# Monitoring of end-to-end delay and throughput in a UMTS network

M.W.J. van Es

Master thesis

# Monitoring of end-to-end delay and throughput in a UMTS network

M.W.J. van Es

**Blekinge Tekniska Högskola**
School of Engineering
Department of Telecommunication Systems
Campus Gräsvik
37179 Karlskrona
Sweden

**Vrije Universiteit**
Faculty of Sciences
Business Mathematics and Informatics
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands

# Preface

The last part of the study Business Mathematics and Informatics consists of an internship, during which the student has to work for at least six months at a company. My master thesis has been conducted at the School of Engineering at the Blekinge Institute of Technology in Karlskrona, Sweden.

I would like to thank Rob van der Mei and Sandjai Bhulai, from the Vrije Universiteit Amsterdam, for their help finding an internship abroad and their support during my stay in Karlskrona. Furthermore, I would like to thank my supervisor at the Blekinge Institute of Technology, School of Engineering, Markus Fiedler for his guidance and time to answer my questions and for his understanding, while I was pretty new in the field of telecommunications. And I would also like to thank him for the opportunity to perform my internship at the Blekinge Institute of Technology. I am also indebted to my last supervisor, Katarzyna Wac from the University of Geneva. Even with the distance she has been a great support to me, whereas she was always there to give some advice and feedback on my questions. I would also want to thank her for the great time when she was in Sweden. I also wish to thank Patrik Arlos and Lennart Isaksson for their help when I had a question during my research. Furthermore, I would like to thank Rick Duursma, he has been supportive in all areas. Last but not least, I would like to thank my friends in Karlskrona for their support, friendship and making me feel at home.

*Mascha van Es*
*Karlskrona, January 2007*

# Abstract

In recent years, wireless networks gained an enormous popularity; people are more and more interested in using these wireless networks as they, operating in the heterogeneous networking environment, support users mobility, but because of this growth the quality of the network can not always be guaranteed. Therefore, a lot of research projects have been conducted to observe how these networks behave under different loads offered to the network. Two of these projects are of great importance for this thesis: MobiHealth and PIITSA.

These two research projects focused on the users and, therefore, they researched (i.e., measured) the *end-to-end* one-way delay and throughput performance parameters observed at the application level in a heterogeneous networking environment with a UMTS network as an access network.

The purpose of this thesis is twofold; (1) to compare the two research projects and (2) to analyse the measured data, collected in these projects, and to find the method of the following research goal:

> *To develop and evaluate a method for the identification of bottleneck conditions in a heterogeneous networking environment based on the relationships between measured one-way delay and throughput performance metrics.*

The first part of this thesis we will be focused on the methodologies used in these two research projects and give a comparison. This will start with some general background information, which is useful to know before getting into details. This will be followed by the differences in the two research projects.

In the second part of this thesis, the method of the research goal will be given and illustrated, hence the relationship between one-way delay and throughput for bottleneck conditions will be given. This relationship is given by the *additional delay* for each packet, this is the extra delay on top of the expected observed one-way delay for this packet transported from sender to receiver in a network.

To identify the bottlenecks in a network we introduced an algorithm. The algorithm starts with selecting a time interval which is quite large and calculate the drift for each of the time intervals. A drift unequal to zero indicates that there is a bottleneck in the order of magnitude if the additional delay. Finally by reducing the time interval we find that the additional delay is at least equal to the time interval, this is only the case when the throughput at the receiver reaches zero and of course when the throughput at the sender is unequal to zero.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# Introduction

The purpose of this chapter is twofold; firstly, to provide the underlying reasons for the research described in this thesis (sections 1.2, 1.3, and 1.4) and secondly, to present the research questions (section 1.5) and the approach towards corresponding answers (section 1.6). The first section is devoted to the presentation of the research institute, where this research has been conducted, while the last section is devoted to the description of the thesis structure (section 1.7).

## 1.1 Blekinge Tekniska Högskola

The Högskolan i Karlskrona-Ronneby [1] was founded in 1989 and is located in Karlskrona and Ronneby, Sweden. In 2000 a new campus in Karlshamn has been opened, so the name has changed into the Blekinge Tekniska Högskola (BTH). But even if it is located on three different campuses, the university is not very big. The BTH has about 6900 students, more than 15% are from outside Sweden, and 480 employees, from which 43% are women. Besides the education, the BTH is known for its research, one third of the institute's turnover are research activities. The research is done in different fields, but the common characteristic for the research is the profile on applied IT and sustainable development of business and society. This is also found in the mission, which is to lead nationally in the profile areas: applied information technology and sustainable development of industry and society. The research and teaching groups together represent about 15 different cultures, and besides that, the university hosts a lot of international students, which at least double the amount of cultures at this relatively small university. This leads to the situation in which all of the five continents are represented at the univer-

sity. Most of the international students are studying at the School of Electrical Engineering.

The School of Engineering is the largest school at the BTH, it has about 200 employees. Its focus is on research (about 45%), education, and engineering within the areas of computer science, electrical engineering, human work science, mathematics and science, mechanical engineering, and software engineering.

The main purposes for research of the School of Engineering are:

- to generate new knowledge;
- to produce technical doctoral and licentiate degrees;
- to form a strong connection between research and undergraduate education;
- to contribute to the development of industry and society.

This MSc thesis has been conducted at the BTH School of Engineering in the time frame of February 2006-December 2006.

## 1.2 Objective of this study

In this thesis we would like to presents a methodology for measurement-based performance evaluation of heterogeneous networking environments comprising a mobile link (e.g., UMTS), supporting delivery of mobile services. Particularly, this thesis focuses on appliance of one-way delay and throughput analysis on measurement data sets in order to infer bottleneck conditions based on the relationships between these parameters. This work includes also the definition of application-specific performance thresholds for issuing performance "alarms". The assignment results in the end-to-end measurements-based performance monitoring methodology facilitating development of self-organizing mobile applications and services.

## 1.3 Rationale

Emerging high-speed wireless networks (e.g., GPRS and UMTS) providing Internet access give rise to new demanding mobile services in various application domains, including health care. An example is a tele-monitoring service, allowing for the continuous monitoring of a patient's vital signs, and generating an alarm in a health care center in case of an emergency. These types of applications pose strict constraints on the delivered Quality of Service (QoS). Particularly, an end-user QoS requirement is expressed in terms of service dependability, accuracy and speed[1] performance criteria. Furthermore, the speed performance criterion is expressed in terms of end-to-end delay, delay variation, and

[1]The time it takes to transport data from a sender to a receiver

throughput performance metrics [2].

To make sure that the strict QoS requirements are met along the service delivery, the service performance should be monitored on a continuous basis. However, in today's practice, measurements are hardly possible in operational environments.

## 1.4 Research projects as a context

The research context for this thesis has been provided by two research projects – MobiHealth and PIITSA – which resulted in the development of methodologies for measurement-based performance evaluation of heterogeneous networking environments supporting delivery of mobile services. Particularly, the methodology developed in the MobiHealth project focused on the *one-way delay* measurements per application-message, while the PIITSA project measured *throughput* on small time scales (typically one second) at service end-nodes.

### 1.4.1 MobiHealth

MobiHealth is a research project for the health care sector [3] performed in the time frame of 2001-2004. It was funded by the European Commission, under the "Information Society Technologies" program, and by cooperating partners from five different countries including Ericsson GmbH in Germany, Universiteit Twente The Netherlands, Luleå Tekniska Universitet Sweden; PHILIPS Research Laboratories England, and Telefónica Móviles España Spain.

MobiHealth researched the feasibility for mobile patients to have their health continuously monitored by healthcare professionals in the hospital. For this purpose, the project implemented and evaluated a Body Area Network (BAN) worn by a patient. The BAN contains a personalized sensor system, from which data is (continuously) collected by the Mobile Base Unit (MBU, central unit of the BAN), processed for transmission and transported over a heterogeneous networking environment to the Back-End system (BEsys) located in, e.g., a hospital.

Nine different service trials with prototyped BANs have been carried out in four different countries: The Netherlands, Spain, Sweden, and Germany. For example, in The Netherlands trauma patient trials and pregnancy trials have been conducted, and in Sweden respiratory and physical activity trials.

The MobiHealth project resulted in the development of a methodology for measurement-based performance evaluation of heterogeneous networking environments supporting delivery of mobile health care services. The methodology focused on the one-way delay measurements on application-level message-time scales, hence every message which arrived

at application-level has been time stamped separately. The results which were obtained with executing this methodology, were directly used to adapt the MobiHealth application protocol to the performance of underlying networks, including a UMTS wireless link [4].

This thesis focuses only on the results which are obtained from measurements conducted at the University of Twente in The Netherlands. In these measurements, data from the (mobile) BAN, has been continuously sent to the (fixed) BEsys, with support of a heterogeneous networking environment. This heterogeneous networking environment includes a mobile operator network, a wireless UMTS link, the Internet and the University network. And a dedicated PC was acting as an NTP server for out-of-band synchronization of both the MBU and BEsys. The focus of performance-measurements performed in the MobiHealth project was on the one-way delay and derived application-level throughput from the MBU to BEsys node (uplink direction). For the measurements at the University of Twente a (pre)commercial UMTS network has been used. This resulted in a collection of best-case scenario measurements: performed for a single network user in a stationary geographical position.

### 1.4.2 PIITSA

The Swedish project PIITSA stands for Personal Information in Intelligent Transport systems through Seamless communications[2] and Autonomous decisions [5]. It is a Vinnova[3] project towards Future Communication Networks. The timespan of this project was 2004 – 2006.

PIITSA aims in defining different network functions which are provided by Intelligent Transport Systems (ITS) solutions with seamless communication capabilities, that can manage several communication aspects and decisions.

There are several objectives in the PIITSA project. One of them is to define an open and flexible network function for ITS applications with needs of adjustable services: (best) Network Selection Box (NSB) together with seamless handover. The NSB chooses and switches between different wireless networks and does this by monitoring the speed and the throughput of networks.

The most important activity of this project for this thesis is the wireless communication, which focuses on analysis of communication requirements. In this context, important criteria are network availability, performance, security, and price.

Out of the above mentioned criteria, performance is the most important one for

[2]The user is not aware of which wireless connection it is using. The connection will be changed when it is needed without notifying the user.

