to different destinations differently. The port number is used as a hint of the application, since the port number in many cases indicates the nature of the traffic, such as port 23 for telnet or port 80 for HTTP traffic.

Figure 5.1 (Figure 4 in [27]) illustrates where the Mobile Policy Table together with the routing table fit in the protocol stack within the Linux kernel. To determine how to send the packets, the routing lookup function is modified to consult both the original routing table and the MPT.

5.3.1 The Routing Lookup

The source IP address in use for a particular packet is passed as an extra argument to the modified route lookup function, which uses the source IP address to determine if the packet is subject to

[9] The information in this section is taken from [27].

policy decisions in the Mobile Policy Table. No mobility decision should be made for a packet which already has the source IP address set to an IP address associated with one of the physical network interfaces. There are two possible cases when the source IP address is set for a packet. One case is if the packet has been looped back by the *virtual interface*, *vif* (see Section 5.3.2). The other case is when the source IP address has been set by a certain application. The mobile host daemon is one example of such an application. When the mobile host daemon sends registration messages to the home agent, it needs to send them over a real interface. The IP address associated with the particular real interface is the address that the mobile node wishes to register as a colocated care-of address with the home agent.

For packets that do not have the source IP address set, the route lookup function must consult the mobile policy table to decide the appropriate delivery method.

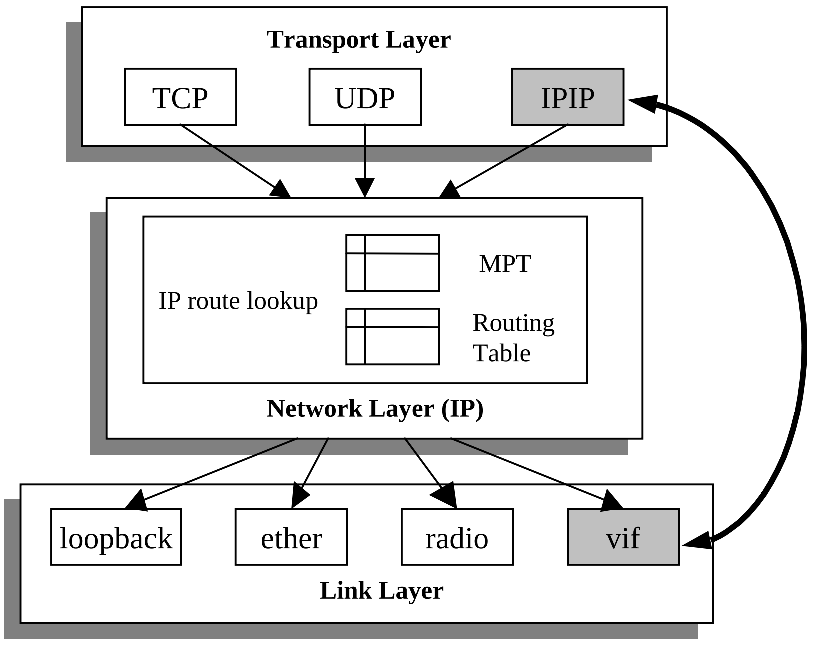

Figure 5-1: The MPT is placed in the network layer and is consulted together with the routing table.

5.3.2 The Virtual Interface (vif)

Packets that need to be encapsulated are handled by the *virtual interface* (*vif*). At the mobile node, packets are only sent to the *vif* if reverse tunneling is required. Packets sent to the *vif* are encapsulated and then looped back to the IP layer, for delivery to the home agent. In this case the packets have already gone through the route lookup function once when the decision of reverse tunneling was taken and the source IP address set. So this time when the packets are passed to the route lookup function, they will be sent through one of the physical interfaces.

Packets that have been tunneled by the home agent and arrive at the mobile node are looped to the *vif* by the IPIP module after decapsulation. The *vif* is configured with the mobile node's home IP address. Hence the packets appear to have arrived from an interface connected to the mobile node's home network.

5.4 Other Mobile IP Implementations

There are several Mobile IP implementations available. This section lists a few of them. All of the different Mobile IP implementations mentioned below have divided the software into two parts – kernel-level code and user-level code. To maximize performance, functions that are performed on every packet, such as encapsulation, decapsulation, and forwarding are implemented in the kernel. Functions such as handling registration messages and agent advertisements are implemented in user-space.

5.4.1 Solaris Mobile IP

The Solaris Mobile IP implementation [13] was developed at SUN Microsystems Inc., Palo Alto. The current release[10] is an experimental prototype which requires a SPARC or x86 platform with at least 16 MB of RAM running Solaris 2.5.1 or later. The implementation supports both the foreign agent solution and the colocated care-of address solution.

5.4.2 CMU Monarc Implementation

This implementation [14] is a part of the CMU Monarc project at the Computer Science Department at Carnegie Mellon University. Version 1.1.0[11] is fully compliant with RFC 2002, but does not support reverse tunneling or route optimization. It has been tested with the NetBSD platform, releases 1.1 and 1.0A, and the FreeBSD platform, releases 2.2.2 and 2.2 GAMMA.

5.4.3 Binghamton

A Mobile IP implementation for Linux was developed at the State University of New York, Binghamton. Version 1.00[12] of Linux Mobile IP complies with revision 16 of the IETF Mobile IP draft and can be implemented using foreign agents or colocated care-of addresses. The software requires that IPIP tunneling is supported in the kernel, which is the case for Linux kernel versions after version 1.3.

5.4.4 NUS Mobile IP Implementation

The latest version of National University of Singapore Mobile IP implementation, version 3.0beta[13] was released in the end of February, 1999 and works under the Linux 2.0.34 kernel.

[10] The current release of SUN's Mobile IP is available at http://playground.sun.com/pub/mobile-ip/

[11] Release 1.1.0 of the CMU Monarc Mobile IPv4 is available at http://www.monarch.cs.cmu.edu.

[12] Version 1.00 of Linux Mobile IP is available at http://anchor.cs.binghamton.edu/~mobileip

[13] Version 3.0beta of the NUS Mobile IP implementation is available at http://mip.ee.nus.edu.sg.

CHAPTER 6

Measurement Environment

The goal is to be able to switch seamlessly between different networks and make use of whatever connectivity available. As mentioned in [16], one example when we may need to switch from an Ethernet or a WaveLAN connection to a radio modem is when we leave our offices, taking our computers with us. If we arrive at a location with a higher speed connection available, we would of course like to switch once again to take advantage of the faster connection. The user will of course notice some obvious changes in throughput if the different networks have widely different characteristics.

In our measurements we have used a Lucent Technologies' WaveLAN connection and a PPP connection over GSM. The mobile host performs *soft* handover with no packet losses. Soft handover is desirable in out setup since this thesis focuses on TCP reaction to the rapid changes of the link characteristics. In the MosquitoNet Mobile IP implementation (see Section 5) this is achieved by having multiple interfaces active simultaneously. A handover is initiated, when the mobile host sends a registration request to its home agent, with the new co-located COA using the "new" interface. From this point in time the mobile host will use the new interface for all outgoing packets. Incoming packets will still be received over the "old" interface until the new co-located address is registered with the home agent. This handover procedure will create completely different results depending on if the file transfer is made from or to the mobile host. To avoid this we modified the Mobile IP implementation, making the mobile host not to use the "new" interface until the registration is completed.

For our measurements to observe the TCP behavior we had to build a measurement environment, which we have tried to make as realistic as possible with the hardware available. The measurement setup is illustrated in Figure 6-1 and described in Section 6.1. In Section 6.2 the link characteristics of the two different wireless links are described. Section 6.3 discusses the limitations of our measurement setup. Finally, the different scenarios that we found interesting are described in Section 6.4.

6.1 Description of the Setup

The mobile IP implementation used in our measurement environment is the MosquitoNet Mobile IP implementation release 1.0.5, which requires Linux kernel 2.0.33 or 2.0.36. We have chosen to use the Linux kernel version 2.0.36. The MosquitoNet implementation does not provide any foreign agents. Instead the mobile hosts are supposed to use co-located care-of addresses whenever they are attached to a foreign network.

Figure 6-1 illustrates the measurement setup. The home network, 192.168.194.128 with the netmask 255.255.255.128, is an Ethernet. The home agent serving this network is located on the computer Saturn, which has three different interfaces. The Home Agent's IP address, i.e., the IP address associated with its Ethernet interface, is 192.168.194.160. In addition to the Ethernet interface, Saturn has two other interfaces which will appear to be the foreign networks: one PPP interface using the IP address 10.0.0.1 and a WaveLAN interface with the IP address 194.231.118.195.

