NetworkPerf – A tool for the investigation of TCP/IP network performance at Saab Transpondertech

by

Magnus Johansson

LIU-IDA/LITH-EX-A--09/039--SE

2009-08-13
Final thesis

NetworkPerf - A tool for the investigation of TCP/IP network performance at Saab Transpondertech

Version 1.0.2

by

Magnus Johansson

LIU-IDA/LITH-EX-A–09/039–SE

2009-08-13

Supervisor: Hannes Persson, Attentec AB
Examiner: Dr Juha Takkinen, IDA, Linköpings universitet
Abstract

In order to detect network changes and network troubles, Saab TransponderTech needs a tool that can make network measurements.

The purpose of this thesis has been to find measurable network properties that best reflect the status of a network, to find methods to measure these properties and to implement these methods in one single tool. The resulting tool is called NetworkPerf and can measure the following network properties: availability, round-trip delay, delay variation, number of hops, intermediate hosts, available bandwidth, available ports, and maximum allowed packet size.

The thesis also presents the methods used for measuring these properties in the tool: ping, traceroute, port scanning, and bandwidth measurement.
Acknowledgments

This master’s thesis would not be half as good as it is, if I had not received help and support from several people.

Many thanks to my examiner Dr Juha Takkinen, without whose continuous feedback this report would not have been more than a few confusing pages.

I would also like to send my greatest thanks to all the people at Attentec, for letting me into their fellowship and for helping me with academic issues as well as more practical problems. It has been a great time being a part of Attentec! Special thanks to Hannes Persson for being such a supportive supervisor and to Daniel Petersen for coping with my sometimes terrible language when proofreading this report.

To Martin Pettersson and other at Saab Transpondertech I also would like to send my thanks, for coming up with the idea that became this master’s thesis, for answering my questions and for helping me with the testing and evaluation.

Finally I would like to thank my girlfriend Annelie for always believing in me. Your support has been needed.
## Contents

1 Introduction .................................................. 1  
   1.1 Background ............................................. 1  
   1.2 Problem description .................................... 2  
   1.3 Purpose .................................................. 2  
   1.4 Scope ................................................... 3  
   1.5 Method ................................................... 3  
   1.6 Structure of the report ................................. 3  

2 Automatic Identification System and Vessel Traffic Service 5  
   2.1 Automatic Identification System ......................... 5  
   2.2 Vessel Traffic Service .................................. 6  

3 TCP/IP networks and performance measurement .................. 8  
   3.1 Network terminology ..................................... 8  
      3.1.1 Physical layer ...................................... 9  
      3.1.2 Link layer ......................................... 9  
      3.1.3 Network layer ..................................... 9  
      3.1.4 Transport layer ................................... 9  
      3.1.5 Application layer ................................ 10  
      3.1.6 Network topology ................................ 10  
   3.2 Network performance .................................... 10  
      3.2.1 Availability ....................................... 11  
      3.2.2 Response time and delay ......................... 11  
      3.2.3 Number of hops ................................... 11  
      3.2.4 Throughput and bandwidth ....................... 12  
      3.2.5 Available port numbers ......................... 12  
      3.2.6 Packet loss rate ................................ 12  
      3.2.7 Allowed packet sizes ............................. 12  
      3.2.8 PGM range ....................................... 13  
   3.3 Discussion and conclusion ................................. 13  
      3.3.1 Summary .......................................... 15  

4 Measurement methods and tools ................................ 16  
   4.1 Ping ..................................................... 16
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1</td>
<td>How it works</td>
<td>17</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Drawbacks</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Traceroute</td>
<td>17</td>
</tr>
<tr>
<td>4.2.1</td>
<td>How it works</td>
<td>17</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Drawbacks</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Port scanning</td>
<td>18</td>
</tr>
<tr>
<td>4.3.1</td>
<td>How it works</td>
<td>19</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Drawbacks</td>
<td>19</td>
</tr>
<tr>
<td>4.4</td>
<td>Bandwidth measurement</td>
<td>19</td>
</tr>
<tr>
<td>4.4.1</td>
<td>How it works</td>
<td>19</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Drawbacks</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>NetworkPerf - the measurement tool</td>
<td>21</td>
</tr>
<tr>
<td>5.1</td>
<td>Development</td>
<td>21</td>
</tr>
<tr>
<td>5.2</td>
<td>Design</td>
<td>22</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Implementation of measurement methods</td>
<td>22</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Graphical design</td>
<td>22</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Class architecture</td>
<td>24</td>
</tr>
<tr>
<td>5.3</td>
<td>Implementation</td>
<td>25</td>
</tr>
<tr>
<td>5.3.1</td>
<td>The Test class and its derived classes</td>
<td>25</td>
</tr>
<tr>
<td>5.3.2</td>
<td>The Settings class and its derived classes</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Use cases</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>Tests and evaluation</td>
<td>33</td>
</tr>
<tr>
<td>6.1</td>
<td>Verification</td>
<td>33</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Test set-up</td>
<td>33</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Ping</td>
<td>34</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Trace</td>
<td>34</td>
</tr>
<tr>
<td>6.1.4</td>
<td>PacketSize</td>
<td>38</td>
</tr>
<tr>
<td>6.1.5</td>
<td>PortScan</td>
<td>39</td>
</tr>
<tr>
<td>6.1.6</td>
<td>Bandwidth</td>
<td>40</td>
</tr>
<tr>
<td>6.2</td>
<td>Conclusion</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions and future work</td>
<td>42</td>
</tr>
<tr>
<td>7.1</td>
<td>Conclusions</td>
<td>42</td>
</tr>
<tr>
<td>7.2</td>
<td>Future work</td>
<td>43</td>
</tr>
<tr>
<td>7.3</td>
<td>Literature study</td>
<td>44</td>
</tr>
<tr>
<td>7.4</td>
<td>Software development</td>
<td>44</td>
</tr>
<tr>
<td>7.5</td>
<td>The use of scrum</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>Appendices</td>
<td>49</td>
</tr>
<tr>
<td>A</td>
<td>Scrum</td>
<td>49</td>
</tr>
</tbody>
</table>
List of Figures

2.1 An example of the operators view of the VTS system.

2.2 Schematic overview of a VTS network

3.1 Internet Protocol layer architecture

5.1 Screenshot from the NetworkPerf client.

5.2 Screenshot from the NetworkPerf server.

5.3 Main classes in NetworkPerf

5.4 The Test class and its derived classes

5.5 The Settings class and its derived classes

5.6 An example of a simple VTS network with NetworkPerf

5.7 An example of a more complex VTS network with NetworkPerf

6.1 Ping results
Chapter 1

Introduction

In this chapter, the background to this master’s thesis is presented. Initially an introduction to the involved companies, Attentec and Saab Transpondertech, is given and thereafter follows a general description of the situation that brings about this master’s thesis.

1.1 Background

This master’s thesis is based on Saab Transpondertech’s need for a tool to aid them when troubleshooting Vessel Traffic Service installations, where network problems could be the cause of the issue.

Attentec and Saab Transpondertech

This thesis was performed at Attentec as a consulting assignment with Saab Transpondertech as client.

Attentec is an IT consulting company founded in 2005 in Linköping targeting the market of advanced and technical software development [2]. This master’s thesis is performed at Attentec with support from consultants at the company.

Saab Transpondertech is a company established in 1996 and based in Linköping. The company primarily works with navigation equipment for shipping. Among other things Saab Transpondertech develop and manufactures a system for Vessel Traffic Service, VTS. See section 2.2 [3]
1.2 Problem description

As in most technical systems, sometimes problems arise in the VTS system. When dealing with problems in the system it is of great importance for Transpondertech to quickly determine whether the problem is caused by an error in their own equipment or in the third party controlled network. Today there is no efficient approach to determine if an adverse change in network status has occurred, and thereby may be the cause of problems. A desirable method to rule out network connection problems would be to have one integrated tool that can be used to analyze the connection.

Since there is not always possible to make an exact specification of what network performance is needed, it is not always good enough to be able to compare the current status of a network connection with such specification. Instead, there should be a possibility to compare the current status of the connection to earlier measurements, which means that the measurement tool somehow should have the capability of saving the result of a performed measurement for later comparison.

Problem formulation

From the previously given problem description, the following concrete problems can be formulated:

- What network connections properties could be interesting to measure to analyze the functionality of the connection?
- What existing methods are there to perform network reliability and performance measurements in an IP network?
- What conclusions can be made concerning the function of a network from performance measurements in the network?

1.3 Purpose

To give answers to the problems formulated in section 1.2, the purpose of this master’s thesis is

- To find measureable properties of a network connection that best reflect the existence of problems with the network,
- To find or develop methods for measuring these chosen properties, and
- To implement these chosen methods in software.