[3]Swedish Governmental Agency for Innovation Systems. It promotes the development of effective Swedish innovation systems within the areas of technology, transport, communication, and working lives.

this thesis and, particularly the delays and throughput characteristics of the underlying networks. Similarly to the MobiHealth project, the PIITSA project resulted in the development of a methodology for measurement-based performance evaluation of heterogeneous networking environments supporting delivery of mobile services (in general). This methodology was mainly focused on the throughput measurements on small time scales (in the order of 1 second). The results which were obtained with the methodology execution were directly used to adapt, for example, the video-streaming application protocol to performance of underlying networks, including the GPRS and the UMTS wireless link.

The PIITSA project considered both the uplink and downlink network behaviour direction for data transport. In the uplink direction, the sender is connected to the operator via the GPRS or the UMTS link and the receiver is connected to the Internet, while for the downlink direction it is the opposite. With this setup, each second of packet flow has been time stamped and analysed over an interval of one minute. The project used a User Datagram Protocol (UDP) traffic generator.

## 1.5 Research goal and approach

The following research goal raised will be answered in this thesis:

> *To develop and evaluate a method for the identification of bottleneck conditions in a heterogeneous networking environment based on the relationships between measured one-way delay and throughput performance metrics.*

To address this research goal, we take existing raw measurement data traces from MobiHealth and PIITSA and apply both one-way delay and throughput analysis on them. To get an answer to the research goal, we start our research with the MobiHealth measurement traces analysis and furthermore we validate the results with the PIITSA traces.

The MobiHealth traces are obtained with different saturation factors and with certain saturation factors the network shows a stressed behaviour. In a stressed network it is clear that there are bottlenecks, therefore the method to identify bottleneck conditions will be developed and evaluated with saturation factors of an unstressed network.

## 1.6 Thesis structure

In Chapter 2 the differences in performance evaluation methodology between the two research projects MobiHealth and PIITSA are given. Chapter 3 describes how the measurement data sets were obtained in these projects and used in this thesis. And in Chapter 4 we provide methods for measurement data analysis. With the results of these analyses, the conclusion will be formulated in Chapter 5.

# CHAPTER 2

# Methodologies comparison between MobiHealth and PIITSA

In this chapter the differences between methodologies developed in MobiHealth [4, 6] and PIITSA [5, 7] will be described. The differences are explained in two sections; first some general background topics, important for these projects, will be discussed. Then the more detailed differences between the two projects will be provided.

## 2.1 Background information

In this section we start with some background information important for the two research projects. This information includes the protocol stacks, the transport protocols and information about the Network Time Protocol (NTP). In section 2.1.1, the network layers will be described, some in more detail then others, depending on their importance. The MobiHealth project has chosen TCP as a transport protocol while in the PIITSA project UDP the transport protocol was. Therefore, in section 2.1.2, we provide some characteristics to motivate the choice for both transport protocols.

At the end of section 2.1.3 we provide some more information about the NTP, because of its importance to the MobiHealth project.

### 2.1.1 Protocol stacks

For all measurements performed in the MobiHealth and PIITSA projects, data was transported from a sender to a receiver. But before the data reaches the receiver, it first goes through several layers, each having its own purpose. These layered network reference models are also known as the OSI model [8], where OSI stands for Open System Interconnection reference model, the TCP/IP reference model [9, 10], or the DoD model [11], Department of Defense. In Figure 2.1 the differences between these three reference models are shown.

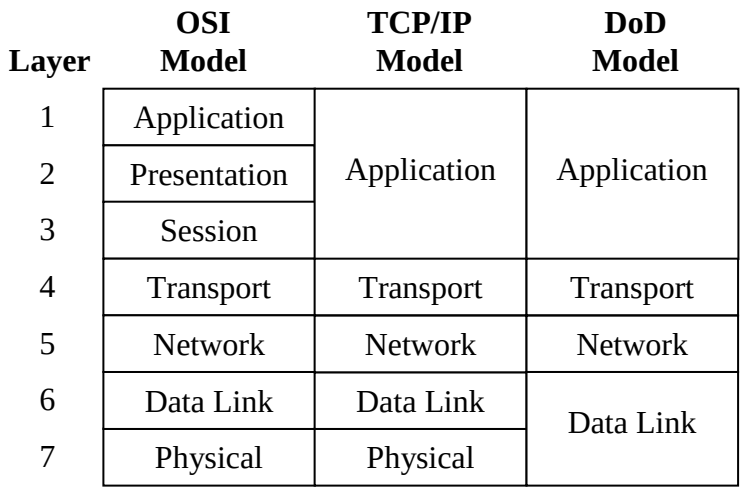



Figure 2.1: Network reference models

In this thesis the focus will be on the TCP/IP reference model. The TCP/IP reference model consists of five layers [9, 11]. Since all layers are in a way connected to each other and thus affecting each other, they will all be illustrated in this section. Particularly, two of these layers, i.e., the transport and the application layer, are more important for this thesis than the remaining layers. Therefore, these two will be described in more detail, while for the three remaining layers only a short description will be given.

Both the PIITSA project and the MobiHealth project were mainly focused on the application layer, as they aimed at measurements of the application-perceived performance. Everything that happened in the lower layers was abstracted from, as it was not possible to put measurement probes along the layers. Hence, everything that happened between the sender and the receiver's application layers was treated as a "black box".

Besides the application layer, also the transport layer is described in more detail, this is because the two projects both use a different transport protocol, TCP or UDP, having different motivations, as we will explain further in this section.

### Application layer

We start with the highest, and most important one for this thesis, layer in the network protocol stack: the application layer. This layer functions as a window through which the application gains access to all of the services provided by the model. It interfaces directly to and performs common application services for the application processes, it also sends requests to the transport layer. By means of the transport layer it provides services for an application program to ensure that effective communication with another application program in a network is possible, but it is not the application itself which is doing the communication. FTP, SMTP and TELNET are some examples which are using this layer. Besides that, the application layer makes sure that the other party is identified and that it can be reached; it authenticates either the message sender, receiver or both; it determines protocol and data syntax rules; it makes sure that the necessary communication resources exist and, in most cases, it also ensures the agreement at both sender and receiver about data integrity, error recovery, and privacy.

### Transport layer

This layer is, in the TCP/IP reference model, responsible for the end-to-end delivery of messages. The purpose of this layer is to provide transparent transfer of data between end-users, thus relieving the upper layers from any concern with providing reliable (TCP) and cost-effective data transfer. Furthermore, it is responsible for governing the transfer of information after a certain route is established by the network layer. It responds to a service request from the application layer and it issues it to the network layer. This was an important layer for the MobiHealth and PIITSA project, because they used different transport protocols. MobiHealth used TCP as its transport protocol and PIITSA UDP. The motivation for these two protocols will be discussed in the next paragraph.

### Internet layer

The third layer is the Internet layer, sometimes also called the network layer. At this layer, the end-to-end transmission of packets takes place, either as datagrams or as virtual-circuits. The transmission is connectionless or connection-oriented. The most important protocol at the Internet layer is the Internet Protocol (IP). The function of this layer is to take care of the connections in the network when the data is going through the network. Thus it is receiving the data from the transport layer and it is forwarding the data to the right destination by sending it to the network access layer.

### Data link and physical layer

The last two layers are the data link and the physical layer. At the data link layer, the transmission of packets on a given link between nodes connected by a communication link, is done. The physical layer arranges the conversion of bits (streams or files) into signals (optical or electrical) that can transfer the information over the network.

### Protocols overhead

The information received by a lower layer is treated as data to be send, and this layer will put its own information in front of that information, i.e., its header, (Figure 2.2). Along the protocol stack there are additional headers (equally for all the three different reference models); all the information that is transferred from a higher layer to a lower layer gets some additional information (called the header) to ensure that the information will be delivered in the right way to the receiver node.

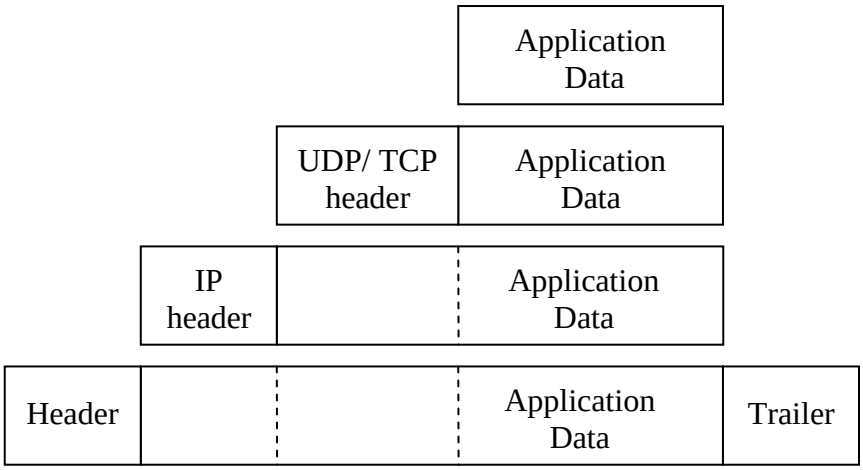



Figure 2.2: Headers per layer of the TCP/IP reference model

As a consequence of the protocols overhead along the TCP/IP protocol stack, application level data at the data link layer is always seen as of bigger size than it is at the application layer itself.

## 2.1.2 TCP vs UDP

In this section the differences between the transport protocols Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) will be illustrated.

### TCP

TCP is a transport protocol commonly used in the Internet. It is connection-oriented, which means that it first requires the establishment of a connection between a sender and a receiver before the application data can be sent. This is particularly done by handshaking (see Figure 2.3); the sender sends a TCP-packet to the receiver with a synchronization flag (SYN-flag), and if the receiver is accepting the connection, it sends an acknowledgement (ACK) packet back, in response to which, the sender will send an ACK packet back to the receiver. From that moment the application data can be sent. The SYN-flag is to

synchronize the sequence numbers of the two entities which are networked. TCP is a reliable protocol, because a sender always knows if the receiver is ready to receive the data, and if the data is received by the receiver. This is facilitated by packet sequencing. The disadvantage of this protocol is that it is slower than if a sender would just send data without waiting for the receiver to indicate that he is ready and available.