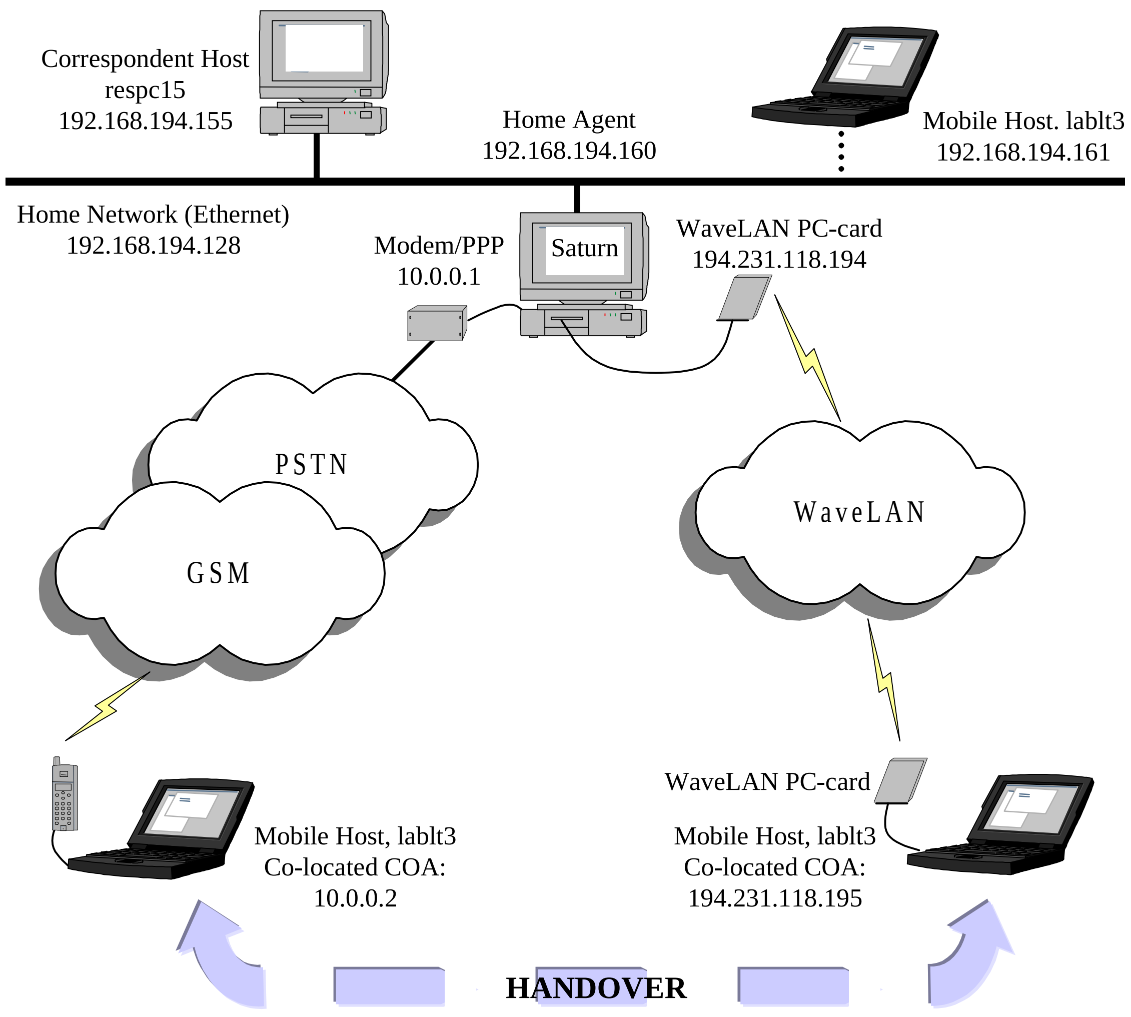

Figure 6-1: The measurement setup

The mobile host is a laptop, lablt3, which can be connected to an Ethernet, a WaveLAN, or a PPP link. When the mobile host is at its home network connected via a Fast Ethernet 16-Bit PC card, it uses its home address 192.168.194.161. When away from home the mobile host is connected to Saturn either via a Point-to-Point GSM link (9.6 kbps) using a GSM modem card or via the WaveLAN (1-2 Mbps) using a PCMCIA WaveLAN card. Non transparent and asynchronous mode is used over the GSM link. Non transparent mode means that the Radio Link Protocol (RLP) is used and the link is considered reliable. In asynchronous mode, a start bit and a stop bit are added to every byte. The mobile host has been assigned static care-of addresses. The mobile host's co-located care-of address over the PPP link is 10.0.0.2, which is assigned by the PPP server (Saturn). When the mobile host is connected to the "WaveLAN foreign network", it will have the co-located care-of address 194.231.118.195. The correspondent host, respc15, is connected to the home network and has the IP address 192.168.194.155.

The measurements will focus on the TCP performance when the mobile host is roaming from the WaveLAN wireless LAN to the GSM link and vice versa. What we mainly are interested in, is the TCP behavior when there are rapid changes in the link characteristics.

6.2 Link Characteristics

The two parameters we consider are the bandwidth and the round-trip time.

Bandwidth: The WaveLAN has a bandwidth of 1-2 Mbits/sec. However the bandwidth of the PPP link is much lower. The specified bandwidth for GSM is 9.6 kbits/sec. The link between the home agent and the correspondent host is an Ethernet with a bandwidth of 10 Mbits/sec. Regardless of whether the mobile host is connected to the WaveLAN or the PPP/GSM link it is obvious that the wireless link will be the bottleneck.

Round-trip time (RTT): The round-trip time was measured from the mobile host to the correspondent host (Respc15). When connected via the WaveLAN, an RTT of ~3.5ms was measured, using the ping program. For the PPP link using GSM, an RTT of ~700ms was measured.

The WaveLAN link is thus quite fat and short, i.e., it has a high bandwidth and a low round-trip time, while the PPP link is thin and long, i.e., it has a low bandwidth and a high round-trip time.

The capacity of a link, normally called the bandwidth-delay product is the number of bits that the link can be filled with. The capacity of a link is calculated as follows:

$$\textit{Capacity [bits]} = \textit{bandwidth [bits/sec]} \times \textit{round-trip time [sec]}$$

The theoretical capacity of the WaveLAN link, provided that the maximum bandwidth is assumed (2 Mbits/sec), is about 7000 bits or 875 bytes. The corresponding value for the PPP link is 6720 bits or 840 bytes, i.e., the difference in capacity for the two links is not very big. The Ethernet has a round-trip time of only 0.5 ms and a bandwidth of 10 Mbits/sec resulting in a capacity of 5000 bits or 625 bytes.

The characteristics of the different links in the two cases are illustrated in Figure 6-2 and Figure 6-3, but note that the scales are not accurate. The cylinders represent the different pipes. The circle area of the cylinders corresponds to the bandwidth and the length of the cylinders corresponds to the round-trip times. Figure 6-2 shows the case when the mobile host is connected to the WaveLAN and Figure 6-3 shows the case when PPP/GSM is the wireless link.

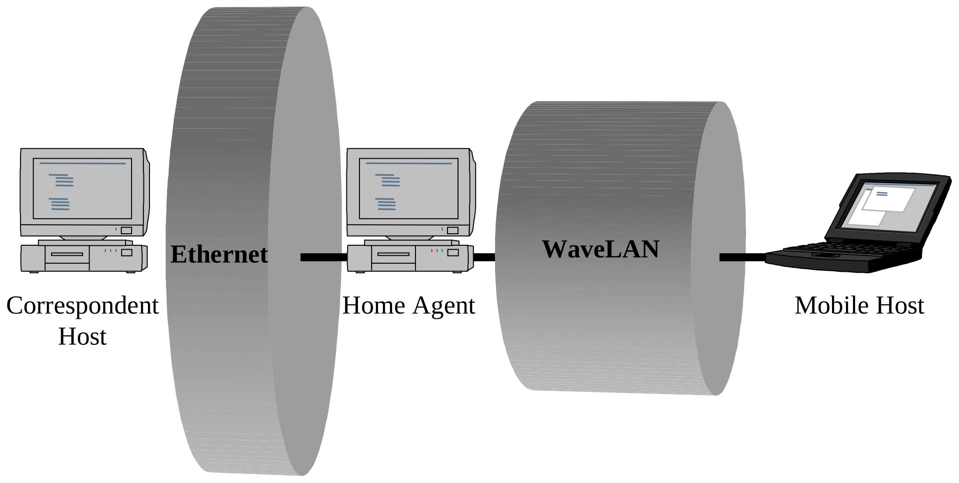

Figure 6-2: Illustration of the different link characteristics of the connection over WaveLAN

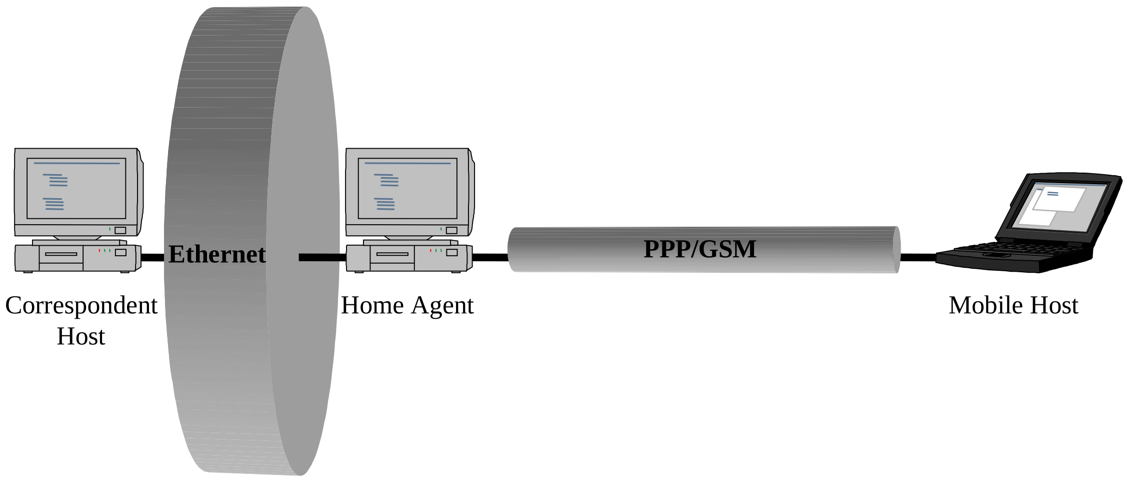

Figure 6-3: Illustration of the different link characteristics of the connection over PPP/GSM

6.3 Limitations of the Setup

Our measurement setup is very simple. We have limited the different access networks to a WaveLAN connection and a PPP connection over GSM. The focus however is not on how TCP reacts to handover between these two specific networks, but rather how TCP is affected by the drastic changes of the link characteristics. Other access networks could as well be used for this purpose.

The mobile host is only one hop away from the home network. The home agent is always connected to the same network as the mobile host, regardless of if the mobile host is at home or at one of the two foreign networks. When the mobile host is connected to the PPP link, it would be more realistic if the PPP link was between the mobile host and an Internet Service Provider (ISP) and if the correspondent host was located somewhere on the Internet. Also when the mobile host is connected to the WaveLAN it would be more realistic to be several hops away from the home network and the correspondent host. However, using the Internet as a testbed would make the measurements hard to reproduce, since the conditions of the Internet change constantly.

Soft handover could be difficult to achieve in reality. The two cells involved in the handover have to be overlapping. To avoid data loss the cells overlap should be large enough with respect to the handover delay and the velocity of the mobile host [23]. In addition it is difficult for the mobile host to detect when it is about to lose connectivity with its current access point. This could however be achieved by measuring the signal-to-noise ratio (SNR), and initiate handover when the SNR drops below some predefined threshold.