The software developed should be easy to integrate into Transpondertech’s existing software.
1.4 Scope

The servers and clients in the VTS system constructed by Transpondertech are built upon Microsoft Windows and are communicating using the TCP/IP network protocol stack. Therefore, this thesis is delimited to networks using the IP protocol and software created during the writing of the thesis will be developed to run under Windows.

1.5 Method

To fulfill the first purpose of the thesis the first step is to perform a literature study. The second step is to find methods for measuring the network properties that are found in the first step, also by studying literature. If no methods are to be found, methods should be developed. The third step is to implement the measuring methods in software and the final step is to perform tests to ensure that the implemented methods accurately measure the target properties.

Scrum

The project of writing this master's thesis is carried out using scrum. Scrum is a framework for agile software development with relative short iterations called sprints [20]. As it is an agile framework, one of the main concepts of scrum is that the planning is performed when needed and not for the whole project before it starts. During the work with this thesis, a sprint length of three weeks will be used. Before every sprint a short planning session will take place, where the work to be carried out during the sprint will be planned. Scrum is intended to be used by teams of five to nine people. Since writing a master thesis is a solo project, it would not be possible to use all parts of scrum to its full extent.

Scrum is described a little bit more in detail in Appendix A.

1.6 Structure of the report

This report consists of eight chapters and two appendices. This first chapter describes the aim of the thesis together with background information and other information about the thesis work. Chapter two brings up some information about the AIS and VTS systems.

Since network terminology and network performance are not areas of common knowledge chapter three is intended to give some fact in these areas. The fourth chapter brings up some common techniques for network performance measurement and the tools most commonly used for measurement.
CHAPTER 1. INTRODUCTION

The development of NetworkPerf is described in chapter five together with details about the implementation. Both chapters six and seven describe how evaluation has been done during this master thesis in different aspects. The test and evaluation of the developed tool are explained in chapter six while chapter seven deals with the evaluation of the working method.

Chapter eight deals with conclusions and how the work with NetworkPerf can be continued. Finally, appendix A gives a brief description of scrum.

This report is typeset using LyX [34].
Chapter 2

Automatic Identification System and Vessel Traffic Service

To fully understand the problem aimed by this master’s thesis, it is important to understand how the Vessel Traffic Service, VTS, system is built up. Automatic Identification System, AIS, is an important part of VTS and is therefore explained before VTS in this chapter.

2.1 Automatic Identification System

The Automatic Identification System, AIS, is a system for identifying naval vessels and for following their movements from land based surveillance centers as well as from other vessels. The AIS system consists of transponders and receivers. [29]

The transponders send data through two VHF radio channels in small packets at regular intervals. With an interval between two to ten seconds, the transponder on the vessel sends information about the vessel’s identity, position, course, speed and maneuver. The length of the intervals depends on the speed and maneuver of the vessel. The information about the vessel’s cargo, size and destination are transmitted with longer intervals.

The AIS transponder on a vessel is usually connected to the ship’s navigation system to get information about speed, course and position. The navigation system usually gets its data from a GPS unit. There is in the normal case no need for an operator to control the system it is working autonomously.

An AIS-receiver can have a simple text display that shows the nearest vessels and their location. The receiver can also be connected to the vessel’s
navigation system and thereby allow a graphical presentation of the received data. For instance, the AIS information can be shown on a chart plotter or together with radar information.

There are international rules about what vessels that must use AIS transponders. Today most larger vessels in both domestic and international traffic are equipped with AIS equipment.

2.2 Vessel Traffic Service

The information from the AIS transponders are received by both vessels and receivers ashore. From these receivers the information is passed along to VTS control centers [30], [16]. The purpose of these control centers is to monitor and aid the water born traffic in the control center’s respective surveillance zones. The control centers also receive information from radar stations and surveillance cameras. The operators in the control centers can communicate with the vessels using VHF radio\(^1\). Some examples of what kind of information the VTS operator can get from the VTS system can be seen in figure 2.1. In the figure can be seen a sea chart with symbols showing the location of ships in the area, information about the marked ship (in this case Shetland Cement), a list of alarms, a chart following a ship and a live feed from a camera.

The receivers ashore are connected to nearby servers that convert the signals so they can be transferred over TCP/IP. These servers are in turn connected to the VTS control center via an external network, leased from a network operator that happens to have a connection to the sometimes very distant places where the receivers are located. These distant places often are located on islands in outer archipelago, the connections are often very slow and unreliable. For instance, both short-wave links and old-fashioned modem connections are used.

In addition the network between the converting server and the control center is controlled by a third party network operator other than Transpondertech. Therefore, changes in the network can occur unknown to Transpondertech and Transpondertech does not always know the network topology. Figure 2.2 schematically describes how a VTS network is built up. In the figure, the cloud marked “Unknown network” is the part of the network out of Transpondertech’s control. On the left side of the cloud the control center is represented by one client computer and one server. On the right side of the cloud there is one remote server connected to data receivers. Only one remote server is represented in the figure for purposes of illustration, but VTS networks typically have more than one.

\(^1\)VHF, Very High Frequency, radio communication using frequencies between 156 and 174 MHz [28].
CHAPTER 2. AUTOMATIC IDENTIFICATION SYSTEM AND VESSEL TRAFFIC SERVICE

Figure 2.1: An example of the operators view of the VTS system.

Figure 2.2: Schematic overview of a VTS network
Chapter 3

TCP/IP networks and performance measurement

This chapter describes the TCP/IP protocol stack and then gives an introduction to network performance.

3.1 Network terminology

The Internet Protocol suite is built on a five-layer architecture, see figure 3.1. These five layers are, from the top down, application layer, transport layer, network layer, link layer, and physical layer. Each layer uses the services of the underlying layer to fulfill its own responsibilities.

Figure 3.1: Internet Protocol layer architecture
3.1.1 Physical layer

The physical layer is the hardware infrastructure; network cables, switches, routers, etcetera. The physical layer is responsible for transferring each individual bit from one node to another.

3.1.2 Link layer

The link layer takes care of moving whole frames from one node to the next. Ethernet and wireless network standards from the 802.11 group are two common link layer protocols, although they both also define standards for wiring and signaling on the physical layer. Addressing on the link layer is achieved by using MAC addresses.

3.1.3 Network layer

The network layer is where the internet protocol, IP, resides. The network layer is responsible for transferring packets from source node to destination node. On the network layer, the addressing uses IP addresses.

The network layer uses the services of the link layer to move packets from one node to the next in the path to the destination node. Different link layer connections can handle packets of different sizes. Sometimes the network layer packet is larger than the link layer can handle, and is therefore split into two or more smaller packets before they are handed over to the link layer protocol. Each small packet is then handled as its own packet on its way to the destination. The packets are not reassembled until they reach the destination.

This procedure is called fragmentation and is sometimes unwanted. To avoid situations where network layer packets are fragmented even though they need to be intact when they reach the destination there is a flag to set in the IP header, the “Don't fragment” flag. When this flag is set for a packet, the packet is not fragmented. Instead, if the packet is too big to be forwarded, an error message is sent back to the originating node of the packet.

3.1.4 Transport layer

TCP and UDP are the two most common transport layer protocols. Both protocols handle process-to-process communication, that is, the communication from the originating process on one host to the target process on the target host. TCP prioritizes quality before transfer speed and guarantees
that the data sent will receive the target. UDP on the other hand is opti-
mized for data transfer where high throughput is most important and it
does not really matter if a packet every now and then is lost on the way.

Another transport layer protocol used by Transpondertech is PGM, Prag-
matic General Multicast. PGM is a reliable transport protocol that handles
multicast data delivery from multiple sources to multiple receivers [32]. The
reliability in the protocol lies in that PGM, in contrast with other multicast
protocols, ensures that a receiver either receives all data packets it should
or detects that packets have been lost.

Port numbers

TCP and UDP use ports to determine which process on a host a received
package is sent to and from. A port basically is just a number between 1 and
65535. Every process that wants to be able to accept incoming connections
binds to a port and is said to listen to the port. Any incoming packages
for that port will be forwarded to the process. A port that has a process
bound to it is said to be open. If there is no process listening for incoming
packages for a specific port, the port is referred to as closed.

3.1.5 Application layer

The application layer is used by the software applications to encapsulate the
actual data to be transferred in a way that suits the application. Commonly
used application layer protocols include HTTP, SMTP, FTP, and SSH.

3.1.6 Network topology

The network topology describes how the nodes in the network are connected
to each other. This description can refer to either the physical or the logical
design of the network. The physical design describes how network elements
are physically connected whereas the logical design describes how nodes
are logically connected to each other on the transport layer. The physical
topology only changes when physical cables are connected or disconnected,
while the logical topology may change due to changes in routing tables,
which in turn may occur due to changes made by an administrator, high
traffic load on a link or by some other automatic mechanism.