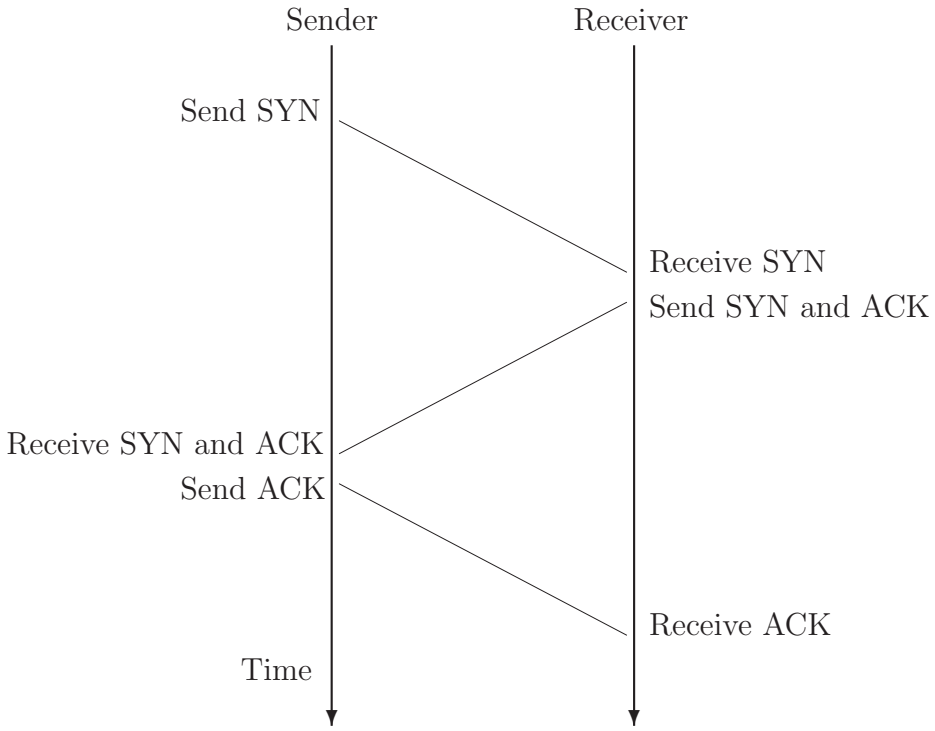



Figure 2.3: Three-way handshake TCP

TCP is transporting application-level data in a streaming fashion. That means that under ideal conditions (no losses and no time-outs), data is streamed.

TCP adapts its data transport speed to a bottleneck network. It is done via monitoring of the Round Trip Time (RTT) and path loss characteristics. Based on the perceived RTT between a sender and a receiver and both loss statistics, TCP decides when to block and forward data at his own convenience.

Furthermore if the packets are arriving out of order at the receiver side, TCP resequences them. Resequencing is possible as TCP is a byte-oriented sequence protocol, which means that a sequence number is necessary to ensure that missing or misordered packets are noted and identified. The disadvantage of this behaviour is that it may affect data streaming.

TCP, as a connection-oriented reliable protocol, is well-suited for file transfers and web browsing.

#### UDP

UDP is, just like TCP, a transport layer protocol. It is one of the basic protocols of the Internet. Compared to TCP, UDP is less reliable. UDP does not use the three-way handshaking before it sends some data, hence it is faster than TCP. UDP is also known as the connectionless protocol. The disadvantage of this protocol is that it does not guarantee that the data will be received by the receiver, so it is very well possible that some packets will be lost without the sender knowing it, and without the possibility of resending the data.

Generally, UDP is faster than TCP because of lack of data verification. Namely, UDP does not verify if packets are received or not, leaving this to the higher application layers. Besides the lack of verification upon receiving the data, UDP also does not verify if the data contains some errors.

Another disadvantage of UDP is that it can overload a bottleneck network, because it does not limit sending data stream.

As UDP is faster than TCP, UDP is used commonly by applications where it is important that the transmission of data is fast and where some data can be lost.

UDP is well-suited for sending small quantities of data and (real-time) streaming, e.g., for multimedia applications as online games and video conferences.

### 2.1.3 Network Time Protocol

The Network Time Protocol is a UDP-based protocol to distribute accurate time through the network from a NTP-server. It synchronizes the clocks of a computer network entity over packet-switched, variable delay, data networks. It is able to synchronize all system clocks distributed within milliseconds from each other over the Internet and up to microseconds over a Local Area Network (LAN).

The reason why NTP is used widely in computer networks are, for example:

- to debug and event time stamps;
- to monitor transaction processes;
- to simulate events;
- to benefit from the fact that NTP calculates the delay of a network when sending the new time;

- to synchronize time over symmetrical links;
- for system maintenance.

Moreover, by using Atomic clocks on the Internet, a Global Position System, or network time servers it is also possible to get the right times for the clocks of the computers. However, we focus only on NTP as used (network time servers) in the MobiHealth project

NTP is also working reliable over symmetrical links, and since wireless links have an asymmetrical link it is not suitable for wireless networks and therefore time synchronization problems may occur. The reason for this is that the NTP protocol takes into account the one-way delay between the NTP server and NTP client when generating the reference time stamp for a NTP client [12, page 117]. For a wireless asymmetrical link, this delay will vary resulting in incorrect NTP synchronization. The MobiHealth project dealt with that by using the NTP out-of-bound synchronization method (section 2.2.7).

## 2.2 Differences in methodologies

With this background information of applications and methods which were used in the projects we now explain the differences in the performance evaluation methodologies of the two, previously discussed, research projects.

### 2.2.1 Measurements setup

The first difference between the two projects is the measurement setup. However in both projects, the heterogeneous networking environment includes the mobile operator network – containing a wireless access network, e.g., GPRS or UMTS –, the Internet and the university network.

#### PIITSA

Two computers have been used for the measurements. Both could generate traffic in uplink direction or receive it in a downlink, depending on needs. Figure 2.4 shows the measurement setup used in PIITSA.

#### MobiHealth

In the Netherlands, where the MobiHealth project has been conducted, a requirement for time synchronized clocks of measurement nodes has been posted. It has been achieved via using an extra computer which acted as a NTP server in a separately dedicated LAN for out-of-bound time synchronization. So both the fixed computer as the (mobile) laptop were connected to the third computer by Ethernet (Figure 2.5). And the laptop, which is acting as a sender, was sending the data over the UMTS network. The computer,

which is the receiver, was receiving the data by a wired university connection. The focus of MobiHealth was the uplink direction, however, there were also conducted some experiments in downlink, as we explain in following sections.

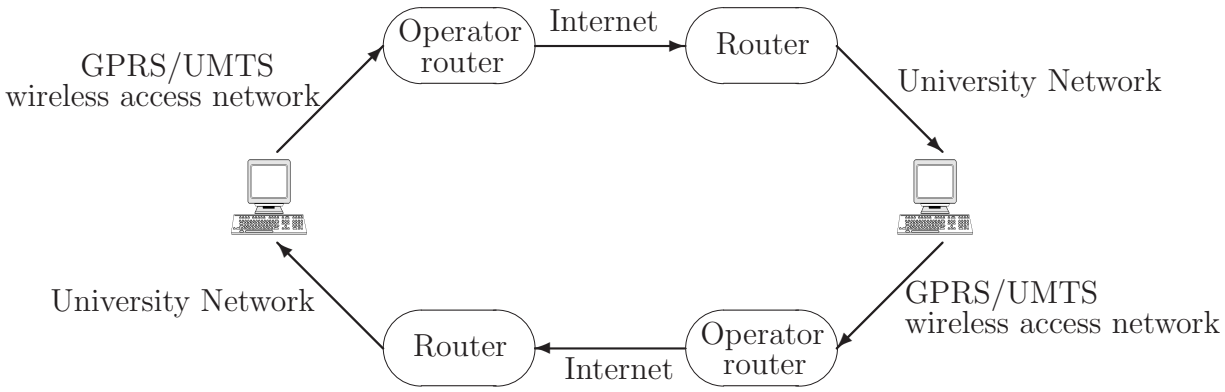



Figure 2.4: Measurement setup in the PIITSA project

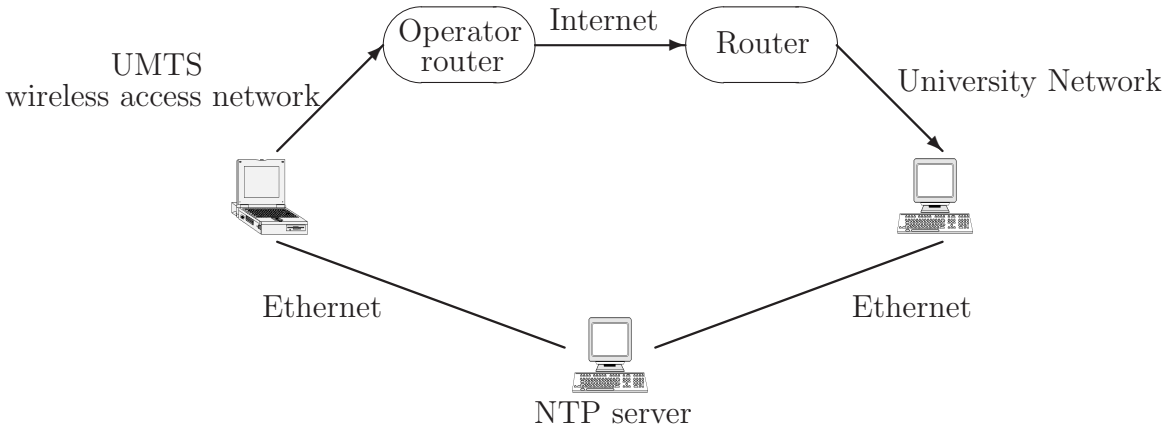



Figure 2.5: Measurement setup in the MobiHealth project

### 2.2.2 Performance parameters of interest

The performance parameters of interest were an important factor for the setup of the measurements in both research projects. The MobiHealth project measured the one-way delay and derived the application-level throughput from it, while the PIITSA measured the application-perceived throughput. The PIITSA project is using the name application-perceived throughput, because the throughput is measured at the application level, whereas throughput is usually measured at the network or the transport layer.

| | |
|---|---|
| **Application-perceived throughput** | The throughput received at the application level. Throughput is amount of data send per time unit. |
| **One-way delay** | The time interval for the transport of the data from a sender to a receiver. |
| **Application-derived throughput** | The throughput received at application level. Throughput is amount of data send per time unit; in the MobiHealth project the throughput was derived from the delay. The application-derived throughput was derived in both uplink and downlink, in uplink this is derived by $\frac{send\ message\ size}{uplink\ delay}$. |

A commonly used term in this context is goodput, but this term often leads to confusion. If we go back to the part about the TCP/IP reference model in the section 2.1, we read that usually at the application layer there are no extra headers. This means that, in general, the goodput will be lower than the throughput, because the goodput is measured at the application level (no headers) and the throughput is often measured between the physical and the data link layer (added headers). Thus, in general one can say that the goodput and the throughput are not the same, but in these two research projects it is, because the PIITSA project measured the throughput at the application level (assuming no headers), and therefore we changed the term into application-derived throughput.