In our measurements the mobile host does not get the colocated care-of address via a DHCP [17] server or a BOOTP [25] server, which maybe would be the most natural way. The colocated care-of address for the WaveLAN interface is configured statically on the mobile host and the colocated care-of address for the PPP link is received from the PPP server (Saturn in our measurement setup). The use of a DHCP or a BOOTP server would increase the handover delay (see [23]) and possibly have a negative impact on the TCP performance. However, this would not have any impact on our measurements since the mobile IP implementation supports soft handover.

For the purpose of studying the impact of sudden changes to the link characteristics on TCP performance, we consider this setup sufficient.

6.4 Measurement Scenarios

We have tried to realize two important and interesting scenarios, which are listed below.

- *Scenario 1: “short fat pipe → long thin pipe”*

  As we noticed in Section 6.2, the capacities of the two links do not differ very much. It is however interesting how well TCP can handle the change from a short fat pipe to a long thin pipe, even if the capacity stays roughly the same. The significant difference in the round-trip times of the two links will probably cause problems for TCP. This scenario can be studied by initially having the mobile host connected to the WaveLAN, and then perform a handover to the PPP/GSM link.

- *Scenario 2: “long thin pipe → short fat pipe”*

  The only difference in this scenario from scenario 1 is that the mobile host starts by being connected to the PPP/GSM link, and then switches to the WaveLAN.

6.4.1 Other Possible Scenarios

Two additional scenarios are described below. They are however beyond the scope of this thesis and no measurements will be made according to them.

- *Scenario 3: high capacity → low capacity”*

  The fact that the capacities of the two links are nearly the same, forces us to make some changes to create a distinct difference between the two link capacities. One possible modification to the setup would be to make the path between the correspondent host and the home agent longer, i.e., increase the number of hops. If the round-trip time between the correspondent host and the home agent could be increased to 100 ms (which is not unrealistic over the Internet), the RTT between the mobile host and the correspondent host would be about 100 ms when the WaveLAN is used and about 800 ms when PPP/GSM is used. This would result in widely different capacities for the two different access networks. The connection including the WaveLAN would be the high capacity link and the connection via PPP/GSM would be the low capacity link. Hence the handover in this scenario would be from WaveLAN to the PPP/GSM link.

- *Scenario 4: “low capacity → high capacity”*

  This scenario is the same as scenario 3 except that the mobile host initially is connected to the PPP/GSM link, and then does a handover to the WaveLAN.

CHAPTER 7

Results

The measurement results from the scenarios described in Section 6.4 are presented in this chapter. A program called Netperf[14] is used for the bulk data transfer between the mobile host and the correspondent host. The mobile host will use its home IP address 192.168.194.161 as the source IP address regardless of connection to the home network or a foreign network (triangle routing). If nothing else is mentioned, the advertised window size by the receiver is 16 kbytes, and the maximum transfer unit (MTU) is 1500 bytes for all links (Ethernet, WaveLAN and PPP). The advertised window size is set by default by Netperf to 16 kbytes. An MTU of 1500 bytes might be considered high for a PPP link. The reason for choosing this value is that Microsoft Windows uses it as default setting for PPP links. Measurements with other window sizes and MTU values were also made.

7.1 Scenario 1: “Short Fat Pipe” → “Long Thin Pipe”

In this scenario the mobile host is first connected to the WaveLAN. During a bulk data transfer, with the correspondent host as the sender and the mobile host as the receiver (download), a handover from the WaveLAN to the PPP connection is performed. The most interesting part in this scenario is how TCP reacts after the handover.

7.1.1 Expected Results

In this scenario there are only minor changes to the capacity of the link before and after the handover, but there is a significant difference in the round-trip time. The RTT increases roughly by a factor 200, which will probably cause spurious timeouts, since the RTO will have a relatively small value before the handover. Recall from Section 2-5 that Linux has a timer precision of 10 ms, which makes it possible to maintain small values of the RTO. The spurious timeouts due to the large and rapid increase of the RTT after the handover will trigger unnecessary retransmissions.

7.1.2 Measurements and Results

Figure 7-1 shows the TCP data segments sent by the correspondent host (respc15). The data segments are transmitted via the home agent over the WaveLAN the first 12 seconds. Then a handover from the WaveLAN connection to the PPP connection occurs. As can be seen in the figure, the throughput of the PPP link over GSM is substantially lower than the throughput of the WaveLAN.

The TCP segments sent and the ACKs received by the correspondent host after the handover are shown in Figure 7-2. The x-coordinate of a data segment mark represents the time in seconds the data was sent and the y-coordinate is the sequence number of the first byte in the segment. Hence data bytes 974121-975540 have the y-position 974121 in the figure. The y-coordinate of an ACK mark indicates the sequence number of the segment expected by the receiver. The figure also

[14] Netperf is a benchmark for measuring network performance, developed at Hewlett-Packard. Its primary focus is on bulk data transfer and request/response performance using either TCP or UDP and Berkeley Sockets interface. The sender creates arbitrary data that it sends to the receiver, who in turn returns acknowledgements if TCP is used and then discards the packets. Netperf is available at http://www.cup.hp.com/netperf/NetperfPage.html.

shows how the RTO value changes over time. The RTO values, obtained from the kernel, can be read from the y-axis on the right side.

Figure 7-2 shows that a window of 11 segments is sent by the correspondent host. Since the wireless link now consists of the PPP link, which is much slower than the WaveLAN, the segments will be queued at the home agent. In addition to the actual increase of the round-trip time, the queuing of segments at the home agent will also increase the round-trip time. This results in artificially high round-trip times, higher than 25 seconds for some segments.

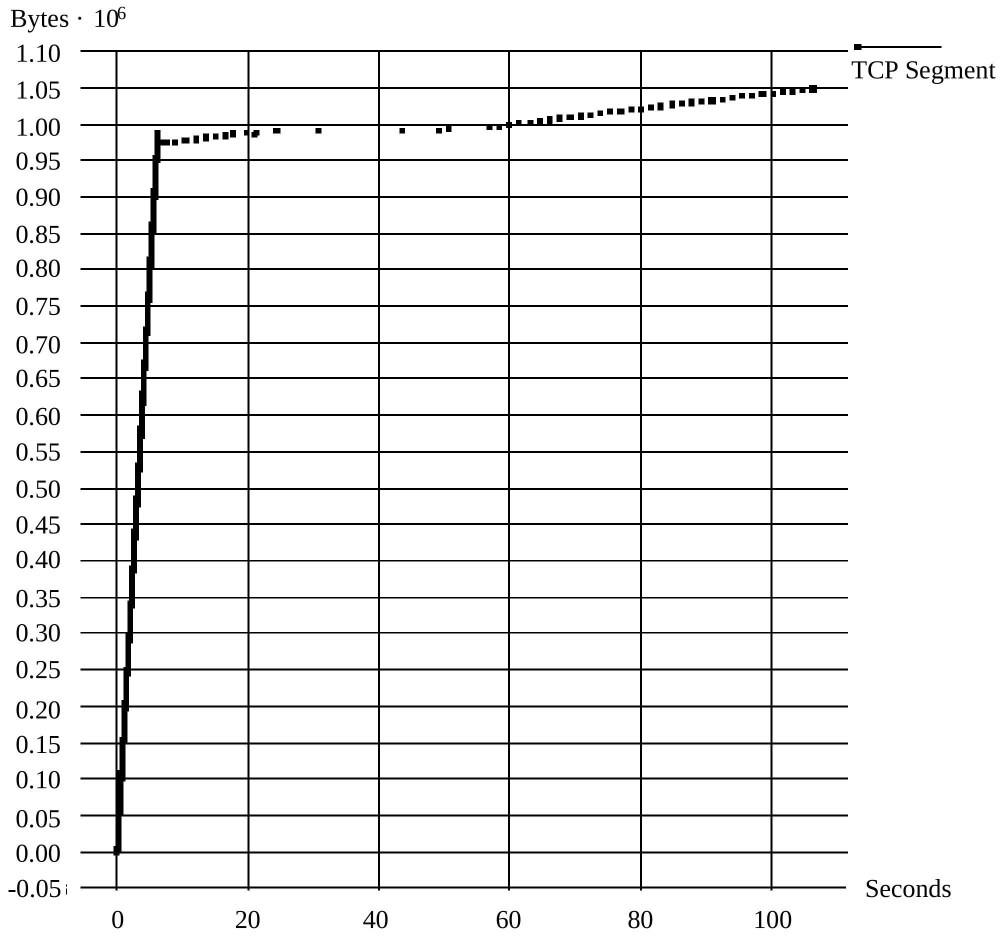

Figure 7-1: TCP data segments in a handover from WaveLAN to PPP/GSM

The RTO value is 200 ms when the handover occurs. As expected the RTO is far too small for the PPP connection, causing spurious timeouts. After the first window has been sent, a timeout of the first segment occurs three times, triggering retransmissions, exponential backoff and slow start. However, the first 11 segments are received and acknowledged by the mobile host, but due to the sudden increase of the round-trip time, the ACKs are not received before the retransmission timer expires at the sender. This causes the sender to retransmit those segments in the interval 6-22 seconds. Since the mobile host already has received the first 11 segments, it creates one duplicate ACK for each received retransmitted and already acknowledged segment. The RTO value is not recalculated when ACKs for the retransmitted segments are received (Karn's Algorithm [4] p. 301). The sender has no possibility to know if an ACK is created in response to the original or retransmitted segment. This is called the *retransmission ambiguity problem*. The updates of the RTO during the retransmissions are caused by timeouts as can be seen in figure 7-2.

Between when segment 991161 was sent (after 24 seconds) and the ACK for that segment was received, approximately 26 seconds elapsed. In this time interval, which is marked in Figure 7-2, the correspondent host only receives duplicate ACKs and makes a few retransmissions, i.e., no new segments are received by the mobile host and no new segments are sent by the correspondent host. TCP does not use the fast retransmit and fast recovery algorithms, although more than three duplicate ACKs are received. After the 15 duplicate ACKs the packet exchange seems to stabilize, except for three more duplicate ACKs, which correspond to the retransmissions during the marked time interval in the figure.