3.2 Network performance

The term network performance can refer to many different properties of a
network. Some properties, like response time and throughput, have an obvi-
ous affect on the overall network performance, while some other properties
may not have a clear connection to the performance. This section is an attempt to sort out what properties affect network performance and why.

### 3.2.1 Availability

When it comes to network performance, the most fundamental, and perhaps even the easiest property to measure is whether the network works at all: is there a network path from the source to the destination that can carry messages? [8]

### 3.2.2 Response time and delay

There are two types of delay, round-trip delay and one-way delay.

One-way delay is most commonly mentioned as the time it takes for a bit or packet to travel through a network from one host to another. There is no way to synchronize the internal clocks of the two hosts easily, it is difficult to measure one-way delay. [17]

Round-trip delay is considered to be the total time it takes for a packet to travel from a source host to a destination host and for the answering packet to get back to the origin host [17]. But, to be able to send an answer the destination host must do some computations, and these computations can take more or less time, which of course affect the total time from when the message is sent to that the answer is received. In many cases, the time it takes for the destination host to calculate the answer would be negligible compared to the delays due to network transmission. E.g., this is the case when testing the delay with tools like the Ping command (see section 4.1).

Blum [8] mentions round-trip response time, which by his definition includes the answer computation time on the remote host.

**Delay variation**

Both one-way and round-trip delay can vary over time due to variable queuing delays. This variation is called delay variation or delay jitter. [17] The former is the preferred term, the term *jitter* has an exact meaning when it comes to electrical signals, but not when it comes to delay measurements. [13]

### 3.2.3 Number of hops

A measure of a route through a network rather than of the performance of the network is the number of hops a packet must do to pass from a certain sending node to a certain receiving node. A change in this number usually indicates some change in the network topology.
3.2.4 Throughput and bandwidth

The most intuitive definition of throughput is given by Comer. He states that throughput is the rate at which data can be sent through the network and is measured in bits per second. With today’s high-speed networks, the units kbps, Mbps or even Gbps are more commonly used [12]. Perhaps this defines possible throughput, as throughput itself is often used when talking about data actually passing a certain node. The term most commonly used for possible throughput is bandwidth.

Available bandwidth

Available bandwidth is what is left when the actual throughput is subtracted from the bandwidth. One way to measure this is to try to send data at as high rate as possible and see how much of the data that comes through [19]. The result of this measurement may differ from the calculated value, traffic from certain nodes may be prioritized.

3.2.5 Available port numbers

For which ports are communication possible from source to destination? Ports can be blocked in either one of the end nodes, or traffic on a certain port can be filtered out along the path.

3.2.6 Packet loss rate

The Internet Protocol transfers data on a best effort basis. That means that there are no guarantees that transmitted data will actually reach the target node without use of higher layer protocols that add that guarantee. Data can be dropped by intermediate routers for any of many possible reasons: the router can be so busy that it cannot handle all packets it receives, the packet can be corrupted and therefore dropped, or the packet can be dropped cause the router does not know the route to the destination.

The ratio between dropped packets and the packets that make it all the way to the destination is called the packet loss rate.

3.2.7 Allowed packet sizes

For many links, there are limitations of the size of packets that can be transferred over the link. Since these maximum allowed sizes of packets could vary from link to link there must be a way to handle these differences in allowed packet size.
A smaller packet can always be sent on a link that allows bigger packets. However, it is not that easy when it comes to a packet that needs to pass through a link that has a maximum allowed packet size that is smaller than the packet itself. In this case, one out of two things could happen; the packet can be dropped or the packet can be split into several smaller packets. In some cases, the sender does not want the sent packets to be split on their way to the destination, and therefore sets the “Don’t Fragment” flag. If a packet with that flag set reaches a node where the outgoing link does not allow big packets, the packet is dropped and an ICMP error message should be returned to the sender.

### 3.2.8 PGM range

Some of Transpondertech’s applications use PGM, Pragmatic General Multicast, see section [3.1.4](#). Therefore, it is of interest to know if PGM messages sent from a certain source host reach a certain destination host. How far away from the sending host the PGM messages reach is hereby called the PGM range.

### 3.3 Discussion and conclusion

When discussing the performance of a network connection between two hosts the first question to ask should be whether there is a connection at all. Therefore, the availability of the connection should be the first thing to measure.

Low round-trip delay is important for the transfer speed when it comes to TCP communication. Since all data needs to be acknowledged by the receiver and the transmitter can have only a limited amount of outstanding data, a high delay can distinctly lower the transfer speed. When it comes to UDP traffic, the situation is different. The absence of acknowledgements makes the delay have none or little impact on the transfer speed. High delays can of course be bad, for example, when transmitting real-time data that has to reach its destination within a certain time, but it does not affect the transfer speed, as is the case with TCP connections.

UDP is often used for streamed data, like audio or video streams, and therefore delay variation becomes more important. An unusually long delay in the middle of a UDP stream may cause the playback of the transferred stream to stutter. For TCP traffic one occasional delayed packet does not make that big difference to the total transfer performance. Based on this reasoning it is important to measure both (round-trip) delay and delay variation for a network connection to be able to make conclusions about its overall performance.
The number of hops a packet has to traverse from sending to receiving node is generally not of big importance to the overall performance of the network, but keeping track of the number of hops between different nodes can help detect changes in the network topology. It is also of interest which intermediate hosts the packet passes on its way from sender to receiver, since changes here may reflect changes in the network topology.

While delay tells you something about how quick the first packet sent from a host to another reaches the destination, the available bandwidth tells you how many packets can be transferred per time unit. Quite reasonably, the available bandwidth therefore is also an important measure when talking about network performance.

When communicating using TCP or UDP, port numbers is important. A server provides each service on a certain port number and if the clients cannot establish contact to that specific port, they cannot use the service. Consequently, to be able to use a service on a certain port on a certain host, it is important to know that nothing on the way to that host blocks traffic for that port.

When sending data from one host to another, the Internet protocol does not guarantee packet delivery. Packets can be dropped due to overloaded routers, routing errors or since a low quality connection introduces bit errors to the packets. How many packets of those sent from one node to another that are dropped along the path between sending and receiving nodes can be calculated based on information available at the end nodes, but not why the packets have been dropped. The information about how many packets that have been dropped by a certain router and why these packets have been dropped is easy to get if you are allowed to query the router itself.

Since NetworkPerf is intended to be used in networks that are not controlled by the user of the tool, it would most likely not be possible to query the routers in the network. Hence the packet loss rate and the reason why dropping packets is not measured by NetworkPerf.

In some cases, it is wanted to be able to send packets of a certain size, and ensure that the packets are not fragmented. To see if this is possible it would be good to be able to measure the largest allowed packet size along the network path between two hosts.

The range for PGM, Pragmatic General Multicast, is most likely something that is not included in the general meaning of performance. Nevertheless, since PGM is used by Transpondertech in some of their systems, they have specifically requested the possibility to use NetworkPerf to see if PGM packets sent from one host reach another host.
3.3.1 Summary

The above discussion ends in a number of properties that is of interest to measure. These properties are:

- availability
- round-trip delay,
- delay variation,
- number of hops,
- intermediate hosts,
- available bandwidth,
- available port numbers,
- maximum allowed packet size, and
- pgm range.
Chapter 4

Measurement methods and tools

This chapter presents how the performance measurements mentioned in the preceding chapter can be done and which tools are commonly used.

Ping and Traceroute are two very commonly used tools for network troubleshooting \cite{11, 14}. Availability and round-trip delay are measured by Ping, and Traceroute is used to determine the number of hops and intermediate hosts between two hosts in a network.

Nmap is a port scanning tool included in most Linux distributions and is considered as the de facto standard for port scanning, as is the case with TTCP when it comes to bandwidth measurement \cite{4}.

4.1 Ping

Ping is the name of a software utility originally written by Mike Muuss \cite{25, 33}. Over the years it has become the term for the method of using the echo request and echo reply messages of the Internet Control Message Protocol, ICMP. As stated in RFC 792, ICMP is an integrated part of IP and must be implemented by every IP module \cite{26}. Therefore, it is a good idea to use echo request and echo reply messages to check if the remote host is reachable, that is, there is a network connection from the local host to the remote host \cite{14}.

In addition to getting to know the pure existence of the remote host and the availability of the network path between the local host and the remote host, ping can be used to calculate the round-trip delay to the remote host.

Mike Muuss original ping software is released as open source, which has resulted in the software being ported to almost every operating system.
Ping is used to measure round-trip delay. By doing multiple measurements ping can also be used to measure delay variation.