### 2.2.3 Workload parameters

An important difference between the two research projects are the characteristics of workload, expressed in terms of the message sizes used for sending application-level data. The MobiHealth project used multiple message sizes while the PIITSA project used just one message size.

#### MobiHealth

The message sizes which were being used, in the Netherlands were in the range of 174-8122 bytes for uplink and 174-48208 bytes for downlink; in total there were 400 different combinations. The mobile operator network was UMTS. Furthermore, TCP was in charge of a possible segmentation of the messages.

#### PIITSA

In the PIITSA project, one message size has been used, namely 128 bytes for sending data over GPRS and 480 bytes for UMTS. Compared to the MobiHealth project, the PIITSA project had no variable message sizes, but it had variable Inter Packet Delays (IPDs).

This yields different offered loads. As messages have been sent over UDP, no message segmentation occurred.

### 2.2.4 Transport protocol

In section 2.1.2, we presented information about TCP and UDP. In this section we focus on motivations for projects to use either one of the two.

#### MobiHealth

The TCP transport protocol was used in the MobiHealth project, because of the hard requirement of reliable communication between sender and receiver. The MobiHealth application protocol carries important data related to the health of a monitored person. Therefore, by using TCP, MobiHealth assured that the patient data will arrive at the hospital in order.

#### PIITSA

The PIITSA project used UDP as a transport protocol. It is because the project focused on sending as much data as fast as possible and on observing of the network behaviour. The most suitable transport protocol for that is UDP.

### 2.2.5 Software implementation

In practice, the question rises often upon which programming language is going to be used to implement the research ideas. The same question applied to MobiHealth and PIITSA. Finally, the MobiHealth project used Java and the PIITSA project used C#.

The choice for the Java language in MobiHealth has been motivated by its flexibility. PIITSAs choice of C# in the .NET framework has been motivated by its stability, performance, and availability of a Windows OS platform. Both Java and C# have object-oriented development capabilities.

An important vision of Java is "Write once, run anywhere" [13]. Therefore, the difference between Java and .NET (to which C# belongs) is that Java can be used on different Operation Systems (OS) and .NET just on one OS: Microsoft Windows.

### 2.2.6 Workload generation and time stamping

Another difference between the two research projects is how the workload has been generated and how the data from this workload has been time stamped.

The PIITSA project time stamped application data flow on small time scales, typically on an one second time scale. Therefore, not every message was time stamped. MobiHealth, on the other hand, time stamped every message at the application-level.

### PIITSA

The PIITSA generator tries to send the UDP packet as regularly as possible. It does this by sending the traffic, i.e., UDP datagrams with a constant length and a sequence number in each datagram, with a specified interval (IPD) to get a certain workload.

Figure 2.6 shows how the workload generator of the PIITSA project is working [7, 14]. When the generator sends its first UDP packet, it starts with keeping the time stamp (TS) of the starting time as a reference for upcoming packets. After generating a new UDP packet, a new time stamp (TS1) will be generated. After this time stamp the new packet will enter a *"while"* loop to send the packets with a certain nominal Inter Packet Delay (IPD). At the beginning (TS2a) and end (TS2b) of the loop the packet will be time-stamped too, and when TS2b equals TS2a the packet will be released from the loop to be send. Before (TS3) and after (TS4) sending the UDP packet, it will be time stamped. Ideally, TS3 equals TS2. TS3 is being used for the measurements of the PIITSA project as a time stamp when the packet has been send.

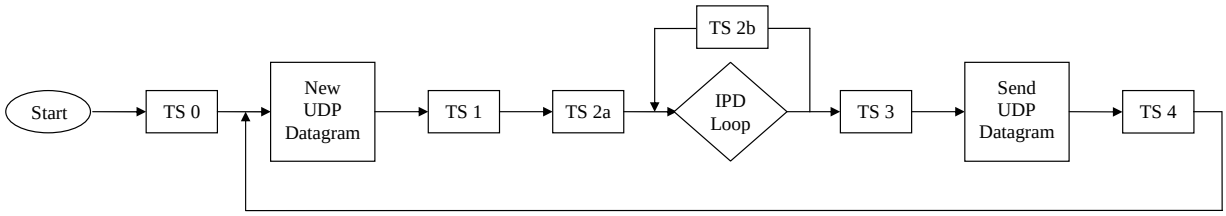



Figure 2.6: UDP workload generator PIITSA project

The received time stamps had originally a resolution of 10 milliseconds, but a higher resolution was required, so the time stamp resolution was improved to one thousand of a millisecond. It was achieved by use of the performance counters – QueryPerformanceCounter[4] and QueryPerformanceFrequency[5] of kernel32.dll[6] – in the C# code [15].

### MobiHealth

For the MobiHealth project the workloads have been generated in a different way. At this research project a certain number of packets is to be send in a time interval of 1 second and

[4]Retrieves the current value of the high-resolution performance counter.
[5]Retrieves the frequency of the high-resolution performance counter, if one exists.
[6]It handles memory management, input/output operations, and interrupts.

the sender tries to keep this number of packets every second. Calculations of the number of packets per time interval are based on the given IPD which is maintained by the Java [16] function Thread.sleep[7]. If we look at Figure 2.6 and compare what happens at the MobiHealth project, we can say that the MobiHealth project only time stamps at TS4, hence only after sending the packet. Time stamps are obtained from the Java function System.currentTimeMillis[8].

### 2.2.7 Time synchronization

Time synchronization is required if one wants to correlate events from different networks, routers, or computers. Because the MobiHealth project focused on measuring the one-way delay, by comparing the time stamps at both the sender and the receiver, it has been required to have clocks of sender and receiver time synchronized.

Hence, MobiHealth used NTP time synchronization protocol with Tardis application for the time synchronization. Tardis is a shareware utility which ensures that a PC clock is synchronized [17]. There are several ways to put the clock in a right time; for example by Atomic clocks on the Internet, a Global Position System, or network time servers. MobiHealth used the last option when synchronizing sender and receiver with the dedicated NTP server.

There are two ways to send the synchronization information from a sender to a receiver: interleaving (inbound time synchronization) or multiplexing (out-of-bound time synchronization) [18]. With in-bound time synchronization, the synchronization information is added to the data unit, to be send together over the same channel (Figure 2.7) as data. And with out-of-bound time synchronization the data streams will not be sent over the same communication channel; one channel is used for the data stream while another channel is used for the synchronization information (Figure 2.8). The latter channel should be very reliable, so that the synchronization information is available when the data from the other channel is received by the receiver.

The MobiHealth project used the out-of-bound time synchronization method. It is because a reliable channel, which is dedicated only for exchanging the time synchronization packets between the sender and receiver and the NTP time-server, was required. But the channel over which the service data has been sent, which is a dedicated LAN network, was a wireless link, as required.

[7] Causes the currently executing thread to sleep for the specified number of milliseconds.
[8] Returns the current time in milliseconds.

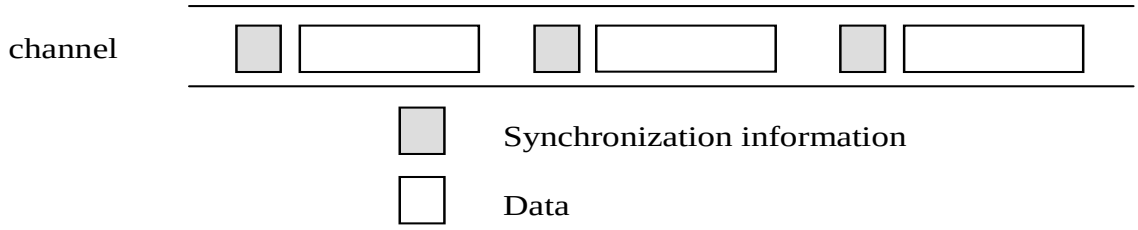



Figure 2.7: Inbound time synchronization

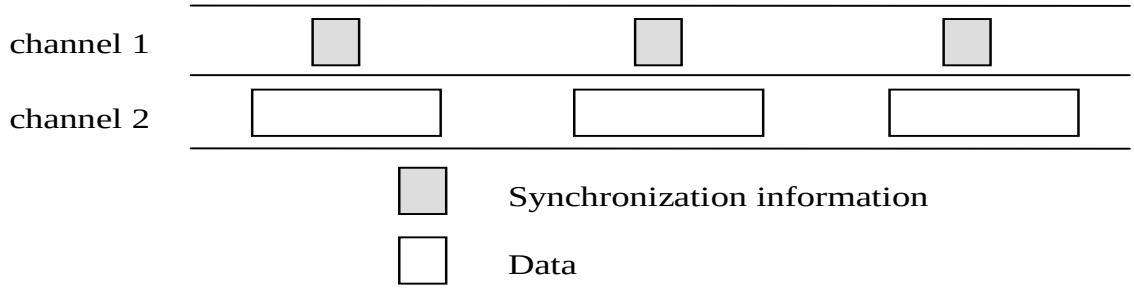



Figure 2.8: Out-of-bound time synchronization

### 2.2.8 Output

Besides the differences in the setup, both research projects differed in measurements and protocols used, resulting in different results and their presentation.

#### MobiHealth

In the MobiHealth project the first results showed raw data which are exposing a behaviour related to the different bearers (i.e., data carrying channels) in the UMTS network. Cumulative graphs of the application-derived throughput, where one could see that the application-derived throughput of UMTS depends on the bearer assignment policy, has been derived later. This cumulative graph shows that the application-derived throughput is asymmetric. From the project, also the uplink, downlink and total delays according to the different message sizes has been plotted. In this graph one can see that the uplink delay increases linearly with the message size.

In the MobiHealth project it has been found that the UMTS application-derived throughput bottleneck depends on the message sizes which are used in uplink and downlink; and for the uplink the bottleneck is 53kbps and for downlink it is at 300kbps. The change of the UMTS bearer depends on the data volume which is transported in both uplink and downlink. However, the bearer assignment policy is of unknown nature, and seems to be difficult to model.

#### PIITSA

In the PIITSA project, throughput histograms have been used to show the uplink and the downlink behaviour for both the sender and the receiver. The comparison of these two histograms gives information on nature and "severity" of the bottleneck. Besides, a histogram difference plot shows an impact on the throughput histogram and on the throughput on both sender and receiver. By the shape of this plot one can get information about the nature of the bottleneck. Furthermore, autocorrelation plots have been plotted in PIITSA to expose an eventual throughput periodicity behaviour.