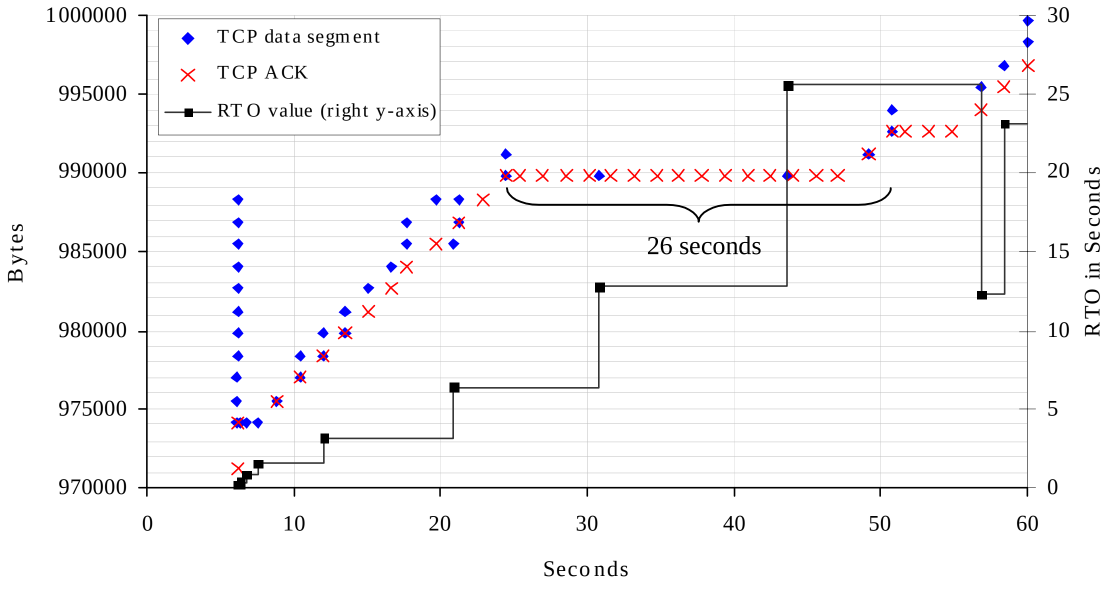

Figure 7-2: TCP data segments sent and ACKs received by the correspondent host after the handover to PPP, and the current RTO value

Figure 7-3 visualizes the entire packet exchange after the handover. A line is included in the figure to show the maximum theoretical throughput of the PPP link. As shown in the figure, the transmission over the PPP link continues almost 40 seconds longer than it would have had if the link achieved its theoretical throughput. However, the dotted line (theoretical throughput) and the line consisting of the ACKs (real throughput) look fairly parallel. The calculated value of the mean throughput is 900 bytes/s in the time interval 58-116 seconds, compared to the theoretical throughput of 960 bytes/s. The mean throughput of the entire data transfer after handover is 680 bytes/s. This indicates that the 26 seconds interval when no new data is transferred is the main reason for the poor throughput. Worth noting is that the mean throughput strongly depends on the total length of the transfer.

Worth noting are the increasing round-times for the segments, which can be seen by the increasing distance between the TCP data segments and the correspondent ACKs in Figure 7-3. After the many duplicate ACKs the round-trip time is approximately 1.5 seconds, but at the end of the connection the round-trip time has increased to almost 11 seconds. This indicates queuing at the home agent's outgoing PPP interface, which is the bottleneck due to the slow PPP link. The figure also shows an increase of the RTO value even after the packet exchange has stabilized. The RTO has a value of about 20 seconds at the end of the transfer. Quite remarkable is that it takes more than 30 seconds for TCP to recover after the handover.

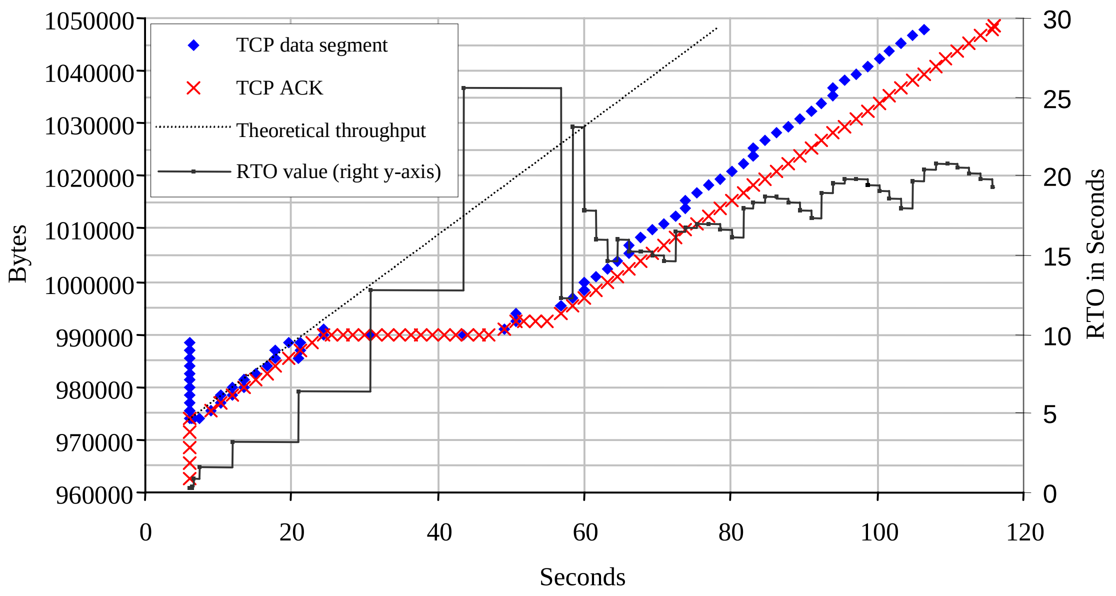

Figure 7-3: TCP segments and ACKs collected at the correspondent host after the handover. The RTO values and the theoretical throughput of the PPP link are also included.

7.1.3 Varying the Advertised Window and the MSS

The measurements in the previous section showed the occurrence of a retransmission of an entire window, directly after handover. If the advertised window is smaller the number of retransmitted packets will decrease. As noted from the previous measurement a large window size also results in filling the buffer at the home agent, which increases the round-trip time. The maximum segment size in these measurements was 1024 bytes, making it easier to maintain the window size in kbytes.

The appropriate window size varies, depending on the capacity of the link and the segment size. Theoretically, it would be sufficient to have a window size equal to the capacity of the link plus one segment. The extra segment is needed since the receiver cannot acknowledge a segment until it is completely received and hence removed from the pipe, leaving a space of one segment size in the pipe. We made measurements with window sizes of 2, 4, 8 and 16 kbytes. In Section 6.2 the capacities of the different media were calculated. Including the Ethernet the pipe between the mobile host and the correspondent host should have a capacity of approximately 1500 bytes. One segment of 1024 bytes is not enough to fill the pipe. Since an extra segment also is needed the window size 4 kbytes is the smallest window (of the above listed sizes), that will keep the pipe filled.

Figure 7-4 illustrates the packet TCP data segments and ACKs at the correspondent host. The advertised window by the receiver is 4 kbytes. Since only four segments is enough to fill up the window, the retransmissions directly after the handover will only concern four segments. The figure shows a significant improvement of the throughput and a much lower stable value of the RTO. The stable RTO value and the constant interval between data segments and ACKs, indicate that segments are not being queued up by a bottleneck. The calculated throughput for the same number of bytes as in the previous measurement is 790 bytes/s, which is an improvement of 16% compared to the measurements with a window size of 16 kbytes and an MSS of 1420 bytes. Observations of the other window sizes were also made, but the throughput decreased.

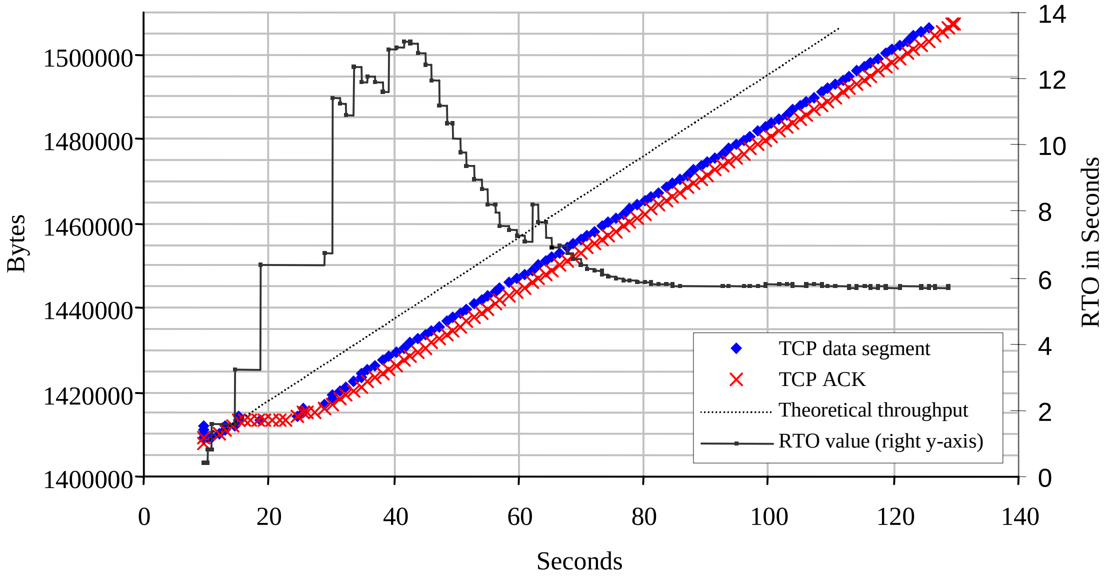

Figure 7-4: The packet exchange after the handover with a advertised window of 4096 bytes and an MSS of 1024 bytes

The problem with a too small RTO value after the handover remains, resulting in three retransmissions of the first segment. If the segment size is reduced, the transmission delay[15] for the segment will decrease. This decrease is notable over the GSM link since the data rate is 960 bytes resulting in a transmission delay of about one millisecond per byte. A smaller RTT could possibly reduce the number of retransmissions of the first segment after the handover.

TCP uses a maximum segment size of 536 bytes as a default value when the correspondent host belongs to another subnet, or when the MSS option is not used in the connection establishment, which makes it interesting to investigate if any differences exist in comparison with a large MSS value.