4.1.1 How it works

The principle behind the ping software is simple. The software transmits an ICMP echo request message to a remote host. The host receiving the request is then supposed to answer by transmitting an echo response message back to the sender. The echo request message can include an optional payload, which the answering host has to return without changes. By changing size of the payload, it is possible to send ping messages of different sizes.

If the host transmitting the request gets an answer, it knows that the network connection is working. Since the requesting host knows when it sent the request and when it got the reply, it can calculate the round-trip time to the remote host.

4.1.2 Drawbacks

Ping can only be used to measure the round-trip time, not the one-way delay. This limitation comes from the fact that the host sending the ICMP echo request message only knows the total time from sending the request until the answer is received, not the time the request takes to reach the destination or the answer takes to return. Since there is no guarantee that the delays are the same in both directions, the one-way delay cannot be calculated using ping.

4.2 Traceroute

Traceroute is an IP network tool originally created by Van Jacobson in 1987 [33]. The tool is used to find which routers a packet must traverse to get to a certain destination host and thereby the number of hops between two hosts.

4.2.1 How it works

The idea of Traceroute is quite simple and is based upon the fact that all routers and hosts using the IP protocol must support ICMP [9, 7]. The host from which the query is performed starts by sending a packet to the destination host, but sets the TTL property of the packet to 1. Then, the first router reached by the packet discards it and return an ICMP time exceeded message to the origin host. From the returning ICMP message, the origin host can read the IP address of the first router.
sends a new packet, this time with the TTL property set to 2, and gets an ICMP time exceeded message from the second router on the path to the destination host. This way the host continues to send packets until it gets an answer from the destination host itself.

To prevent the host from continuing sending packets if for some reason the destination host do not answer, the Traceroute program most often stops after sending a packet with TTL set to 30 hops.

The probing packets sent can be of different kinds. When the Traceroute program was first created, it was using ICMP packets. However, since some router vendors had misinterpreted the requirement of not answering to an ICMP error message by sending another ICMP error message, many routers did not reply to the ICMP echo request message, which is not an error message. Therefore the creator of the Traceroute program for Unix systems used UDP packets instead [10].

In Windows systems, the Traceroute tool uses ICMP echo request messages whereas Linux and Unix systems traceroute tool can be configured to use ICMP, UDP or TCP packets. The default configuration of Linux’ traceroute uses UDP packets.

4.2.2 Drawbacks

The administrator of an IP router may choose to let the router silently drop ICMP echo requests without sending an echo reply, although it does not follow the IP standard. Because of this there could be some situations where a UDP version of traceroute is needed [7]. This is not implemented in NetworkPerf.

4.3 Port scanning

The purpose of port scanning is to determine the state of ports on one or many host(s). In particular, port scanning is often used to determine which ports that accepts incoming connections. By doing a port scan, it is also possible to see that no firewall along the path from the scanning host to the scanned host is blocking packets heading for a certain port. Port scanning can also be used to gather information about the scanned host, but since this work is aiming at gathering information about the network connection rather than about a certain host this is not covered in this report and not implemented in the NetworkPerf software [22].

1 The Traceroute software on Microsoft Windows systems is named Tracert.
4.3.1 How it works

The most basic way to try if a port is open is to try to establish a connection to that port. If it succeeds, the port is open. If it does not, the port may be closed, in some other way blocked, or filtered. This method is called TCP connect scan. Other methods for scanning port status are TCP SYN scan and UDP scan. [22]

4.3.2 Drawbacks

The biggest drawback of the TCP connect scan method compared to the others is that it is quite slow. If the scan is done for the purpose of gaining unauthorized access a system, then the method also has the drawback that it can easily be detected from the target system.

One problem with all port scanning techniques is that most network equipment uses a filter to protect its ports. This filter silently drops all incoming packets to ports not explicitly opened. This causes all port scanning techniques to be quite inexact; the absence of an answer could indicate that the port is open or closed depending on the type of scan technique used, or just filtered. This problem is most severe when the scan method uses a returning packet as a sign that the port is closed. In these cases, the absence of a returning packet indicates that the port is open or filtered, which are two very different things. For the TCP connect scan method the problem is of less importance since the absence of an answer packet indicates that the port is either closed or filtered, which is roughly equivalent.

4.4 Bandwidth measurement

TTCP, Test TCP, is a TCP bandwidth measurement tool developed by Mike Muuss that can be used to measure the throughput from one host to another. TTCP operates on the TCP layer, meaning that the result of the measurements done with the tool tells how fast data can be sent over TCP over a given connection. The actual transfer rate of the links in the connection will always be higher than the result reported by TTCP, since there is some overhead added in each protocol (TCP, IP and network layer protocol). This can be seen as something bad, but for all applications where TCP is used this measurement gives the most interesting values. [31]

4.4.1 How it works

TTCP consists of a server part and a client part. A measurement width TTCP is done by starting the server part on one host and then run the client part on another host. The client then sends a certain amount of data
to the server to see how much time it takes. When the data transmission is
done, both the server and the client know what time it took and how much
data that was transmitted and thereby can calculate the transfer speed, that
is, the bandwidth used.

4.4.2 Drawbacks

One possible drawback by using TTCP for measuring the available band-
width is that it only measures the throughput with TCP packets. If, for
some reason, TCP packets are prioritized different than other protocols’
packets, the test result will not be correct for network traffic using other
protocols. For TCP traffic though, the results will be correct.
Chapter 5

NetworkPerf - the measurement tool

NetworkPerf is the name of the tool that has been developed for this master’s thesis. The tool is intended as a base for future development and integration into other software, and not for direct use in its current condition. This chapter therefore focuses on how the software is designed and works, mostly from a programmer’s point of view. The chapter also deals with how the measurement methods from the previous chapter is implemented in NetworkPerf.

Nevertheless, the tool has a graphical user interface, which also is described in this chapter, and can be used to investigate the performance of a network.

5.1 Development

NetworkPerf has been developed in C# using .NET 3.5\footnote{1} and Microsoft Visual Studio 2008\footnote{2}. The biggest reason for choosing the C# language is that Transpondertech uses it for developing their software and, as stated in section 1.3, this tool is intended to be integrated into their software in the future. This integration can be easier if the tool is written in the same language as Transpondertechs existing software. Microsoft Visual Studio is the natural choice of development environment when developing in C#.

\footnote{1}{Dotnet, or as Microsoft prefers to write it, .NET, is a software framework including a large library of code solving many common programming problems.}

\footnote{2}{Microsoft is kind enough to provide students with their development tools free of charge. For this master’s thesis such a free student licence has been used.}
5.2 Design

NetworkPerf was developed using object-oriented programming. Both the composite and the factory method design patterns have been used [15].

NetworkPerf is built as a client-server system. The role of the server(s) is to perform the actual measurements, while the client is only used for the user interaction. Every measurement is performed using instances of two different classes. The first class is the class that performs the actual measurements and the second class is a setting class that is used to initialize the measurements.

5.2.1 Implementation of measurement methods

Ping is implemented in NetworkPerf the same way as in the original Ping software. The tool sends an ICMP echo request message to the host to which the round-trip time is to be calculated and then waits for the echo response message.

The traceroute test is implemented using ICMP echo request packets to probe for the path to the destination host.

NetworkPerf includes the most basic type of port scanning, the TCP connect scan method. The tool simply tries to open a connection to each port that is the target of the scanning, and if the connection attempt is successful, the port is open. If NetworkPerf does not succeed to connect to a port, the port is considered not open, that is, closed or filtered.

TTC is implemented in NetworkPerf using PCATTCP, which is a PCAUSA’s port of TTC to windows sockets [18]. The client part of PCATTCP is written in C# and is incorporated into NetworkPerf by using some of the classes from PCATTCP in NetworkPerf. The server part is only included as an external executable file that can be run by NetworkPerf when needed. See also §5.3.1

5.2.2 Graphical design

The graphical design of NetworkPerf is simple and straightforward. The look of the windows is the standard look for Windows applications created in Visual Studio 2008 and the input fields in the applications is the very least to interact with the applications. To start with, in the NetworkPerf client, see figure 5.1, the text field labeled “Server IP” is used to specify what NetworkPerf server to connect to. The “Connect” button establishes the connection and the “Run Test” button runs the test specified by the Settings.xml file in the same directory as the application itself. The NetworkPerf server,
see figure 5.2 has only one button, the “Start server” button. This button starts the server service, enabling NetworkPerf clients to connect to the server.

Figure 5.1: Screenshot from the NetworkPerf client.