With these histograms and plots PIITSA showed that both GPRS as UMTS have jitter[9]. In GPRS jitter is always available, while in UMTS one could say that it is almost transparent (i.e., it does not have any significant impact on the traffic statistics) if it uses a small share of the nominal capacity[10] of the network. Furthermore, the PIITSA project found a UMTS bottleneck in the uplink of 59 kbps and at downlink of about 360 kbps. Both projects came up with similar conclusions that UMTS is suitable for streaming services, but GPRS is not.

### 2.2.9 Initial delay vs changing bearer discovery

In both projects two important issues were discovered and worth to mention.

#### PIITSA

The PIITSA project discovered some initial delay in the UMTS network, while the MobiHealth project did not. The initial delay is the time it takes before the first application-level message which is sent, arrives at the receiver with a delay which is more than the average one-way delay. According to [19], the substantially higher delay of the first packet, compared to the packets following it in a packet stream, is caused by a Temporary Block Flow (TBF) setup times and is very specific to the UMTS network implementation.

#### MobiHealth

On the other hand, in the MobiHealth project, the UMTS bearer changing behaviour has been exposed which was not the case in PIITSA.

In summary, quite a few important differences were between the two projects. Some of these differences, i.e., the clock synchronization, and workload parameters, are important for the rest of the work. In the next chapter we will first describe the traces which we received from the two different research projects and mainly how these traces have been conducted.

[9] Variable part of the delay, due to queueing

[10] The required capacity which can be different from the actual sending capacity

# CHAPTER 3

# Data traces

In this chapter the traces will be described. We start with the presentation of the data from the MobiHealth project (section 3.1 and 3.2), because this data will mainly be used for the analysis, which will be then checked with selected data from the PIITSA project (section 3.3).

## 3.1 MobiHealth short traces

We have obtained data for 30 different short traces of 30 seconds, but we also have data of a long trace of 900 seconds.

The project goals have been explained in section 1.4.1, while the setup for the traces in section 2.2.1. a UMTS network with an uplink nominal capacity of 64 kbps has been used, and the MobiHealth team was the only user in this network. This is because the project used a (pre)commercial UMTS network. The measurements were done at a stationary geographical position. The mobile node was a laptop and a USB mobile phone. These measurements were done in a best-case scenario.

Furthermore, the data was send along the unconfirmed service type, meaning application-level connectionless service, thus the sender does not wait for a reply of the receivers before sending a new data (Figure 3.1). So compared to the confirmed service (Figure 3.2), there are two transport service requests, i.e., the request and following confirmation from the sender and the indication and response of the receiver, so two exchanges of protocol messages.

In the MobiHealth project the measurements were done mainly in the uplink direction, so the data was send from the sender to the receiver, where the sender was the mobile node and the receiver was the fixed node. The measurements in the downlink direction were not of high importance for the MobiHealth project, because at the moment it is only important to receive in a reliable and time manner the vital signals from the mobile patient, e.g., his heart rate, by the receiver in the hospital and not to give a response. The nominal capacity of UMTS for sending data in an uplink direction is lower than sending data in a downlink direction, 64 kbps and 384 kbps, respectively. The data was always sent as a message size of 524 bytes at the application level.

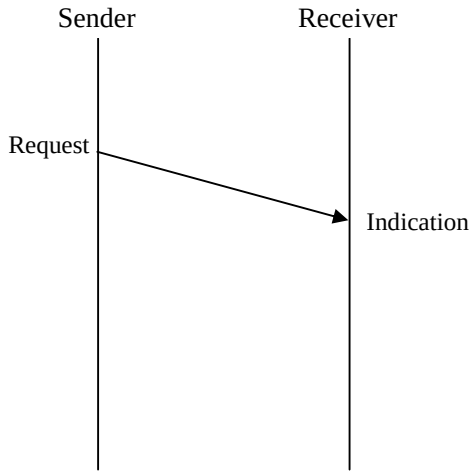



Figure 3.1: Unconfirmed service element

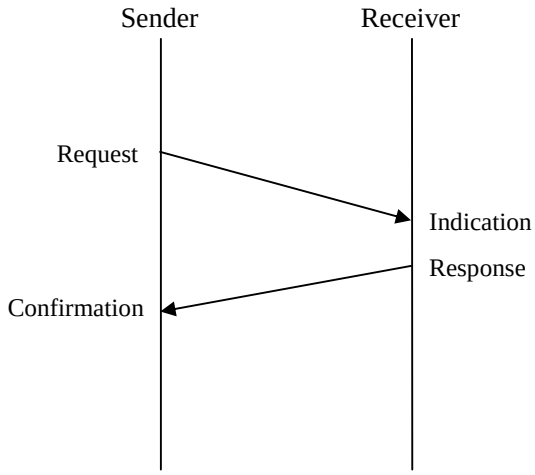



Figure 3.2: Confirmed service element

For all the short traces, the measurements were done for about 40 seconds, and repeated 30 times. The 40 seconds include a so-called *"slow start"*, meaning that the measurements start with a sending rate of 1 packet per second, then 2 packets per second, and so on, this to avoid queueing of packets due to UMTS barrier switching behaviour. This slow start phase is ignored for our analysis in this thesis (details in the next chapter). During the slow start a backoff time of 500 ms has been used. Between the measurement repetitions a backoff time of 30000 ms has been used. This backoff time is to "calm down" the network and to start the new measurement with a "clean" network, so that there is no traffic left in the network and that the bearer assignment behaviour of the UMTS is also "restarted". Besides that the measurements were done for different saturation factors, indicating network saturation level, as a function of nominal capacity, i.e., theoretical capacity of the network. For a saturation factor of 1.0, data has been sent assuming saturation of 100% of network nominal capacity, resulting in 14 packets per second. In Table 3.1 we present the different saturation factors which are used and the corresponding number of packets per second. Thus for a link saturation factor 0.5, application flow saturates a half uplink capacity, i.e., uses 32 kbps and sends 7 packets a second over that link.

| SAT[11] | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 | 1.1 | 1.2 | 1.5 | 2.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| packets per second | 7 | 8 | 9 | 11 | 12 | 14 | 15 | 16 | 21 | 28 |
| repetitions | 30 | 30 | 30 | 30 | 30 | 30 | 30+ | 30 | 30 | 3 |

Table 3.1: Workload parameters: Saturation factor, corresponding number of packets per second and number of measurement repetitions

As mentioned before, the UMTS network which was used during the experiment had an uplink capacity of 64 kbps. And sending packets with a saturation factor equal to and larger than 1.0 stresses the network, which is clear from the data. Figure 3.3b, presents a network behaviour if it is stressed, while Figure 3.3a presents the network behaviour when it is not stressed. For SAT 2.0 a major hardware (mobile phone) crash has been experienced, due to buffer overflows. Hence these measurements have been dropped after only three repetitions.

When the link got saturated and the network got stressed for more than 12 packets a second, the TCP-congestion control mechanism has been activated. It prevents an application from sending too many packets to the network, but at the other hand, it causes that packets are buffered, that can result in buffer overflow.

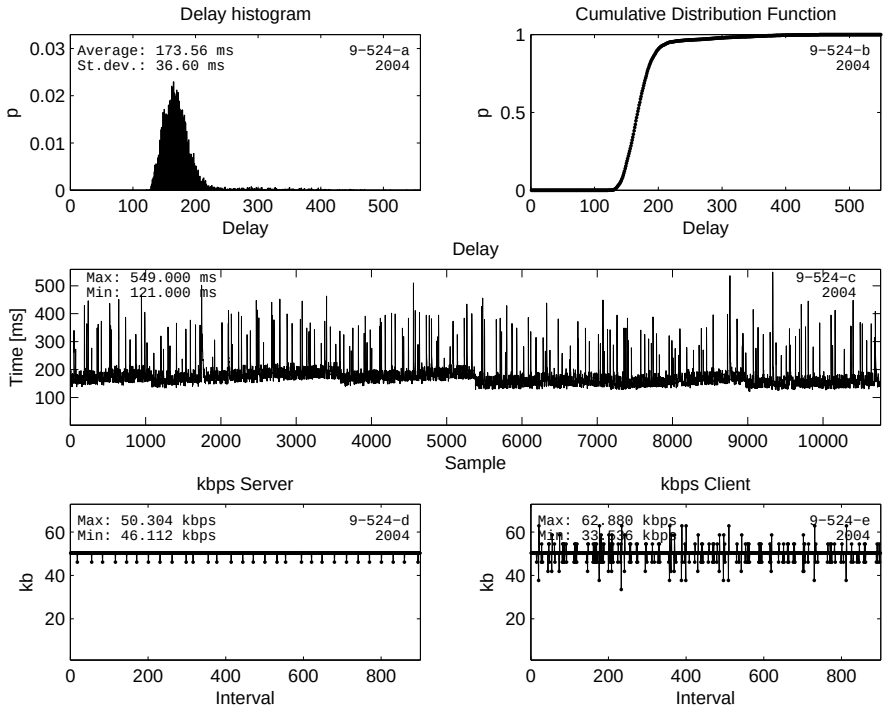
Because of the fact that the network is stressed from saturation factor 1.0 ongoing, we have chosen to work with the MobiHealth measurements data for saturation factors which are smaller than 1.0. The analysis was done for data with saturation factor 0.9 (12 packets per seconds were being send, which is easy to divide if one wants to look at smaller time intervals).

## 3.2 MobiHealth long trace

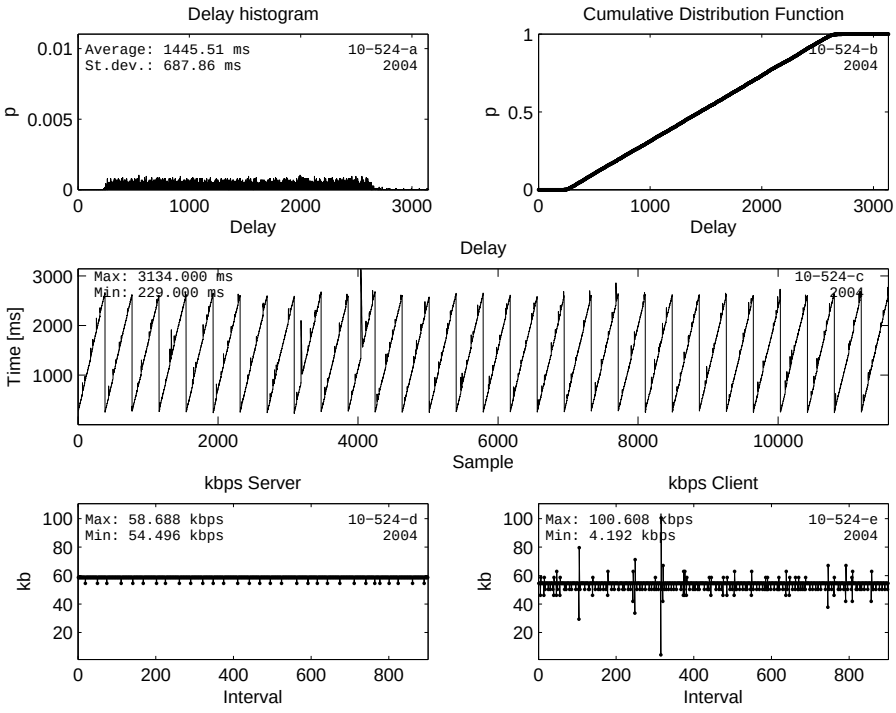
The MobiHealth long trace is almost the same as the short traces (Table 3.2). In the setup, there is a small difference, because the short traces were obtained using a (pre)commercial UMTS network (in year 2004), while for the long trace, a commercial UMTS network has been used (year 2006). So the MobiHealth team was not the only user in the UMTS network anymore. Nevertheless the measurements were still done at a stationary geographical position.