A window of four segments of 536 bytes should be enough to fully use the available bandwidth of the pipe. Figure 7-5 illustrates the packet exchange after the handover using an MSS of 536 bytes and a window size of 2144 bytes. The problem with several retransmissions of the first segment sent after the handover remains, as shown in the figure. The calculated throughput is 700 bytes/s.

[15] The time to send a packet depends mainly on *the propagation delay* and *the transmission delay* The propagation delay is caused by the finite speed of the signal in the media and latencies in the transmission equipment, while the transmission delay depends on the data rate of the media. The transmission delay, which increases with the packet size, is dominant for slow links (see [4] p. 289).

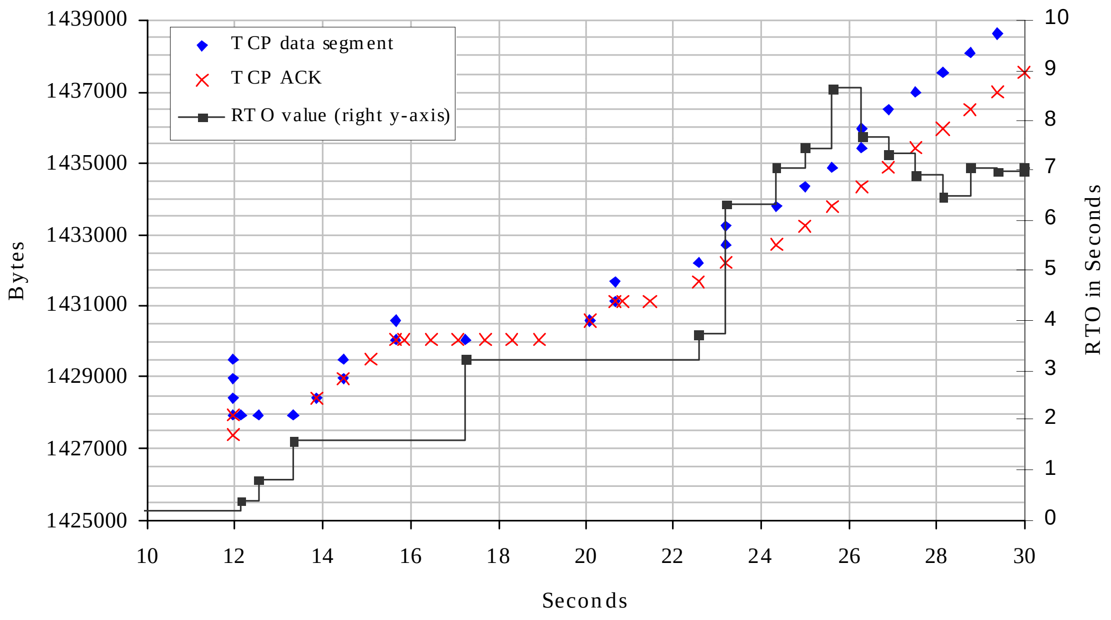

Figure 7-5: The packet exchange after the handover with an advertised window of 2144 bytes and a MSS of 536 bytes

7.1.4 Upload

As mentioned in Chapter 6, we modified the handover procedure in the MosquitoNet Mobile IP implementation to avoid differences between a download and an upload. To verify this upload measurements with the mobile node as the sender and the correspondent host as the receiver were also made.

Figure 7-6 illustrates the packet exchange right after the handover with a time line diagram, created from the tcpdump output. The sequence number for the first TCP segment sent after the handover has been set to ‘0’ as well as the time when it was sent. The labels on the far left of the figure represents the time in seconds when segments are sent and received by the mobile host. The numbers within the brackets are the time interval between two events. The labels at the top include the IP addresses and the port number of the two communicating hosts. Transmitted TCP data is shown by thicker lines.

As can be seen from Figure 7-6, the first 1460 data bytes are retransmitted three times before the first ACK arrives at the mobile host. It is also obvious that all four segments are received correctly by the correspondent host. The data bytes 1460-2919 are sent three times. Also in this case all three segments containing data bytes 1460-2919 arrive at the receiver. After this, the packet exchange seems to be quite normal.

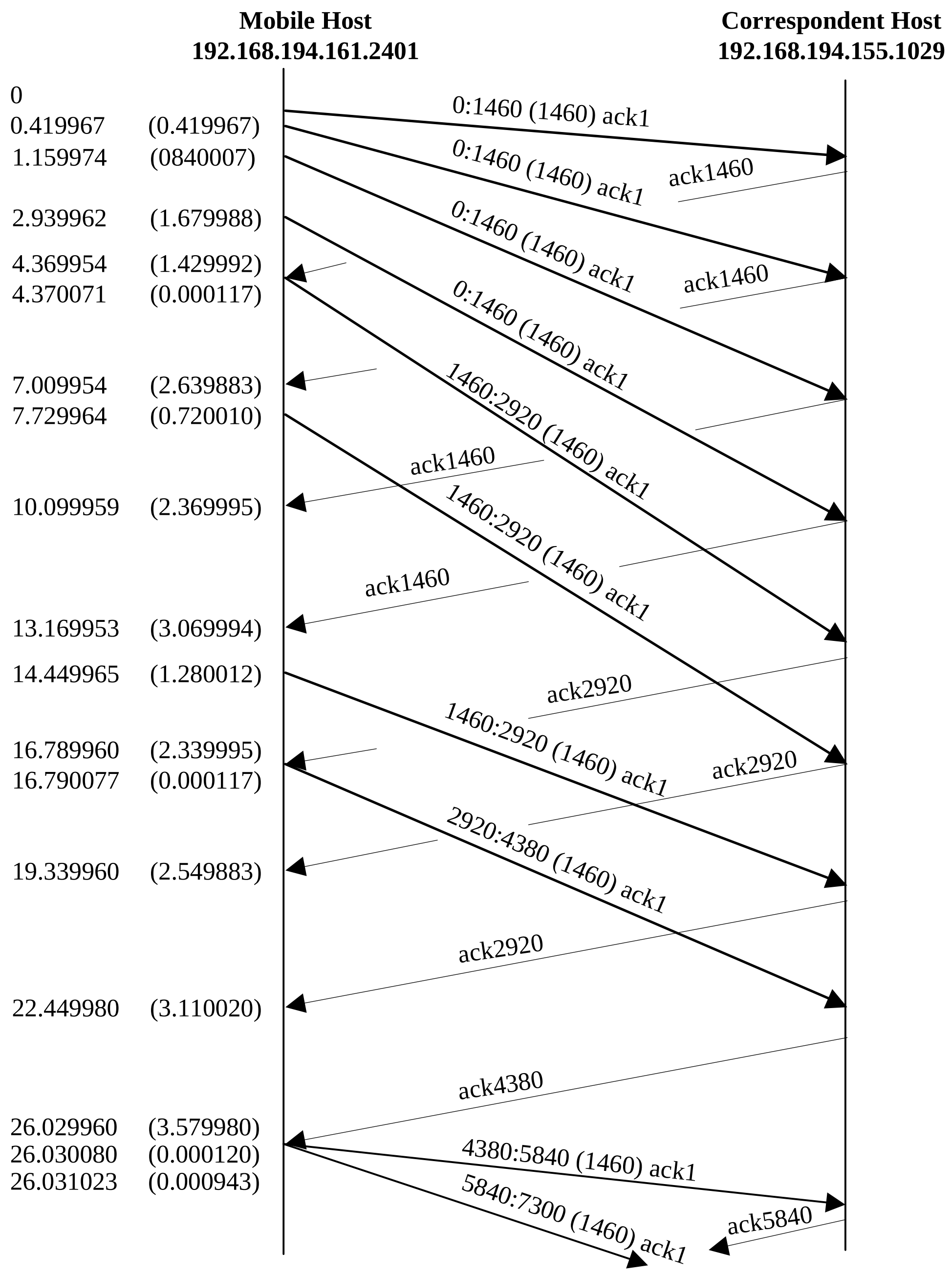

Figure 7-6: Time line of the packet exchange after the handover from the WaveLAN connection to the PPP connection

The TCP data segments sent and the ACKs received by the mobile host after the handover are illustrated by Figure 7-7. The figure also shows a dotted line, which represents the maximum theoretical throughput of the PPP link.

Figure 7-7 indicates a usable window of only one segment at the mobile host, since only one segment is sent directly after handover. The distance between the ACKs and the TCP data segments is growing steadily. The RTT reaches a value of almost twenty seconds at the end of the data transfer. Comparing the number of bytes sent and acknowledged with the theoretical throughput line, shows there is a poor throughput. The calculated value of the actual throughput obtained over the PPP link is less than 390 bytes. More measurements of bulk data transfer from the mobile host to the correspondent host showed a constant low throughput of less than half the theoretical value.

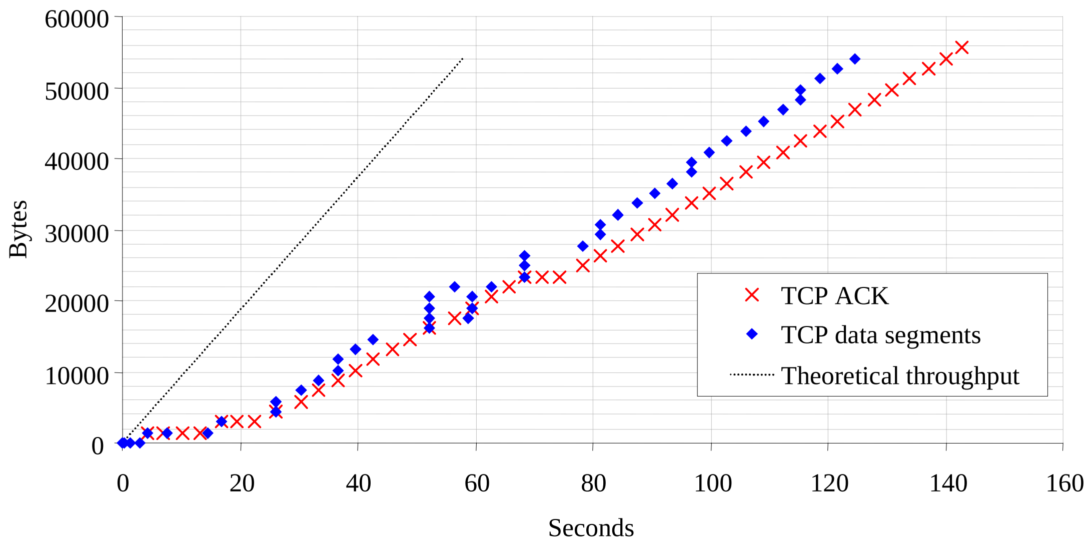

Figure 7-7: TCP segments sent and ACKs received by the mobile host after the handover.