Figure 5.2: Screenshot from the NetworkPerf server.
5.2.3 Class architecture

NetworkPerf is built around the Test class and its derived classes. Each
derived class, except the CompositeTest class, represents one kind of mea-
surement that can be performed by NetworkPerf. The CompositeTest class
is part of the composite design pattern implementation and cannot perform
any tests on its own; instead, it is used to hold multiple instances of the
other Test classes. The Test class itself is an abstract class that defines the
structure of all its derived classes.

Each class derived from the Test class contains an instance of a class derived
from the Settings class. This class keeps track of all settings related to
the test class. For example, an instance of the PingTest class contains an
instance of the PingSettings class, an instance of the PortScanTest class
an instance of the PortScanSettings and so on. The Settings class and its
derived classes are also used as parts of the factory method design pattern
implementation and are used to instantiate objects of the classes derived
from the Test class.

Some more details about the design of each test and settings class is pre-
sented along with how the classes are implemented in section 5.3.

Server and client

NetworkPerf consists of two applications, the NetworkPerf client and Nwp-
Server. The central classes in NetworkPerf, except the test classes and
settings classes, can be seen in figure 5.3 The main class in NwpServer
is the Program class which show the application window and lets the user
interact with the application. It also instantiates the NwpServer class which
is the core of the server application. The ServerStarter class is used to make
the instance of the NwpServer class accessible via remoting [23].

The NetworkPerf client application also has a Program class that handles
the showing of the application window, but it is not the same class, just
another class with the same name. The core of the client application is
made up of the NwpClient class. The NwpClient class has functions to load
an xml file with settings and to save the measurement result to another xml
file.
The client application communicates with the server application using Remoting, which is a Microsoft technology to remotely instantiate objects and call their methods. Remoting does not work very well through firewalls or NATs. However, Transpondertech already uses Remoting in their software and their system does not need to communicate through firewalls or NATs, so this has not been considered an issue.

5.3 Implementation

5.3.1 The Test class and its derived classes

As mentioned before, the Test class is the base class of all test classes (figure 5.4). The Test class family includes Test, CompositeTest, PingTest, TraceTest, BandwidthTest and PortScanTest. The Test class itself is an abstract class that provides an interface for the other test classes.

The BandwidthTest class

The BandWidthTest class uses functionality from the PCAUSA TTCP software [18]. As described in section 4.4, TTCP consist of a server part and a client part. The client is available in a version written in C# and dot net, released as open source. The BandWidthTest class uses an instance of the TtcpClient class from the dotnet version of TTCP, which takes part of the client part of TTCP. As remote TTCP server, the binary from the PCA TTCP package is used.

The PacketSizeTest class

The purpose of the PacketSizeTest class is to determine the maximum allowed packet size that is allowed along the path to a certain destination.
Figure 5.4: The Test class and its derived classes
host. To detect what this maximum packet size is, the class uses a helper class called Pinger. The Pinger class has one function, Ping(int buffersize) that pings the destination host with a packet of buffersize bytes. The PacketSizeTest class uses an interval halving method to minimize the number of pings that have to be sent.

The PingTest class

The class built to perform ping tests is the PingTest class. The ping functionality is created using the dot net System.Net.NetworkInformation.Ping package, which needs very little programming to create the functionality of sending ICMP echo requests and receiving the answers.

The PortScanTest class

The PortScanTest class is used to perform port scans. The class uses a helper class called Port to keep track of every port that should be scanned and does the actual scanning for each port. As for the rest, the PortScanTest class is quite short and self-explanatory.

The RemoteServiceStarter class

The RemoteServiceStarter class is a class derived from the Test class, but it does not represent a measurement that can be done. Instead, the class is used to start external software on the remote host. This is used by the bandwidth measurement test, which needs a measurement server to be run on the remote host to perform the measurement. The reason for the RemoteServiceStarter class to have the Test class as a parent is that this way the RemoteServiceStarter class can be remotely run the same way as the other classes derived from the Test class, and no separate handling of this feature is needed.

The RemoteServiceStarter class is not intended to be initiated from the Settings.xml file as the other classes derived from the Test class, although it is possible but is thought of as an internal class. As such, the class can be used by other classes than the BandWidthTest class.

The TraceTest class

The TraceTest class performs the traceroute test. Like the PingTest class, the TraceTest class uses the dot net System.Net.NetworkInformation.Ping package to send ICMP echo requests and receive the answers.
5.3.2 The Settings class and its derived classes

All classes derived from the abstract Settings class (see figure 5.5), except the CompositeSettings class, are used to keep track of the settings for a certain test. As the Settings classes also are parts of the factory method design pattern they are used to instantiate the Test classes. To create a set of tests there are a few steps to perform:

- Create a CompositeSettings object
  - Fill the CompositeSettings object with other objects derived from the Settings class, each representing the settings for the tests wanted
  - Invoke the createTest method on the CompositeSettings object. This will in turn invoke the same method for all its child objects and will return a CompositeTest object filled with object derived from the Test class.

There is one setting that is common to all kind of tests, the IP address of the remote host. All tests in one way or another measure the communication to a specific host, which is why the IP address of a remote host is needed for all tests. The IP address is therefore the only setting specified in the Settings class itself.

The BandWidthSettings class

The BandWidthSettings class keeps track of the settings for the available bandwidth test. The settings that are imported from the Settings.xml file are the buffer size, the number of buffers to be transmitted, the remote port to connect to, the IP address of the remote host (receiver) to connect to and the IP address of the host with whom to perform the test.

There are also some settings that cannot be changed from the Settings.xml file in the BandWidthSettings class, these settings are the SocketOptionBufferSize and SocketOptionNoDelay settings. The reason why these two options cannot be changed by the user is that is was first not thought necessary. A better solution would probably be to let the user change the settings via the Settings.xml file, but have default values in the BandWidthSettings class that is used when the user does not set these settings.

There are two reasons why these settings are kept in the BandWidthSettings class, in contrast to the BandWidthTest class, although they can’t be changed from user input. The first reason is that it is good programmer’s practice to keep all similar data in the same place, and the other is that it is now easy to change the behavior of the application so that in a future version, it has the feature of changing also these two settings from user input.
Figure 5.5: The Settings class and its derived classes

The PacketSizeSettings class

There are only three parameters that need to be set for the PacketSizeTest class. The MinPacketSize and MaxPacketSize settings tell the interval half-
ing method what initial values to use, and the NumberOfRetries setting is used to the test how many times to retry pinging with a certain packet size with no answer before quitting.

**The PingSettings class**

One instance of the PingSettings class is used to keep track of the settings for each PingTest object. In addition to the IP address of the host to ping, the PingSettings class handles information about if fragmentation should be allowed on outgoing ICMP echo request packages, how big the packages should be (or more exactly how much data load they should carry), how many requests should be sent and how many hops the packets should be allowed.

**The PortScanSettings class**

The port scanning test needs a list of the ports to be scanned, this list is handled by the PortScanSettings class. For the possibility of future improvements, there is also an option to add information about what kind of scan to perform with each port.

**The RemoteServerStarterSettings class**

The RemoteServerStarterSettings class has three settings, the path and filename of the external program to be started and the arguments, if any, to be passed along to the program.

**The TraceSettings class**

For the TraceTest there are no settings to handle except the IP address of the host to trace to, and therefore the TraceSettings class just implements the abstract Settings class.

### 5.4 Use cases

As mentioned in section 5.2.3, NetworkPerf consists of both a server application and a client application. The server application is intended to be installed and run on every server in the VTS network. The server application is the part doing the actual measurement and acts as the counterpart for some measurements that needs the remote end to interact with the local end for the measurement to be performed. The client application is used to control the measurements and to save the result to an XML file.
An example of how NetworkPerf can be used is given in figure 5.6. The computer to the left is an ordinary desktop computer in the VTS control center, the left server is one of the servers in the VTS control center and the right-most server is one of the servers connected to external antennas or receivers. The desktop computer from which the test is initiated does not have to be located in the control center; the important thing is that it is connected to the same network as the servers between which to perform measurements. Since the connection between the two servers in the figure typically is out of Transpondertech’s control, this is a typical situation where NetworkPerf might be useful. To make a measurement of the network between the two servers the user starts the NetworkPerf client at the desktop computer, and connects to one of the servers. The server then performs the measurements and returns the result to the client.

Figure 5.6: An example of a simple VTS network with NetworkPerf

A more complex scenario can be seen in figure 5.7, where many client computers are connected to the VTS control center network and many remote servers are connected to the control center through the cloud, the unknown network. In this case, a user at any of the desktop computers can run measurements between any two of the servers in the network. Actually, all tests except the bandwidth test can be run between one server and any other computer in the network. The bandwidth test is the only measurement that needs the target for the measurement to have the NetworkPerf server installed and running.
Figure 5.7: An example of a more complex VTS network with NetworkPerf
Chapter 6

Tests and evaluation

When creating a measurement tool it is important to verify that the tool really measures what it claims to measure, and that the measurements are correct. This chapter describes how the verification can be done, and the results for the verification performed with NetworkPerf.