The measurements for the long trace were only performed for a saturation factor of 0.5, which corresponds to 7 packets per second. The link capacity is 32768 bps, for network nominal capacity of 64 kbps. In the slow start phase a backoff time of 500 ms has been used. There were no repetitions so the backoff time between the samples, changed from 30000 ms to 1000 ms when comparing to short traces, is not important here.

[11]Saturation factor




(a) Saturation factor 0.9: normal network behaviour




(b) Saturation factor 1.0: stressed network behaviour

Figure 3.3: Network behaviour for a light-load and a heavy-load

| | Short trace | Long trace |
|---|---|---|
| duration | 40 seconds | 900 seconds |
| network | (pre)commercial | commercial |
| users | single-user | multiple users |
| position | stationary | stationary |
| repetitions | 30 | 1 |
| slow start | yes | yes |
| backoff time slow start | 500 ms | 500 ms |
| backoff time | 30000 ms | 1000 ms |
| service | unconfirmed | unconfirmed |
| SAT | 0.5–2.0 | 0.5 |
| nominal link capacity | 64 kbps | 64 kbps |
| year of measurements | 2004 | 2006 |

Table 3.2: Overview of the MobiHealth short and long measurement traces

## 3.3 PIITSA traces

The PIITSA traces are the traces which are conducted in Sweden. We have two different kind of traces for this section, the first traces are with an IPD of 50 ms and 60 ms and they are for both uplink and downlink (Table 3.3), while the second part consists of traces which are based on the saturation factors of the MobiHealth project, these saturation factors are 0.5–0.9, hence IPDs of 83 ms – 166 ms.

### 3.3.1 PIITSA traces with an IPD of 50 ms and 60 ms

The PIITSA project provided us also with four different kind of traces; traces in uplink and downlink and for an IPD of 50 ms and 60 ms. The measurements [20] were done in a GPRS and a UMTS network. In this thesis we will focus us on the UMTS network, because the traces of the MobiHealth project were only focused on the UMTS network and thus for the PIITSA traces we will also use the UMTS network so we can compare the results.

Furthermore, nominal link capacities for UMTS were 64 kbps uplink and 348 kbps downlink. The UMTS network which was used is available for everybody in Sweden, thus the PIITSA project was not the only user in the network. Compared to the MobiHealth project no slow start was used, because of the fact that the UDP transport protocol was used.

| | PIITSA traces |
|---|---|
| network | commercial |
| users | multiple users |
| position | stationary |
| slow start | no |
| IPD | 50 ms – 60 ms |
| nominal link capacity | 64 kbps (uplink) 348 kbps (downlink) |
| year of measurements | 2004 |

Table 3.3: Overview of the PIITSA traces

### 3.3.2 PIITSA traces with different IPDs and a Traffic Shaper

For these traces a new experiment has been done [14]. This is conducted in Sweden, where a new setup has been made. This setup consisted of two nodes: a source and a destination. The measurements were conducted in both uplink and downlink. Furthermore, the tool of the MobiHealth and the PIITSA project was installed on both nodes and this tool generated different amount of packets for a period of 25 minutes. The loads which were generated are from a saturation factor of 0.5 until a saturation factor of 0.9, hence 7–12 packets per second. The packets all had a size of 524 bytes.

For these traces a traffic shaper (see Figure 3.4) has been used. The traffic shaper controls the traffic for a better throughput and performance. It is doing this by rescaling the packets, if a packet is delayed when it arrives at the traffic shaper, the shaper spaces them equally, hence with a certain IPD, again and sends them to the receiver. Therefore, it is uncommon if there are a lot of outliers in these measurements.

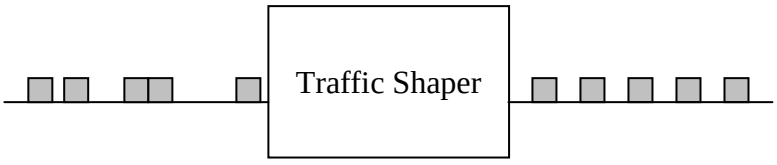



Figure 3.4: Traffic Shaper

With this setup different saturation factors (0.5–0.9) were used, but to compare it to the PIITSA traces, the saturation factors 0.6 and 0.8 were used. This is because for these saturation factors we get an IPD of 90 ms and 125 ms (Table 3.4). But for the analysis we will focus on the saturation factor of 0.9, because then we can compare the results to the MobiHealth traces. Furthermore these traces have a duration of 40 seconds and repeated 30 times. A last thing is that there is no slow start used for these measurements, hence the full number of packets are sent from the start.

| SAT | 0.6 | 0.8 |
|---|---|---|
| Number of packets per second | 8 | 11 |
| IPD [ms] | 125 | 90 |
| OWTT [ms] (traffic shaper) | 180 | 180 |

Table 3.4: Proposed Workload Parameters

There are some difference between these traces and both the PIITSA and the MobiHealth traces. The difference between these traces and the PIITSA traces is of course the time synchronization, for these traces the clocks on both computers are synchronized whereas for the PIITSA traces the two clocks are not synchronized. Another difference is the IPD, the PIITSA traces we use here in this thesis are trying to send a packet every 50 ms or 60 ms, this means, respectively, 16 and 20 packets per second, while these traces are more based on the MobiHealth project with a saturation factor of 0.9, thus sending 12 packets per second. But of course this difference is only valid for this thesis, because there are traces from the PIITSA project with an IPD of 90 ms and 125 ms. A difference with the MobiHealth traces is that in these traces no slow start is used and also no backoff time between the repetitions.

# CHAPTER 4

# Trace analysis

In this chapter the results of the measurement data analysis will be discussed. The main goal of this chapter is to come with the method which is described in the research goal which is presented in the introduction. To this end, we divide this chapter into two parts; firstly, the results of the analysis of the traces from the MobiHealth project will be discussed (section 4.1 and 4.2), and secondly, we give the method described in the research goal based on the additional delay and the variation of the time interval, over which the measurement traces from MobiHealth and PIITSA are analysed. The answers to this goal will both be qualitative and quantitative. This will be followed by the analysis of the PIITSA traces (section 4.3). After analysis of these traces, we come to the second part of this chapter, where firstly in section 4.4 we have a look at the analysis of the tail of the one-way delay followed by the analysis of measurement data while varying the time interval in section 4.5.

## 4.1 MobiHealth short traces

This section we start with the individual analysis of the MobiHealth traces. This will be done for different saturation factors (SATs), but mainly based on the data for SAT 0.9, where 12 packets per second were sent in multiple short time-frame traces. After this, these short traces will be joint to have a look at all them as one long trace.

### 4.1.1 Individual analysis

For the individual (short) traces, we first have a look at the saturation factors 0.5, 0.9, 1.0, and 1.5. The figures of the remaining saturation factors can be found in Appendix A.

With the lowest saturation factor 0.5 (Figure 4.1), which corresponds to sending 7 packets per second, the observed network behaviour is stable. At the sender side, the packets are sent with an IPD between 140 ms and 150 ms; on average IPD equals to 142.73 ms and is close to the required $\frac{1000}{7} = 142.86$ ms). Whereas at the receiver side the IPD is a bit more variable; there are more outliers. For example, there is a sample (packet number 126) where a packet is arriving at the same time as the previous packet (IPD = 0). But even when there is more variation at the receiver side, the average IPD is still close to the required IPD, namely 142.78 ms. Furthermore, when we look at the one-way delay plot of Figure 4.1, we see that there are some samples which have a relatively high delay. Correlating it with the IPD of the receiver we see that there are some high IPDs; particularly we see that the high delay is at the same packet number as the high IPD, which is normal. The lower plots of these figures expose that at the sender the sending behaviour is the same all the time, while at the client side there are a 2 packets which arrived in the next interval (1 second). We can see this also in the throughput difference histogram, which is exposing a shared bottleneck behaviour [7]. However, we conclude that the network has no problems with these number of packets per second and we observe its behaviour as expected.

In Figure 4.2 (SAT 0.9), it is clear that the sender is sending the packets regularly, while for the receiver this is not the case. There are more fluctuations at the receiver side. This can also be found in the third picture of the graph where the delay is shown, when comparing to SAT 0.5. A high IPD at the receiver side, corresponds to a high delay in the delay plot. Furthermore, it is obvious, that because of the regular IPD at the sender, the throughput at the sender is also more stable than at the receiver. From the last two throughput plots, the throughput histogram difference plot is made. In this plot one can see that the bottleneck is shared, which means that the burstiness of the packet stream has increased.

In Figure 4.3 we present results for SAT 1.0, where 14 packets per second were sent. The expected IPD would be $\frac{1000}{14} = 71.43$ ms. In the plot for the sender we have an IPD between 70 ms and 80 ms, with an average of 71.54 ms. Hence, we conclude that the sender is still able to send the packets regularly. But the receiver at the other hand start to have some problems in receiving the packets; the average here is 77.76 ms, and it is rising. When we look at the delay plot, we see that from the start, the delay is increasing, which could mean that sending 14 packets a second can not be handled by the network. The strange thing here is that the throughput in kbps is still not that variable as expected when looking at the delay and the average IPD.