After double checking the modem parameters and flow control settings between the computer and the modem, we still do not have a clear indication of the reason for the poor throughput. In meaurements using File Transfer Protocol (FTP) [31] for the bulk data transfer instead of Netperf normal throughput was obtained, possibly indicating a problem with Netperf. Since we have modified the handover procedure of the MosquitoNet Mobile IP implementation, similar results in upload and download measurements were expected. Due to the inability to locate the source of our problem, the remaining measurements will be made only in download direction.

7.1.5 Possible Ways to Improve Performance

The problems TCP has encountered when the link characteristics suddenly change are spurious timeouts due to the minimal value of the RTO compared to the RTT of the new link, unnecessary retransmissions as a consequence of the spurious timeouts and artificially high round-trip times.

One way to avoid the spurious timeouts after the handover could be to decrease the maximum segment size (MSS) over the PPP link. The results in Section 7.1.3 show that this approach does not reduce the number of timeouts.

As seen from the measurement results a large advertised window causes extensive queuing at the home agent in the case of a download in our setup. In the case of an upload the mobile host TCP sends the packets to the link layer buffer where they are buffered if the link is slow. However, TCP considers the packets sent even if they are stored in the buffer for a while, and hence the RTT takes on an artificially high value. The artificially high RTT leads to an exaggerated high value of the RTO, causing timeouts and retransmissions to occur much later than they could have.

To keep the round-trip times as small as possible, we used a small window size. By having a small window size the throughput increased as shown in Section 7.1.3. The performance improvements are expected to be even more notable if packet losses occur over the link.

Since redundant retransmissions occur of an entire window after a handover, adding unnecessary load to the network and resulting in poor utilization of the slow wireless link, a small window would limit the number of retransmissions and mitigate the problem. However, the approach of setting the window size to an appropriate static value is quite simplistic, since a suitable window size depends on the link characteristics and the segment size which are dynamic variables over the Internet and thus unpredictable in the general case.

A better approach to avoid excessive window sizes and prevent the buffer at the bottleneck from getting filled up would be to use *Active Queue Managem*ent [30], which basically is a congestion avoidance mechanism at routers preventing the buffers from filling up. *Random Early Detection* (RED) is a recommended active queue management mechanism. To maintain a reasonable size of the queues, RED drops packets before the queues are filled up. The authors of [30] expect that the RED algorithm will provide significant performance improvement. It would be interesting to implement an active queue management mechanism such as RED at both endpoints of the wireless link, which often is the bottleneck.

TCP is unaware of the handover, which caused rapid changes in link characteristics. If TCP could recognize that a switch from a link with low round-trip delay to a link with high round-trip delay is about to occur, TCP could set the RTO value to a higher value. This would prevent the retransmission timer at the TCP sender from expiring after the handover. Another parameter that could be set to a more appropriate value is the congestion window (cwnd). For example setting cwnd to one segment and entering slow start to explore the characteristics of the new link would probably be better from a performance point of view, than keeping the values of these parameters from the old link. This solution could be implemented on the mobile host, but would require some communication between TCP and the link layer. However, if the TCP sender is the correspondent host there is no way for it to know when the mobile host performs a handover. Thus, the effect of this solution would only be noticed for upload traffic, when the mobile host acts as the TCP sender. One way to change the values of the RTO and the cwnd if the sender is the correspondent host could be to insert these values as TCP options in the header.

7.2 Scenario 2: “Long Thin Pipe” → “Short Fat Pipe”

In this scenario the mobile host is first connected to the PPP link. During a bulk data transfer, with the correspondent host as the sender and the mobile host as the receiver (download), a handover from the PPP to the WaveLAN connection is performed.

7.2.1 Expected Results

The round-trip time decreases by a factor 200, but the capacity of the link stays nearly constant. No timeouts and retransmissions are expected as a consequence of the handover. However, TCP could perform very poorly if packets were lost *during or directly after the handover*, since the RTO value at that moment probably would be much larger than the round-trip time of the new link. In this case it would take a substantial amount of time before the retransmission timer at the TCP sender expired and triggered retransmissions. The result would be very low utilization of the available bandwidth. Since no timeouts and retransmissions are expected, the interesting part in this scenario is to observe the RTO value before the handover and also how quickly it adapts to the new round-trip time after the handover.

As noted in Scenario 1, the home agent is the bottleneck in the case of a download resulting in queued packets in the buffer. The packets in the send buffer of the PPP interface at the home agent will not be redirected since they have already been sent to the link layer buffer. Thus, the packets in the buffer will be sent over the PPP link *even though* the handover is completed. Since the PPP link is much slower, the receiver (the mobile host) will probably receive some packets over the WaveLAN before the old packets from the home agent’s PPP buffer arrive. This will probably result in packets being delivered out-of-order, causing duplicate ACKs to be sent by the mobile host. If the usable window at the TCP sender is larger than the size of three segments, there is a possibility that fast retransmission will occur. However, the retransmitted packets will be sent across the WaveLAN interface and the packets arriving later via the PPP link will simply be discarded.

7.2.2 Measurements and Results

In Figure 7-8 the TCP data segments sent from the correspondent host are shown. The figure gives an illusion of the handover from PPP to WaveLAN being performed after 85 seconds without any timeouts or retransmissions. However, according to the tcpdump output, the handover occurs already after 70 seconds. The segments queued up in the PPP send buffer at the home agent are the reason why the bandwidth is so poorly utilized the first 15 seconds after handover.

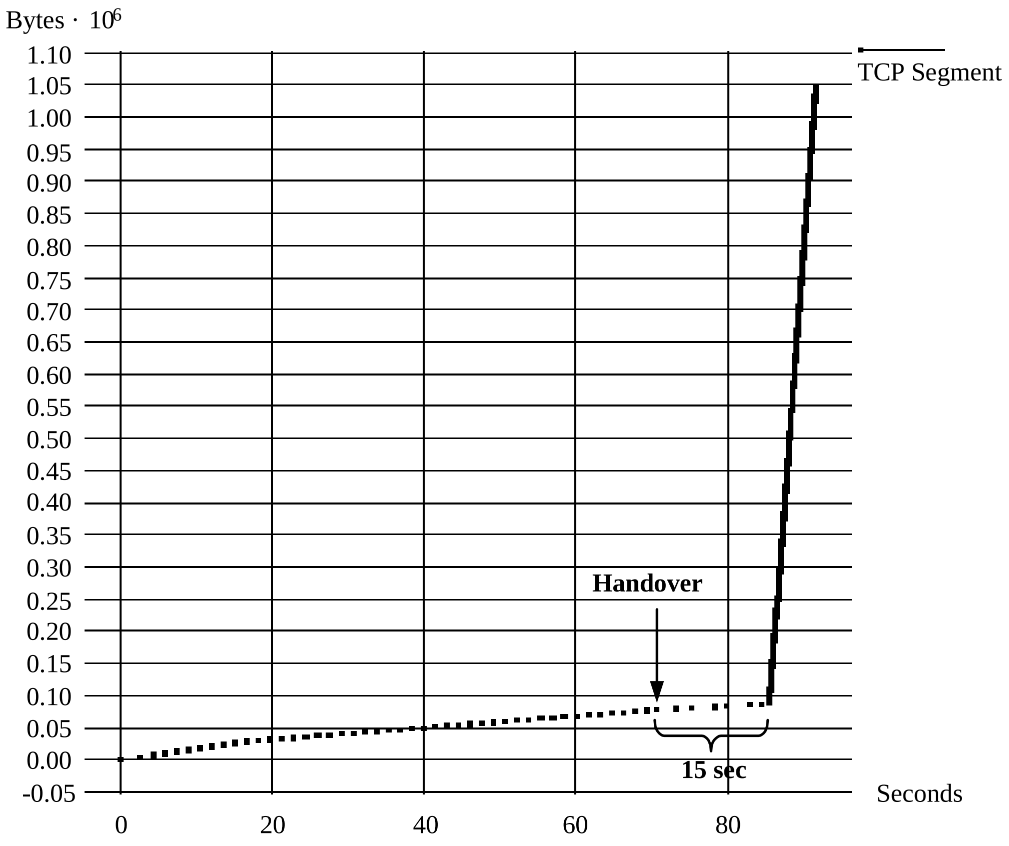

Figure 7-8: TCP data segments sent by the correspondent host before, during and after a handover from PPP/GSM to WaveLAN

Figure 7-9 shows the exchange of TCP segments between the home agent and the mobile host directly after the handover. Both the PPP interface and the WaveLAN interface of the two nodes are included in the figure. Note that the sender is the correspondent host, respc15, and not the home agent.

Segments are still queued up at the home agent's outgoing PPP buffer after handover has been completed. Figure 7-9 indicates that 11 segments (Segment 1 in the figure and the ten buffered segments) arrive at the PPP interface of the mobile host after the handover.