6.1 Verification

To verify that each type of measurement that can be performed by NetworkPerf is performed correctly, I have done comparisons of measurements made by NetworkPerf and well-known tools publically available. The measurements available in NetworkPerf and whose correctness was verified are the Ping, Trace, PacketSize, PortScan, and Bandwidth tests.

Since the properties measured by NetworkPerf are of different types, the methods for verifying the results of the measurements also must be of different types.

6.1.1 Test set-up

For the verification test, three internet connected computers with the NetworkPerf server were used. One of the computers was connected to the internet at Linköping University and the two others via Bredbandsbolaget at two different locations in Linköping. For an overview of the used computers, their operating systems, internet connection, and their IP addresses, see table 6.1

The NetworkPerf client was run on Computer A. The reference tools were run on different computers depending on the actual tests performed. Computers B and C was accessed using Windows remote desktop in order to start the NetworkPerf server and to run the reference tools when needed.
### 6.1.2 Ping

The Ping test in NetworkPerf was compared to the ping functionality in Microsoft Windows [5].

This test should result in a numerical value, which can change over time. Statistics is a good way to ensure that the values obtained from measurements are correct, especially when the value naturally varies over time. By doing a large number of measurements with both NetworkPerf and a reference tool, the correctness can be verified. As a reference tool for the Ping test, the ping software included in Microsoft Windows was used.

To verify the correctness of the measurements performed by the Ping test, NetworkPerf and the Microsoft Windows ping tool were both used. First, NetworkPerf was run, sending four echo requests and waiting for the answers, then ping was run, sending four requests and waiting for the answer, and so on. The round-trip time values returned by the two tools were then plotted into the same graph, see figure 6.1. In an ideal world, where the ping time does not fluctuate, both lines in the graph would be straight and on top of each other. In the real world a good result would be that both lines fluctuate about just as much, which I think is the case. Therefore, I consider the verification of the ping test successful.

### 6.1.3 Trace

The Trace test in NetworkPerf was compared to the Tracert tool available in Windows [6].

While statistics is useful for verifying the correctness of the result of the Ping and Bandwidth tests, it is not applicable when it comes to the correctness of the Trace test. This test does not result in a single numerical value that can be used in statistics. Instead the Trace test returns a list, which can be either correct or incorrect.
The results from the Trace test can change over time, but not as quickly as the measurement values from the ping test. Therefore, another approach for verifying the correctness of the measurement result is needed when it comes to the trace test. First, the test should return the same answer every
time it is run. Second, it should return the same answer as the reference tool, which is considered the correct answer. The Tracert tool included in Microsoft Windows is used as reference tool for the Trace test.

The trace test was run three times towards two different hosts. Also, the tracert tool was run the same number of times towards the same hosts. The result of the test runs can be found in tables 6.2, 6.3, 6.4, and 6.5. For both hosts, each of the three runs with both tools returned the same list of intermediary hosts, except for some small differences. The differences lie in the answer times the tools report for each host and also in the host names. The IP addresses reported for the hosts were always the same from both tools, but sometimes NetworkPerf does not detect the host name when Tracert does.

There is nothing strange in the variation of answer times. The load on the network and the answering host may vary and thereby cause the varying answer times. However, the lack of some host names is a little strange, although I would consider it a minor issue since the IP addresses always seem to be correct. Therefore, with a small reservation concerning the host names, I consider the verification of the trace test successful.

<table>
<thead>
<tr>
<th>Time</th>
<th>IP address</th>
<th>Host name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>130.236.216.1</td>
<td>gw-216-1.netlogon.liu.se</td>
</tr>
<tr>
<td>0</td>
<td>130.236.7.193</td>
<td>g-lkpnetlogon.net.liu.se</td>
</tr>
<tr>
<td>0</td>
<td>130.236.7.25</td>
<td>a-g-1.net.liu.se</td>
</tr>
<tr>
<td>4</td>
<td>130.236.9.6</td>
<td>green-a.net.liu.se</td>
</tr>
<tr>
<td>4</td>
<td>193.11.0.17</td>
<td>*</td>
</tr>
<tr>
<td>5</td>
<td>195.245.240.158</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>146.172.105.73</td>
<td>ti3001c310-ae4-0.ti.telenor.net</td>
</tr>
<tr>
<td>0</td>
<td>146.172.98.26</td>
<td>ti3010d320-ae0-0.ti.telenor.net</td>
</tr>
<tr>
<td>0</td>
<td>146.172.108.138</td>
<td>ti3053a210-x99-1.ti.telenor.net</td>
</tr>
<tr>
<td>9</td>
<td>85.231.146.112</td>
<td>c-7092e755.03-434-6c6b701.cust.bredbandsbolaget.se</td>
</tr>
</tbody>
</table>

Table 6.2: NetworkPerf trace test from 130.236.219.189 to 85.231.146.112
### Table 6.3: Traceroute from 130.236.219.189 to 85.231.146.112

<table>
<thead>
<tr>
<th>Time</th>
<th>IP address</th>
<th>Host name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10</td>
<td>130.236.216.1</td>
<td>gw-216-1.netlogon.liu.se</td>
</tr>
<tr>
<td>&lt;10</td>
<td>130.236.7.193</td>
<td>g-lkpnetlogon.net.liu.se</td>
</tr>
<tr>
<td>&lt;10</td>
<td>130.236.7.25</td>
<td>a-g-l.net.liu.se</td>
</tr>
<tr>
<td>6</td>
<td>130.236.9.6</td>
<td>green-a.net.liu.se</td>
</tr>
<tr>
<td>3</td>
<td>193.11.0.17</td>
<td>c1sth-ae0-1003.sunet.se</td>
</tr>
<tr>
<td>3</td>
<td>195.245.240.158</td>
<td>netnod-ix-ge-a-sth-4470-2.bredband.com</td>
</tr>
<tr>
<td>9</td>
<td>146.172.105.73</td>
<td>ti3001c310-ae4-0.ti.telenor.net</td>
</tr>
<tr>
<td>8</td>
<td>146.172.98.26</td>
<td>ti3010d320-ae0-0.ti.telenor.net</td>
</tr>
<tr>
<td>9</td>
<td>146.172.108.138</td>
<td>ti3053a210-xe9-1.ti.telenor.net</td>
</tr>
<tr>
<td>10</td>
<td>85.231.146.112</td>
<td>c-7092e755.03-434-6c6b701.cust.bredbandsbolaget.se</td>
</tr>
</tbody>
</table>

### Table 6.4: NetworkPerf trace test from 130.236.219.189 to 96.17.108.9

<table>
<thead>
<tr>
<th>Time</th>
<th>IP address</th>
<th>Host name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>130.236.216.1</td>
<td>gw-216-1.netlogon.liu.se</td>
</tr>
<tr>
<td>0</td>
<td>130.236.7.193</td>
<td>g-lkpnetlogon.net.liu.se</td>
</tr>
<tr>
<td>0</td>
<td>130.236.7.29</td>
<td>a-g-2.net.liu.se</td>
</tr>
<tr>
<td>3</td>
<td>130.236.9.6</td>
<td>green-a.net.liu.se</td>
</tr>
<tr>
<td>3</td>
<td>193.11.0.17</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>193.10.252.161</td>
<td>se-fre.nordu.net</td>
</tr>
<tr>
<td>13</td>
<td>193.10.68.118</td>
<td>dk-ore.nordu.net</td>
</tr>
<tr>
<td>79</td>
<td>64.214.143.137</td>
<td>64.214.143.137</td>
</tr>
<tr>
<td>106</td>
<td>67.17.193.42</td>
<td>67.17.193.42</td>
</tr>
<tr>
<td>135</td>
<td>68.86.98</td>
<td>pos-2-6-0-0-cr01.chicago.ilibone.comcast.net</td>
</tr>
<tr>
<td>163</td>
<td>68.86.250</td>
<td>pos-1-12-0-0-cr01.denver.ilibone.comcast.net</td>
</tr>
<tr>
<td>181</td>
<td>68.86.209</td>
<td>pos-0-7-0-0-cr01.seattle.ilibone.comcast.net</td>
</tr>
<tr>
<td>180</td>
<td>96.17.108.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: NetworkPerf trace test from 130.236.219.189 to 96.17.108.9
CHAPTER 6. TESTS AND EVALUATION