Finally, the results for SAT 1.5, while sending 21 packets a second, are presented in Figure 4.4. In the IPD plots we can see that at the beginning of the measurements the behaviour is as expected, but from a certain point, the network can not keep up with the amount of application data it would like to send, hence the IPDs are fluctuating very

much at both sender as receiver. At the receiver less packets arrive in an interval, the IPD is higher than at the sending side, respectively 75.62 ms and 59.39 ms. This time, not only from the delay and IPD plots it is clear that the network has some problems with sending and receiving 21 packets a second, but also the plots of the bandwidth show more variation. At the beginning the sender tries to send the 21 packets a second, which succeeds, but halfway the trace it seems to be that it is able to send just an average of 13 packets a second. Whereas the receiver is showing this behaviour from the start, it is receiving 13 packets a second and towards the end, it is still receiving on average 13 packets a second, but with more fluctuation. Furthermore, the throughput histogram difference plot for this saturation factor is showing a typical overloaded bottleneck.

In summary, the network does not exhibit problems in behaviour until 12 packets (saturation factor 0.9) per second are offered by the application to the network; for 14 packets per second the sender still has no problems to send them, but the receiver is having some problems to receive them in a continuous way. This is seen by the fact that the average IPD is rising at the receiver, which means that there are less packets per second received. Furthermore, from saturation factor 1.0 on, the delay is increasing, packets are probably buffering somewhere before arriving at the receiver. In Figure 4.4 one can see that the sender can not get the expected number of packets send through in one second; the number of packets are lower at the receiver than at the sender, this is shown by the lower throughput, in the throughput plots, at the receiver side. And besides the throughput plots, this behaviour is also exhibited in the throughput difference histograms. For a saturation factor lower than 1.0 these histograms show a shared bottleneck, while for the saturation factors of 1.0 and higher, this shared bottleneck disappeared completely. For these reasons, we further focus on measurement data obtained for the saturation factor 0.9 only, corresponding to 12 packets per second; we continue with these measurements analysis to find answers to the research questions posed in this thesis.

### 4.1.2 Joint analysis

In this section, the joint analysis of all the short traces of 30 seconds of the MobiHealth project put together is given, to get a better overview of the network behaviour for one saturation factor (SAT 0.9), and to see if all those traces exhibit roughly the same behaviour.

For the joint traces of the MobiHealth project we go back to Figure 3.3a, in this figure the joint traces for a saturation factor of 0.9 are shown. In this figure we see that the main part of the delay histograms shows a normal behaviour, but that it also has a right-hand sided tail, caused by the one-way delay outliers. This is also illustrated in the delay plot; one can see here that most of the packets have a delay between 150 ms and 200 ms (with an average of 173.56 ms), but that there are quite a few outliers. This can also be found in the throughput plot of the receiver; mainly in the spikes which are bigger than 54.496 kbps and smaller than 46.112 kbps. A throughput of 54.496 kbps and 46.112 kbps can happen,

because then just one packet arrives in the next time interval, but with throughputs other

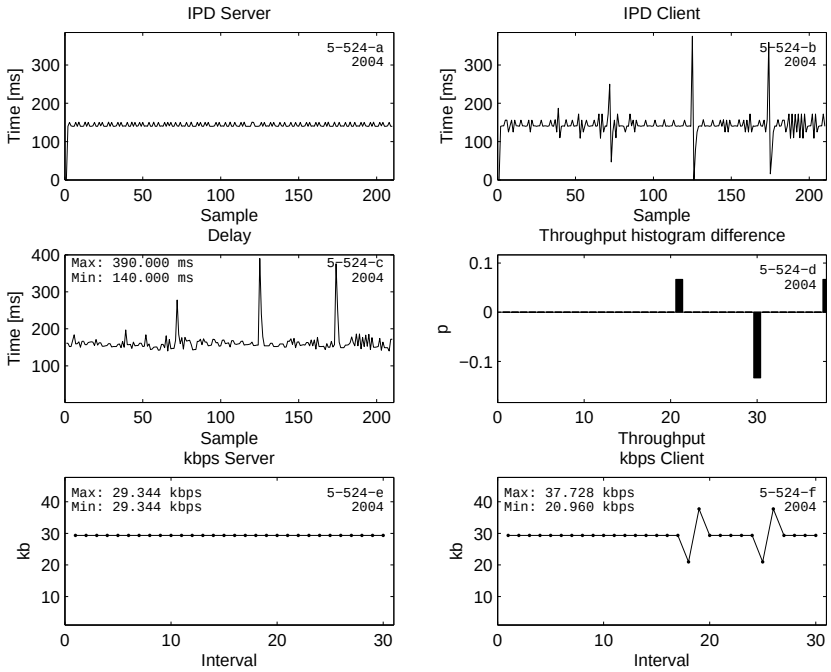



Figure 4.1: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 0.5

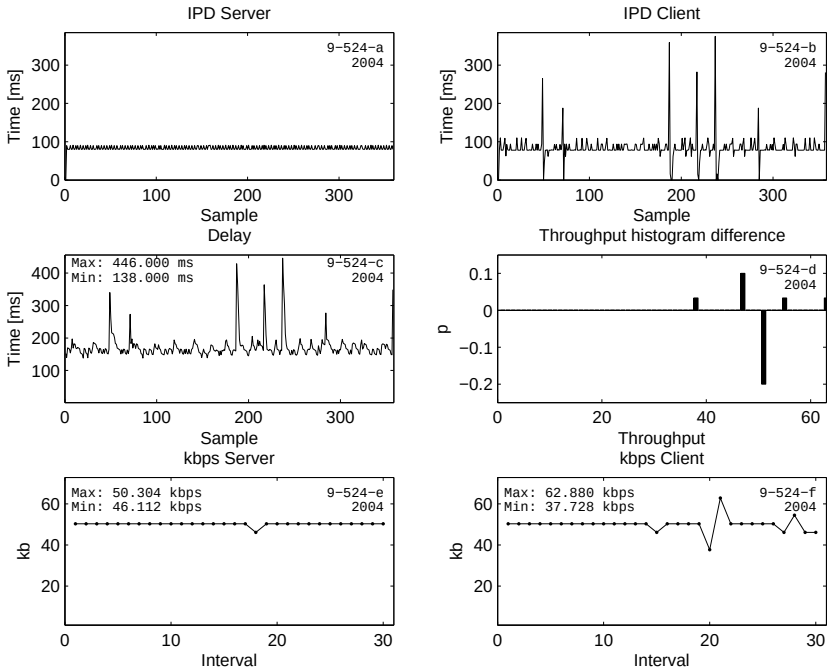



Figure 4.2: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 0.9

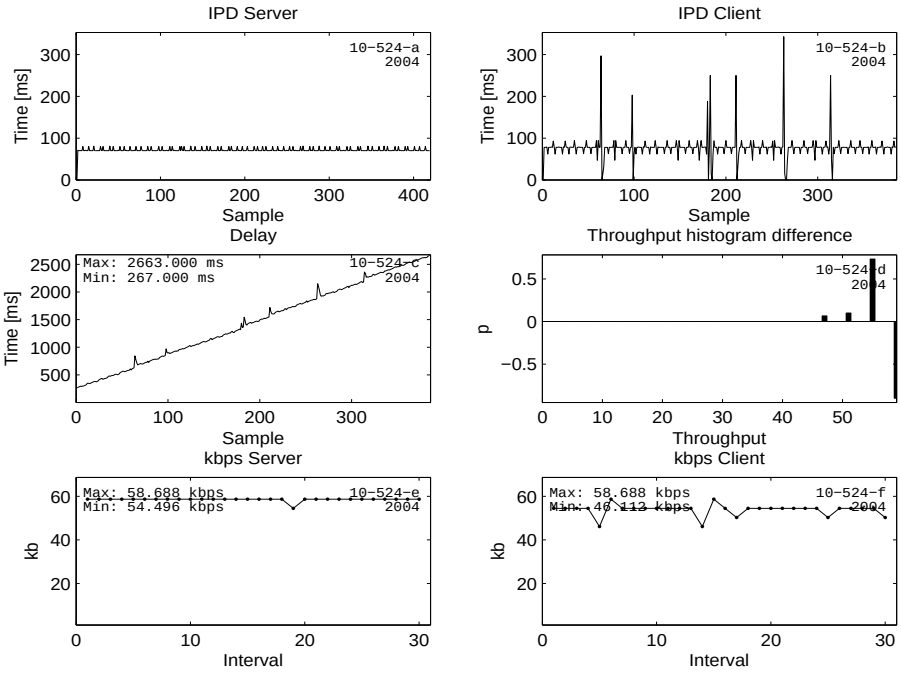



Figure 4.3: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 1.0

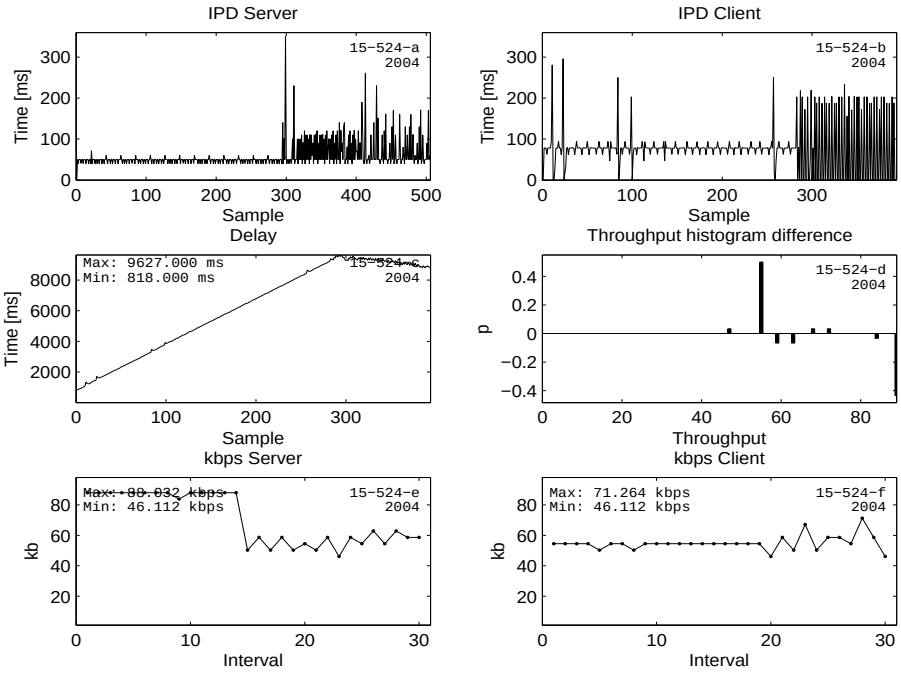



Figure 4.4: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 1.5

than the average and these mentioned ones, it is assumable that it is caused by the network delays. A last thing which is nice to see in the delay plot of the joint traces are the time synchronization points. As known, the MobiHealth project used the NTP for the sender's and receiver's clock synchronization, and in this figure we see that the clocks had some drift and we also see the moments at which the clocks were synchronized.