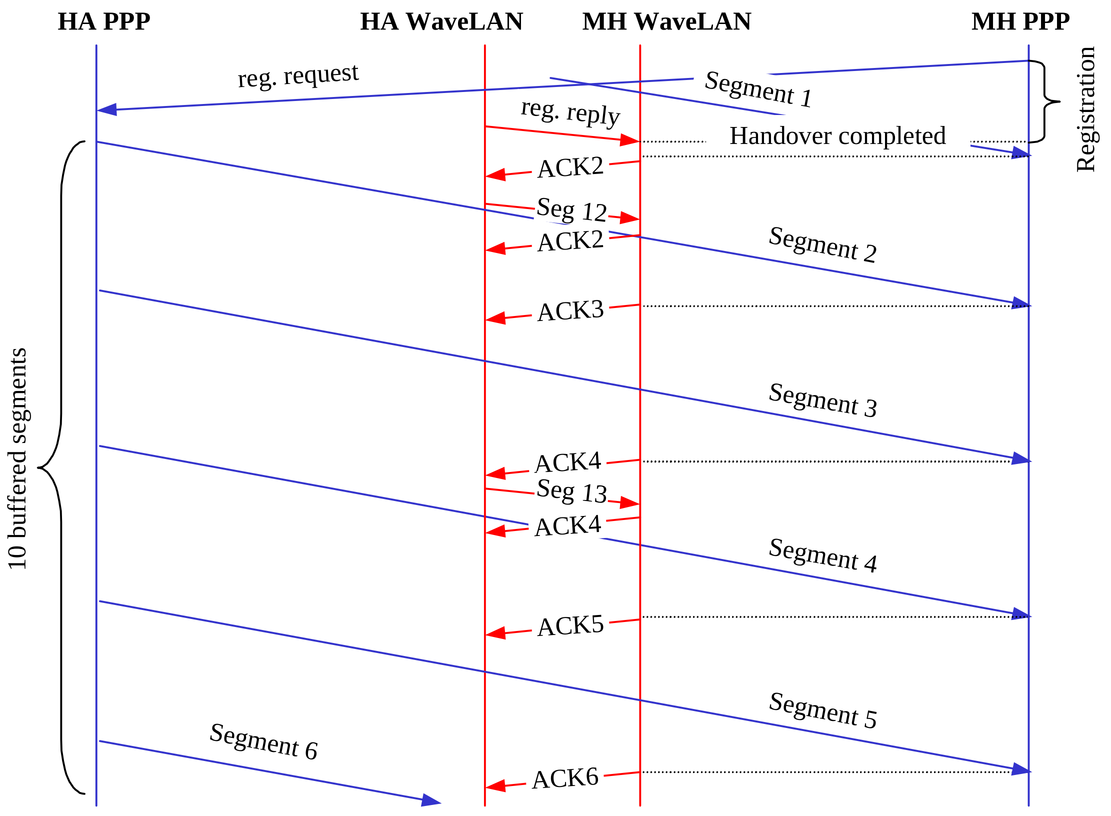

Figure 7-9: Time line diagram of the TCP segment exchange between the home agent (HA) and the mobile host (MH) after the handover from PPP to WaveLAN

An entire TCP window is unacknowledged, since the segments are queued up at the home agent, resulting in a closed usable window at the TCP sender (respc15), i.e., no segments can be sent until new ACKs arrive at the sender. When the acknowledgement for Segment 1 arrives at the sender (ACK2) the next segment (Segment 12) can be sent. This is the first TCP data segment sent over the WaveLAN after the handover. Since the WaveLAN has a much higher bandwidth and shorter RTT than the PPP link, Segment 12 is received by the mobile host before Segment2, which causes a duplicate ACK to be sent by the mobile host. This pattern continues until the PPP buffer has been emptied, and the segments have been acknowledged, resulting in a poor utilization of the high available bandwidth over the WaveLAN for the first 15 seconds after handover.

Figure 7-10 shows the packet exchange over the PPP link. What is interesting in this figure are the RTO values. The segments start to go over the WaveLAN after approximately 86 seconds and the round-trip time decreases significantly. However, the RTO value increases from 20 seconds to more than 50 seconds, as shown in the figure. The reason for the increase of the RTO value despite the decrease of the round-trip time, is that the algorithm for calculating the RTO adds the variance of the measured RTTs even when the RTT is decreasing (see [15]). The algorithm for the RTO value works well when the probability of an increase of the RTT is equal to the probability of a decrease. However, if the RTT suddenly decreases significantly and several RTT measurements show the same value, the RTO should be decreased to a value close to the new RTT, instead of adding the variance.

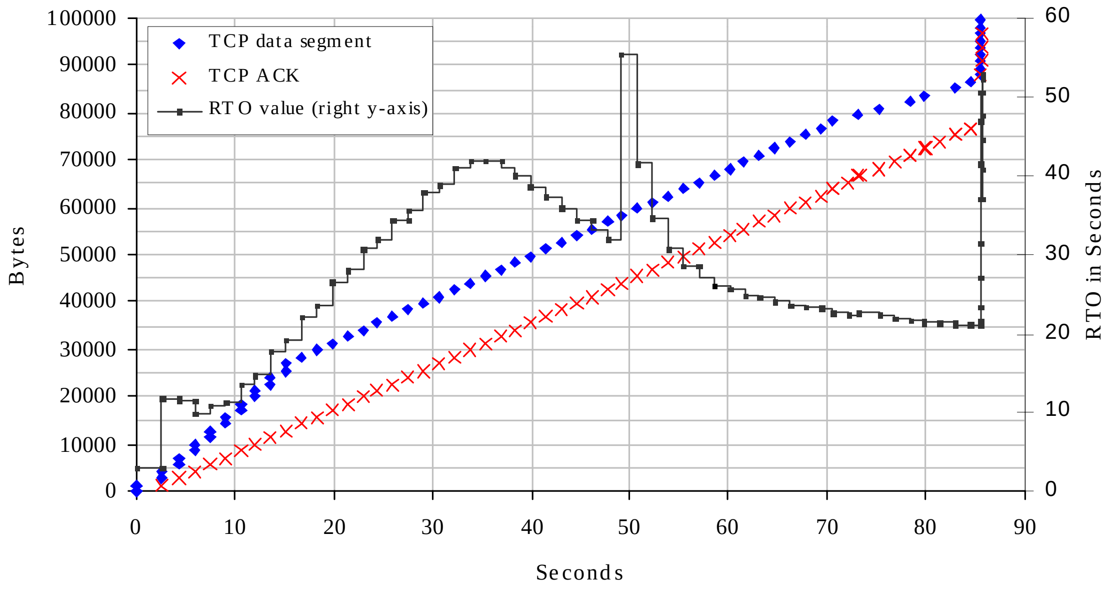

Figure 7-10: Packet exchange over PPP and RTO values

Figure 7-11 illustrates the first part of the packet exchange over the WaveLAN. The figure shows that it takes a large amount of updates of the RTO value until it stabilizes to a value of approximately 200 ms.

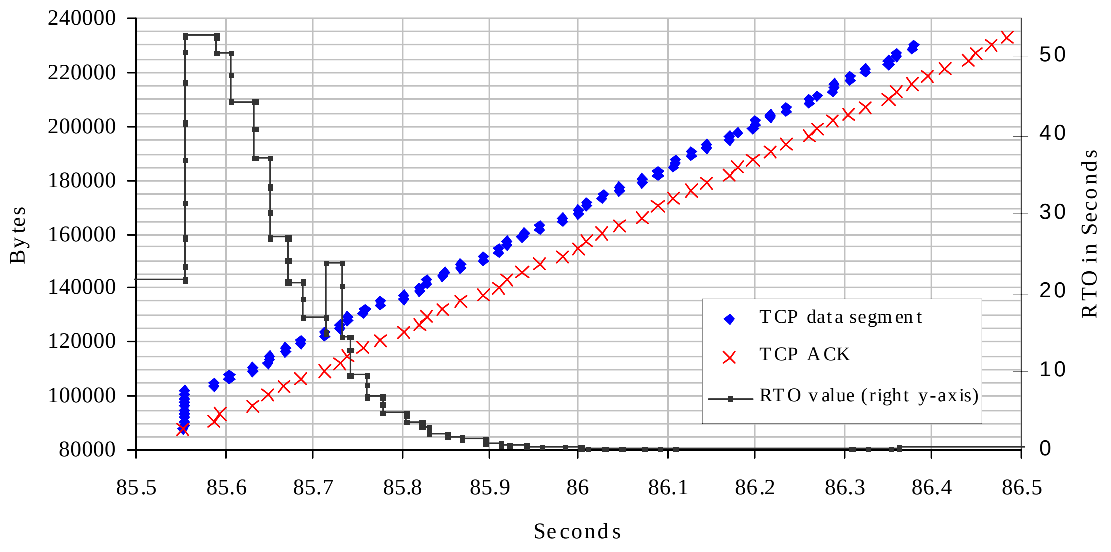

Figure 7-11: Packet exchange over the WaveLAN and RTO values

If a segment would be lost or corrupted just after the segments have started being sent over the WaveLAN, it would take a large amount of time until the segment is retransmitted. Since the RTO value is much higher than the round-trip time (more than 250 times higher RTO in the worst case) the sender would wait much a longer time than necessary to retransmit the lost segment.

7.2.3 Possible Ways to Improve Performance

An entire window is queued at the home agent's PPP buffer. If the window size is large, the time before the data packets can be exchanged completely over the WaveLAN is substantial. In our measurements we had a period of 15 seconds after the handover, when segments still were sent over the PPP link. As mentioned in Section 7.1.5 an interesting approach to avoid such problems with large window sizes would be to use Active Queue Management.

It is desirable to reroute packets to the higher bandwidth interface after the handover. However, the packets already sent to the link layer queue, cannot possibly be rerouted to another interface. To reduce the number of packets queued in the link layer buffer, the IP sending process should delay putting packets in the queue of a slow link. This would enable utilization of the high bandwidth link faster and could be achieved by flow control between the IP layer and the link layer.

Another solution could be that the sender discards the queued packets in its buffer, in the case of a handover from the slower PPP link to WaveLAN. This would cause the sender to timeout and retransmit the packets. However, as confirmed by the measurement carried out the RTO value at the sender will be quite high, i.e., the retransmission timer will expire after several tens of seconds in some cases. Together with the solution proposed in Section 7-5, that would set the RTO to a new value when handover is performed this approach could improve handover performance significantly. However as mentioned in Section 7-5, these kinds of solutions are only possible at the mobile host or in a local environment. If the sender is a host somewhere in the Internet, not much can be done to inform the sender of the handover.

CHAPTER 8

Conclusions and Future Work

8.1 Conclusions

Many studies have been made on IP mobility impact on TCP performance. This study has investigated the TCP problems caused by rapid changes of the link characteristics due to handover in a wireless mobile environment.

One goal of the thesis was to create a realistic measurement environment, which has been accomplished. The testbed works well except for the poor throughput for upload data transfer (from the mobile host to the correspondent host), as measured when using Netperf. Exact monitoring of packet flow and TCP behavior is possible with the testbed, making it a useful environment for future investigations.