<table>
<thead>
<tr>
<th>Time</th>
<th>IP address</th>
<th>Host name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10</td>
<td>&lt;10</td>
<td>gw-216-1.netlogon.liu.se</td>
</tr>
<tr>
<td>&lt;10</td>
<td>&lt;10</td>
<td>g-lkpnetlogon.net.liu.se</td>
</tr>
<tr>
<td>&lt;10</td>
<td>&lt;10</td>
<td>a-g-2.net.liu.se</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>130.236.9.6 green-a.net.liu.se</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>193.11.0.17 clsth-aeo-1003.sunet.se</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>193.10.252.161 se-fre.nordu.net</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>193.10.68.118 dk-ore.nordu.net</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>64.214.143.137</td>
</tr>
<tr>
<td>105</td>
<td>106</td>
<td>67.17.193.42</td>
</tr>
<tr>
<td>134</td>
<td>134</td>
<td>68.86.86.98 pos-2-6-0-0-cr01.chicago.il.iblebone.comcast.net</td>
</tr>
<tr>
<td>163</td>
<td>163</td>
<td>68.86.85.250 pos-1-12-0-0-cr01.denver.co.iblebone.comcast.net</td>
</tr>
<tr>
<td>181</td>
<td>181</td>
<td>68.86.85.209 pos-0-7-0-0-cr01.seattle.wa.iblebone.comcast.net</td>
</tr>
<tr>
<td>180</td>
<td>181</td>
<td>96.17.108.9 a96-17-108-9.deploy.akamaitechnologies.com</td>
</tr>
</tbody>
</table>

Table 6.5: Traceroute from 130.236.219.189 to 96.17.108.9

6.1.4 PacketSize

A third type of test method is needed for the maximum packet size test. The result of the test is one single numerical value as is the case for the ping and bandwidth tests. However, this value has some similarities with the measurement results of the trace and portscan tests as well. The value does not change over time, or at least it does not fluctuate with changing load on the network connection, and the measured value is either correct or wrong.

Therefore, the correctness of the value returned by the maximum packet size test can be verified by using a reference tool to find out the correct value and then comparing the value returned by the test with the value returned by the reference tool. Once again, the ping tool that comes with Microsoft Windows is used as reference tool. The ping tool does not include functionality to automatically measure the maximum packet size allowed to a certain remote host, but this can be tested by manually sending packets, not allowed to be fragmented, of different sizes and see which packets get through.

The NetworkPerf PacketSize test was used to detect the maximum allowed packet size and thereafter this value was verified with the ping tool, by first sending a packet with the packet size detected by NetworkPerf and then a packet that is one byte bigger. If the first ping got through and the second returned an error stating that the packet needed to be fragmented, the test was considered successful.

Table 6.6 shows that for all four tested network paths, the NetworkPerf PacketSize test works as intended. All paths were tested several times with the same result.
One can argue that it is not verified that the PacketSize test works for any other packet sizes than 1472 byte. Since I have not found a network with a different maximum allowed packet size I have not had the possibility to prove this, but based on how the test is implemented I am convinced the PacketSize test works for all packet sizes.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Result</th>
<th>Successful</th>
</tr>
</thead>
<tbody>
<tr>
<td>130.236.219.189</td>
<td>192.36.125.18 (ping.sunet.se)</td>
<td>1472</td>
<td>Yes</td>
</tr>
<tr>
<td>130.236.219.189</td>
<td>85.231.147.103 (lemon.madcap.se)</td>
<td>1472</td>
<td>Yes</td>
</tr>
<tr>
<td>130.236.219.189</td>
<td>85.231.146.112 (madcap.se)</td>
<td>1472</td>
<td>Yes</td>
</tr>
<tr>
<td>85.231.147.103 (lemon.madcap.se)</td>
<td>85.231.146.112 (madcap.se)</td>
<td>1472</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 6.6: PacketSize test results

### 6.1.5 PortScan

For the NetworkPerf PortScan test, Nmap was used as a reference. To verify the PortScan test, a host with some open and some closed ports was scanned with both NetworkPerf and the reference tool. Nmap was configured to use the TCP connect scan method, which is the same method that the PortScan test uses. While NetworkPerf only reports whether a port is open and accessible or not, Nmap distinguishes between closed and filtered ports. I think this is a minor difference since connections cannot be made to neither closed nor filtered port.

The result of the scan can be found in table 6.7. As can be seen in the table, all ports reported open by NetworkPerf are also reported open by Nmap, and all ports reported not open by NetworkPerf are not reported open by Nmap. So, as far as I can see, the PortScan test in NetworkPerf works fine.

<table>
<thead>
<tr>
<th>Port</th>
<th>Nmap</th>
<th>PortScan</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>closed</td>
<td>False</td>
</tr>
<tr>
<td>22</td>
<td>filtered</td>
<td>False</td>
</tr>
<tr>
<td>25</td>
<td>filtered</td>
<td>False</td>
</tr>
<tr>
<td>80</td>
<td>open</td>
<td>True</td>
</tr>
<tr>
<td>81</td>
<td>filtered</td>
<td>False</td>
</tr>
<tr>
<td>3389</td>
<td>open</td>
<td>True</td>
</tr>
</tbody>
</table>

Table 6.7: Comparison between port scan result from Nmap and NetworkPerf’s PortScan

---

1472 byte is what is left of a standard ethernet 1500 byte frame when the IP and ICMP headers are removed.
6.1.6 Bandwidth

For the Bandwidth test in NetworkPerf, the original TTCP for Windows implementation by PCAUSA was used as reference [18]. The Bandwidth test and TTCP were both been set up to measure the available bandwidth between the same hosts. TTCP has been used with its default settings.

Tables 6.8, 6.9 and 6.10 show the result of the measurements.

The differences in the test results returned by the two tools are small and well within the variation in the results returned by TTCP. Therefore the measurements performed by the NetworkPerf Bandwidth test is considered correct.

<table>
<thead>
<tr>
<th></th>
<th>Server 1</th>
<th>Server 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time</td>
<td>2009-07-01 18:10:30</td>
<td>2009-07-01 18:12:34</td>
</tr>
<tr>
<td>Used time [s]</td>
<td>37.69</td>
<td>48.83</td>
</tr>
<tr>
<td>Transmitted data [byte]</td>
<td>16777216</td>
<td>16777216</td>
</tr>
<tr>
<td>Transfer speed [KB/s]</td>
<td>434.70</td>
<td>335.54</td>
</tr>
</tbody>
</table>

Table 6.8: TTCP bandwidth measurement result

<table>
<thead>
<tr>
<th></th>
<th>Server 1</th>
<th>Server 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used time [s]</td>
<td>40.31</td>
<td>41.50</td>
</tr>
<tr>
<td>Transmitted data [byte]</td>
<td>16777216</td>
<td>16777216</td>
</tr>
<tr>
<td>Transfer speed [KB/s]</td>
<td>406.44</td>
<td>394.82</td>
</tr>
</tbody>
</table>

Table 6.9: TTCP bandwidth measurement result

<table>
<thead>
<tr>
<th></th>
<th>Server 1</th>
<th>Server 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time</td>
<td>2009-07-01 18:29:44</td>
<td>2009-07-01 18:35:40</td>
</tr>
<tr>
<td>Used time [s]</td>
<td>37.877</td>
<td>43.415</td>
</tr>
<tr>
<td>Transmitted data [byte]</td>
<td>16777216</td>
<td>16777216</td>
</tr>
<tr>
<td>Transfer speed [KB/s]</td>
<td>442.94</td>
<td>386.44</td>
</tr>
</tbody>
</table>

Table 6.10: NetworkPerf bandwidth measurement result

6.2 Conclusion

My conclusion after doing the verification of the different NetworkPerf tests described in 6.1 is that they all work properly and perform correct measure-
ments of a network.
Chapter 7

Conclusions and future work

NetworkPerf was planned to be tested in the VTS network actually used by the Swedish Maritime Administration (Sjöfartsverket). Due to some technical problems this test could not be performed. The measurement result from this test was to be used as a basis for a discussion with Transpondertech about what use they will have using NetworkPerf.

Martin Pettersson at Transpondertech says that the tool has the potential to be useful to them, in two different ways [1]. First, the tool can be used as intended, to detect unwanted and unexpected changes in the network. Second, Martin says, the tool would probably be useful before installing a new system, to check whether the network to be used by the new VTS system has performance good enough for the application.

To better detect changes in the network, Martin thinks it would be useful to set a schedule so that the performance tests with NetworkPerf are performed on a regular basis, perhaps once every day or twelve hours. This would give better statistics to help make the analysis of the measurement results more precise.

7.1 Conclusions

In this section I summarize the answers to the questions from the problem formulation in section 1.2.

What network connection properties could be interesting to measure to analyze the functionality of the connection?