Our measurements show that TCP encounters problems when significant and rapid changes of the link characteristics occur. Handover from a high bandwidth link with low round-trip delay to a low bandwidth link with high round-trip delay (WaveLAN to PPP over GSM in our measurements), causes spurious timeouts triggering unnecessary retransmissions of an entire window. The reason for the spurious timeouts is the sudden increase of the RTT. In some cases it takes TCP more than 40 seconds to recover after the handover. To mitigate the problems with the unnecessary retransmissions we recommend the use of a window size as small as possible without undermining the use of the available link capacity. A smaller window size will also reduce the artificially high round-trip time.

In the case of a handover from a low bandwidth link with high round-trip delay to a high bandwidth link with low round-trip delay, we discovered a poor utilization of the high bandwidth available after handover. The reason is queued packets in the buffer associated with the low bandwidth link. These packets will be sent over the slow link even after handover has occurred. Thus, the high bandwidth link will not be used for sending data segments until the queued packets at the low bandwidth link have been sent. To reduce the number of packets queued in the buffer, a small window size should be used. The IP sending process should delay putting packets in the queue of a slow link to be able to reroute packets when a faster interface is available after handover. Another problem in this scenario was the unnecessarily high RTO value. The algorithm for the RTO value works well when the probability of an increase of the RTT is equal to the probability of a decrease. However, if the RTT suddenly decreases significantly and several RTT measurements show the same value, the RTO should be decreased to a value close to the new RTT, instead of adding the variance.

In both handover scenarios investigated, it is desirable to use a window size as small as possible. Using Active Queue Management at both ends of the wireless link will probably prevent the use of large window sizes. Investigation of the impact of using Active Queue Management should be made in future work.

Another possible solution is to reset the RTO value after a handover to an initial value. This would prevent the retransmission timer from expiring when the RTT increases after the handover. In a handover from a slow link to a fast link, discarding the packets queued in the sender buffer at the moment of handover would probably improve TCP performance. These approaches require some communication between the link layer and the transport layer. It is however impossible for the correspondent host to know when the mobile host will perform a handover. Local modifications, e.g., changes at the mobile host can be made, but will only improve the performance when the mobile node is the sender. If the correspondent host is the TCP sender, a notification of a more appropriate RTO value after handover could be included in the TCP flow as an option in the header.

8.2 Future Work

For future work, measurements according to the two additional scenarios described in Section 6.4.1 would be interesting to study.

The problem with the poor throughput when uploading data from the mobile host to the correspondent host should be solved.

An Active Queue Management mechanism such as Random Early Detection should be integrated into the testbed at both endpoints of the wireless link in order to investigate if the performance improves using this method.

The proposed solutions to reset the RTO value and possibly enter slow start at the moment of a handover, should be implemented at the mobile host. Flow control between the IP layer and the link layer could improve performance significantly in the case of handover from a high bandwidth link to a low bandwidth link. Measurements should be made to evaluate the positive impact of these changes on TCP performance.

The problems with high RTO values when switching from a slow link to a fast link could cause major delays in the case of packet losses. Handover with packet losses should be studied to investigate the behavior of TCP in case of handover between non-overlapping cells. To have a more accurate RTO value, the algorithm should be modified to achieve a fast adaptation of the RTO value in the case of rapid and significant decrease of the RTT, due to handover from a slow link to a fast link.

List of Abbreviations

ACK	Acknowledgement
BOOTP	Bootstrap Protocol
BS	Base Station
CH	Correspondent Host
CN	Correspondent Node
COA	Care-Of Address
cwnd	Congestion window
DHCP	Dynamic Host Configuration Protocol
ESBN	Explicit Bad State Notification
FA	Foreign Agent
FTP	File Transfer Protocol
GRE	Generic Record Encapsulation
GSM	Global System for Mobile Communication
HA	Home Agent
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol version 4
ISP	Internet Service Provider
I-TCP	Indirect Transmission Control Protocol
LAN	Local Area Network
MH	Mobile Host
MN	Mobile Node
MPT	Mobile Policy Table
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
PPP	Point-to-Point Protocol
RFC	Request for Comments
RED	Random Early Detection
RTO	Retransmission Timeout
RTT	Round-Trip Time
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
TTL	Time-To-Live
UDP	User Datagram Protocol
vif	Virtual Interface
WSP	Wireless Socket Protocol

References

[1] Charles E. Perkins, *IP mobility support*, Request for Comments: 2002, http://www.stacken.kth.se/doc/rfc/rfc2002.txt, Network Working Group, October 1996

[2] Charles E. Perkins, *Mobile IP: Design Principals and Practice*, Addison-Wesley Longman, Reading, Massachusetts, 1997

[3] Charles E. Perkins, *Mobile Networking Through Mobile IP*, http://computer.org/internet/v2n1/perkins.htM

[4] W. Richard Stevens, *TCP/IP Illustrated, Volume 1, The Protocols*, Addison-Wesley Longman, Reading, Massachusetts, 1994

[5] W. Richard Stevens, *TCP Slow Start*, *Congestion Avoidance*, *Fast Retransmit, and Fast Recovery Algorithms*, Request for Comments: 2001, http://www.cis.ohio-state.edu/htbin/rfc/rfc2001.html, Network Working Group, January 1997

[6] G. Montenegro, S. Dawkins, M. Kojo, V. Magret, *Long Thin Networks*, Internet draft, ftp://ftp.ietf.org/internet-drafts/draft-montenegro-pilc-ltn-00.txt, November 1998

[7] *Stanford MosquitoNet Project MobileIPv4 Distribution, Users Manual, Release 1.0.5,* http://mosquitonet.stanford.edu/software/mip.html, Stanford University, April 1999

[8] Xinhua Zhao, Mary Baker, *Flexible Connectivity Management for Mobile Hosts, Technical Report: CSL-TR-97-735*, http://mosquitonet.stanford.edu/software/mip.html, Stanford University, September 1997

[9] Ajay Bakre, B.R. Badrinath, I-TCP: Indirect TCP for Mobile Hosts, Rutgers, ftp://paul.rutgers.edu/pub/badri/itcp-tr314.ps.Z, University, Piscataway, October 1994

[10] Ajay Bakre, B.R. Badrinath, *Handoff and System Support for Indirect TCP/IP*, Rutgers University, Piscataway, in Proc. Second Symposium on Mobile and Location-Independent Computing, pages 11-24, USENIX, April 1995

[11] Hari Balakrishnan, Srinivasan Seshan, Randy H. Katz, *Improving Reeliable Transport and Handoff Performance in Cellular Wireless Networks*, in ACM Wireless Networks, December 1995

[12] Bikram S. Bakshi, P. krishna, N. H. Vaidya, D. K. Pradhan, *Improving Performance of TCP over Wireless Networks, Techical Report TR-96-014*, Texas A&M University, May 1996

[13] Viptul Gupta, *Solaris Mobile IP: Design and Implementation*, SUN Microsystems Inc., Palo Alto, California, February 1998

[14] David A. Maltz, David B. Johnson, *The CMU Monarch Project IETF Mobile IPv4 Implementation User's Guide*, Carnegie Mellon University, June 1997

[15] Van Jacobson, Michael J. Karels, *Congesion Avoidance and Control*, in Proceedings of SIGCOMM '88, Stanford, California, August 1988, ACM

[16] Mary G. Baker, Xinhua Zhao, Stuart Cheshire, Jonathan Stone, *Supporting Mobility in MosquitoNet*, in the Proceedings of the 1996 USENIX Technical Conference, San Diego, CA, January 1996

[17] Ralph Droms, *Dynamic Host Configuration Protocol*, Request for Comments: 2131, http://sunsite.hr/cgi-bin/rfc/rfc2131.txt, Network Working Group, March 1997

[18] C. Perkins, *IP encapsulation within IP*, Request for Comments: 2003, http://www.sunsite.auc.dk/RFC/rfc/rfc2003.html, Network Working Group, October 1996

[19] Jon Postel, Editor, *Internet Protocol*, Request for Comments: 791, http://www.sunsite.auc.dk/RFC/rfc/rfc791.html, September 1981

[20] S. Deering, *ICMP Router Discovery Messages*, Request for Comments: 1256, http://www2.hunter.com/docs/rfc/rfc1256.html, Network working Group, September 1991

[21] Van Jacobson, *Compressing TCP/IP headers for Low-Speed Serial Links*, Request for Comments: 1144, http://www2.hunter.com/docs/rfc/rfc1144.html, Network Working Group, February 1990

[22] Ramon Caceres, Venkata N. Padmanabhan, *Fast and Scalable Handoffs for Wireless Internetworks*, in Proceedingsof ACM MobiCom '96, November 1996

[23] Jon-Olov Vatn, Gerald Q. Maguire Jr., *The effect of using co-located care-of addresses on macro handover latency*, http://www.it.kth.se/~vatn

[24] Stephan Baucke, Yuri Ismailov, Mikael Latvala, Reiner Ludwig, Michael Meyer, *Wireless TCP – Problems and Recommendations*, Technical Report EED/R-98:595

[25] Bill Croft, John Gilmore, *Bootstrap Protocol (BOOTP)*, Request for Comments: 951, http://www.sunsite.auc.dk/RFC/rfc/rfc951.html, September 1985

[26] Jon Postel, *Transmission Control Protocol (TCP)*, Request for Comments: 793, http://www.sunsite.auc.dk/RFC/rfc/rfc793.html, September 1981

[27] Xinhua Zhao, Claude Castelluccia, Mary Baker, *Flexible Network Support for Mobility*, in the Proceedings of ACM/IEEE MobiCom '98

[28] James D. Solomon, *Mobile IP: The Internet Unplugged*, Prentice Hall, 1997

[29] *WaveLAN Wireless LAN Technology and Market Backgrounder*, Lucent Technologies, Bell Labs Innovations, http://wavelan.wavelan.com/about/wavelan.html

[30] B. Braden, et al., *Recommendations on Queue Management and Congestion Avoidance in the Internet*, Request for Comments: 2309, http://www.sunsite.auc.dk/RFC/rfc/rfc793.html Network Working Group, April 1998

[31] Jon Postel, K. Reynolds, File Transfer Protocol (FTP), Request for Comments: 959, http://www.sunsite.auc.dk/RFC/rfc/rfc793.html, Network Working Group, October 1985