The following network properties have been found interesting to measure: availability, round-trip delay, delay variation, number of hops, intermediate hosts, available bandwidth, available port numbers, maximum allowed packet size, and pgm range.
What existing methods are there to perform measurements in an IP network?

To measure the properties availability, round-trip delay, delay variation, and number of hops, pinging is a good method. Ping is implemented as a tool in most modern operating systems and also as a test in NetworkPerf. The ping tools can also be used to detect the maximum allowed packet size. A certain test to detect the allowed packet size is built in into NetworkPerf.

The TTCP tool is a tool created to measure the available bandwidth for a certain network connection. A test with similar functionality has also been built in into NetworkPerf.

To detect which port numbers that are accessible on a remote host, a port scan should be performed. There are several port scanning methods. In NetworkPerf the TCP connect scan method has been implemented, allowing the user to detect what ports on a remote host that accept incoming TCP connections.

What conclusions can be made concerning the function of a network from measurements performed in the network?

This is a question that has not really been answered by this master’s thesis. The reason of asking the question was, when the work with the thesis started, a vague idea of that a performance analysis of a network done by a automated tool like NetworkPerf would be able to point out exactly what is causing any trouble in the network. During the work with NetworkPerf and the thesis, no algorithms for finding problem causes has been developed and this is probably a good subject for a master’s thesis on its own.

Nevertheless, some basic conclusions about the function of the network can be made from the measurements performed by NetworkPerf. The result can for instance show that there is a network connection all the way between the two servers in the measurement, that there is at least some minimum available bandwidth between the servers and that packets from one of the servers to the other need to pass a certain number of routers.

7.2 Future work

As stated in section 3.3 PGM range is of interest to Transpondertech. Because of the limited time for writing this master’s thesis, a method for measuring the PGM range has not been developed or implemented in NetworkPerf. Therefore, finding a method for measuring the PGM range and implementing the method in NetworkPerf is desired.

There are also some other possible improvements of the NetworkPerf tool:

More port scanning techniques At the moment NetworkPerf only implements one method of scanning for open ports. Many other methods...
CHAPTER 7. CONCLUSIONS AND FUTURE WORK

can be used, for instance the TCP syn scan, null scan or fin scan. Before implementing more port scanning techniques, it can be a good idea to investigate what techniques give good results in different situations and which techniques to implement.

Comparison between new and old results The advantages of using NetworkPerf would increase if the tool could be used for some sort of comparison between different measurements. This comparison would more explicitly show changes in the measured network between different measurements.

Graphical user interface for test setup The current version of NetworkPerf has no built in method to configure the tests to be performed. This is instead done by editing a configuration XML file. This could be integrated into the tool, which in some cases could make the test set-up more intuitive.

Measurement result presented in graphs The outcome of a measurement could be presented in a graph, so that it would be easier to spot changes in the network’s performance. This could be done either in a separate tool that reads the resulting XML file created by NetworkPerf, or integrated in the NetworkPerf tool itself.

7.3 Literature study

There are many book on the topic of network measurement. From the several books I have read I have come to the conclusion that there are not so many methods for measuring each network property as I first thought there would be.

I can’t say that any single book has been more important to the work with this master’s thesis than any other book, but a few books and articles have been contributing more to my understanding of the topics. Jain and Dovrolis together with Prasad, Dovrolis, Murray and Claffy gave a good analysis of the terms bandwidth and throughput in their articles [19, 27] and Comer helped me with the understanding of the ICMP protocol and its use in his book [12].

7.4 Software development

During the work with this master’s thesis and the development of NetworkPerf I have not only learnt a lot about network performance, but also the C# programming language. To get into the language I had good help from the book Programunining C# by Jesse Liberty [21]. By reading this book I have earned a basic understanding of the language, and thereafter
mostly used the MSDN Library to look up things I have needed to understand better [24].

Scrum has been a great tool to help me prioritize the development of NetworkPerf.

Early in the development process I came up with the idea of how to organize the functionality of NetworkPerf into classes. After some discussion with my supervisor Hannes Persson, I decided to go on with my initial plan for the class structure and this has turned out well. Adding new tests and new functionality to the software has worked well with the chosen class structure, which I take as a proof for that the structure is good.

7.5 The use of scrum

As mentioned in section 1.5 the project resulting in this thesis was carried out using scrum, albeit in a slightly modified version since it has been a one-man project. The use of scrum has helped me organize the work with the project, especially in the beginning of the project when it was very unclear what to do.

I have used both the product backlog and the sprint backlog and I have also had sprint plannings, but only for myself. During the sprint plannings I have planned the upcoming sprint and decided which stories from the product backlog to move to the sprint backlog.

In the beginning of the project I used the scrum framework a bit more strict than later on. For the first four sprints, a sprint planning was held and time estimates for the tasks on the scrum board was made. The sprint planning was later on in the project skipped, as it did not feel necessary to have when one person at the same time had the role of product owner, scrum master and (the only) developer. When one sprint ended, there was already a prioritized list of tasks for the next sprint. The time estimates where skipped, at least the explicit estimates, along with the sprint plannings.

Maybe the removal of time estimates was not so good, the work may have been more efficient towards the end of the project if the time estimates had been used all the way to the end of the project.

The parts of the scrum framework that have been most useful is the project backlog and the sprint backlog. In order to keep track of what has been done and what to do they have come out to be invaluable tools.
Bibliography

[1] Interview with Martin Pettersson, Saab Transpondertech, 2009-08-12.


Appendix A

Scrum

Scrum is an agile software development method based on relatively short iterations, with scrum terms named sprints. An important concept within scrum is the Product backlog or shorter, just backlog, which is a list of stories. Each story can consist of a task to be done, a feature to be implemented or something else that need to be carried out.

A.1 Artifacts

One thing that differs Scrum from more traditional software development methods is the low number of artifacts. In Scrum, only two artifacts are used, except the software produced.

Product backlog is a list of stories. Each story can be a features to be implemented, a test to be carried out, some research to be done or so one. Everyone should be allowed to add tasks to this list, but the responsibility of prioritizing the tasks and giving them time estimates is reserved fot the product owner and the team respectively, see section A.2.

Sprint backlog is a list of tasks to be performed during the current sprint. The sprint backlog is filled with tasks from the product backlog during the sprint planning, see section A.3.

A.2 Organization

Product owner The product owner is the person that gives the team its mission and represents the customer’s interests. The product owner
decides what the team should work with by making sure that the stories on the product backlog are always prioritized.

**Scrum team** The team is the people carrying out the actual development. It is also the responsibility of the team to make estimates for all the stories on the product backlog, since the people in the team are those with the best knowledge of how much time a given task will take. An appropriate size of a scrum team usually is five to nine people. Kniberg [20] however, states that it would probably work with as few as three people in each team.

**Scrum master** Within scrum a project is not managed by a project manager like in other projects. Instead, there is a role called scrum master. The scrum master has the duty of helping the team members to follow scrum and to remove obstacles so the team can work more efficiently.

### A.3 Activities

**Sprint** Each iteration or cycle is called a sprint.

**Sprint planning** Before every new sprint there should be a sprint planning meeting, when planning for the upcoming sprint is carried out. It is important to make sure that the product owner is to decide the prioritization of different stories and that the teams are making the estimates. In a sprint planning, the top stories of the product backlog are moved into the sprint backlog until the summed estimate of the stories on the sprint backlog is enough to fill the sprint. In case there is more than one scrum team, each team chooses what stories to add to their sprint backlog.

**Daily scrum** Every day during a sprint the team should have a short meeting, called daily scrum or just scrum. During the meeting, every team member should tell the team what he or she has been working on since the last scrum, what they plan to work with till the next scrum and if they have met any obstacles that need to be dealt with.

**Sprint review** In the end of each sprint there should unconditionally be a demonstration of the stories that have been finished during the sprint. The time for the demonstration should be planned already during sprint planning. How to do the demonstration has to be adjusted to fit to each story. After a story concerning performance enhancements, it might be better to show a test report from a load test than trying to show each little change in the code.

**Sprint retrospective** To enhance the work in the scrum team it is important to do evaluations. That is what sprint retrospectives are for.
Retrospectives are held as the very last step of each sprint. Defects and shortages should be fixed as quickly as possibly, so that upcoming sprints will be more efficient.
In order to detect network changes and network troubles, Saab Transpondertech needs a tool that can make network measurements.

The purpose of this thesis has been to find measurable network properties that best reflect the status of a network, to find methods to measure these properties and to implement these methods in one single tool. The resulting tool is called NetworkPerf and can measure the following network properties: availability, round-trip delay, delay variation, number of hops, intermediate hosts, available bandwidth, available ports, and maximum allowed packet size.

The thesis also presents the methods used for measuring these properties in the tool: ping, traceroute, port scanning, and bandwidth measurement.