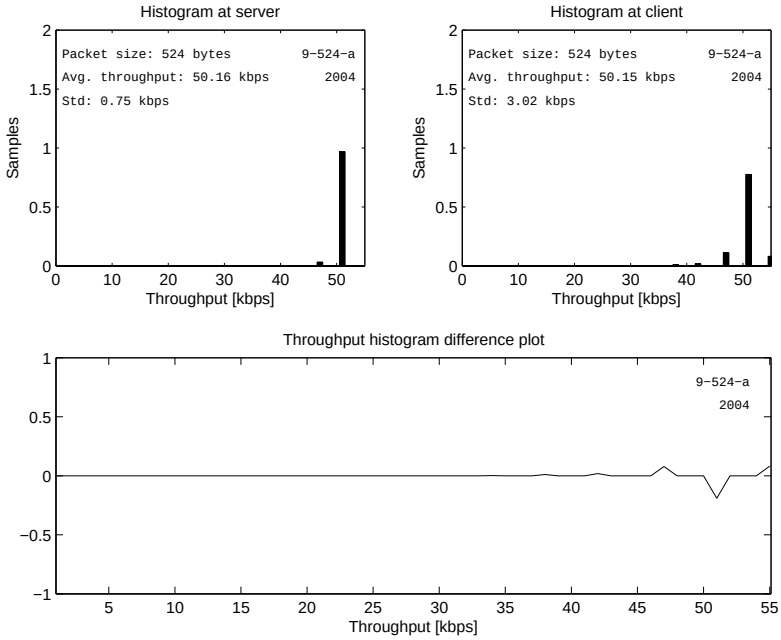



Figure 4.5: Throughput histograms and throughput histogram difference for a saturation factor of 0.9

In the throughput histograms in Figure 4.5 we see that the sender is sending in all traces almost all the time with the same speed, while the receiver has some problems with receiving the packets; the average of both the sender and the receiver is about the same, but the standard deviation is larger at the receiver side. It is difficult to say something about the autocorrelation plots (Figure 4.6), because these are all separate traces which are put behind each other. For the sender it is thus not clear if the periodicity which we see is caused by the fact that we put the traces behind each other or that there is a real periodicity in the traces, but we think that there is a real periodicity, because the throughput does not show a big standard deviation. But what we can say is that the receiver's behaviour of receiving is pretty scattered in all traces, and there is no periodicity at all.

If we also have a look at the joint traces for a saturation factor of 1.0 (Figure 3.3b) we see, compared to Figure 3.3a, that the delay is not showing the normal behaviour

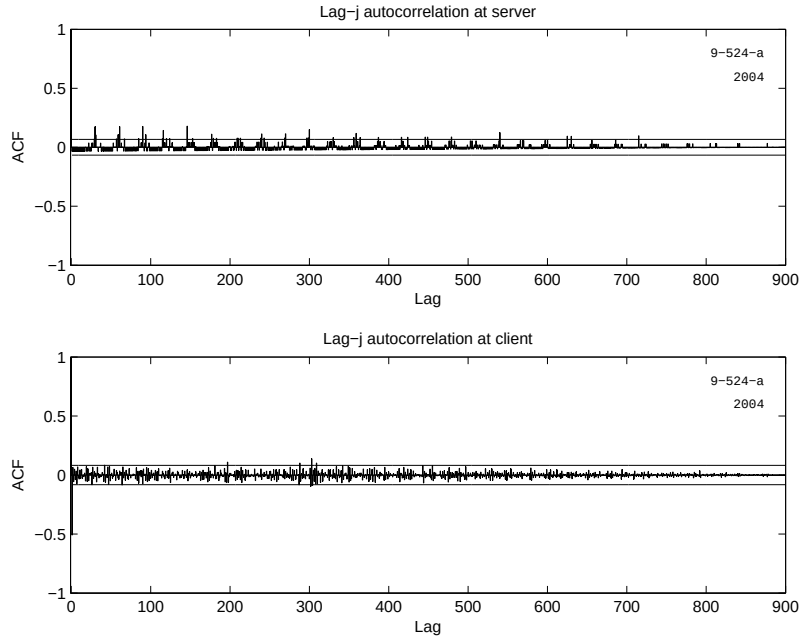



Figure 4.6: Autocorrelation plots for a saturation factor of 0.9

at all. In the delay histogram we see that the delay is really spread, the minimum delay is 229 ms whereas the maximum delay is 3134 ms, this is really a huge difference. When we look at the delay plot we see where this is coming from, for each trace the packets are queueing up and therefore, the delays are getting higher and higher and when we start with a new trace we get back to about the minimum delay and the same behaviour is showing again. Therefore, we decided to continue the joint analysis with a saturation factor smaller than 1.0.

## 4.2 MobiHealth long traces

When analysing the long traces obtained by the MobiHealth project in 2006, we discover that even if at the beginning of our analysis result seemed to be understandable, the data exposed some abnormalities and therefore no conclusions were drawn.

From Figure 4.7 one could say that the observed network behaviour is as expected. The average throughput is the same on sender and receiver side, while the standard deviation grew significantly. This is also seen from the histogram itself, the sender sends with the same speed (almost) all the time, while the receiver has some more variation. It is obvious that the burstiness of the packet stream has increased, which is shown by the shape of the throughput histogram difference plot. This behaviour is a good example for a shared bottleneck [7]; at the input, the throughput is constant, at the output the throughput is variable and the throughput histogram difference plot has a “M” shape. Furthermore the

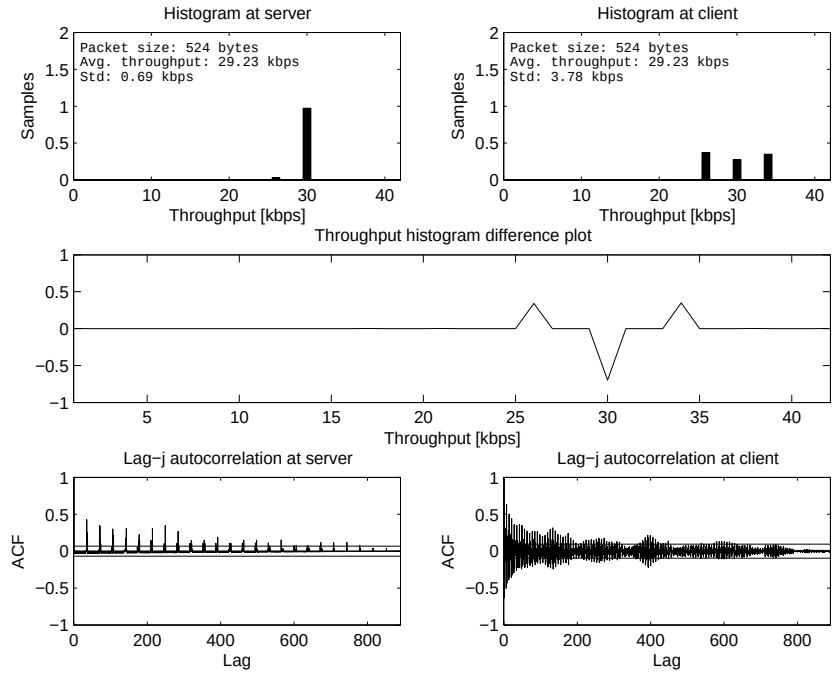



Figure 4.7: The MobiHealth long trace: throughput histograms, difference plot and autocorrelation plot

autocorrelation plot (bottom left) of the sender shows a periodicity (of about 32 seconds), while at the receiver side (bottom right) there is no periodicity at all. While Figure 4.7 showed normal results for the long traces of the MobiHealth project, Figure 4.8 on the other hand shows that these traces expose abnormalities and therefore they are not useful for answering the research question. This is because there have been some problems with the time synchronization of the nodes participating in the measurements. In the Figure 4.8 one can see that packets arrived at the receiver before they were send, there is a negative delay, from (about) sample 1450, so about after 24 minutes since measurements started.

Summarizing our analysis, we conclude that unfortunately this long trace is useless for our analysis of the relationship between one-way delay and throughput, hence further in this thesis we will focus on the analysis of the short traces obtained in the MobiHealth project.

## 4.3 PIITSA traces

The PIITSA traces provided us with traces in both uplink and downlink direction and therefore we will have a look at both; we start the analysis of these traces by looking at the downlink traces followed by the uplink traces.

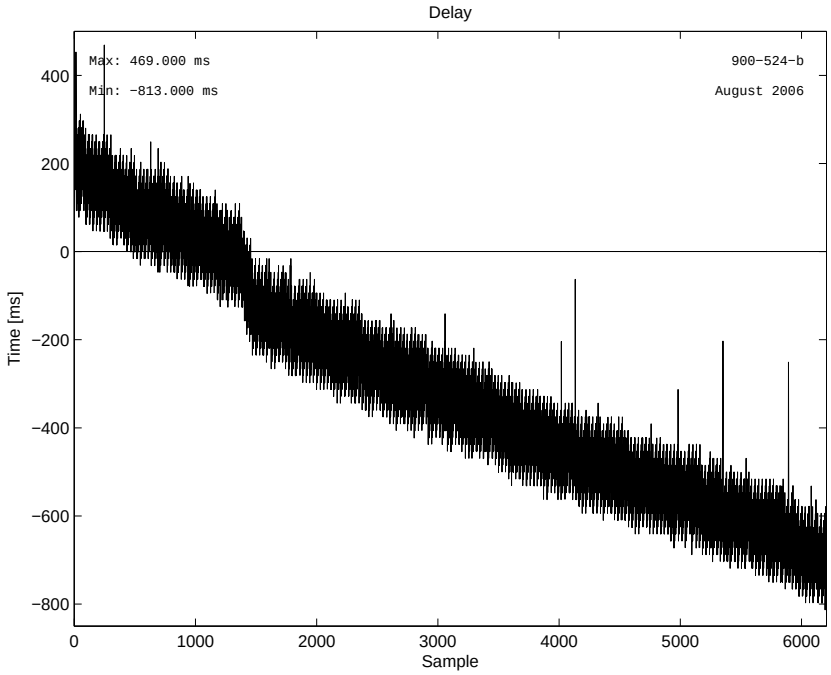



Figure 4.8: The one-way delay plot for the long traces of the MobiHealth project

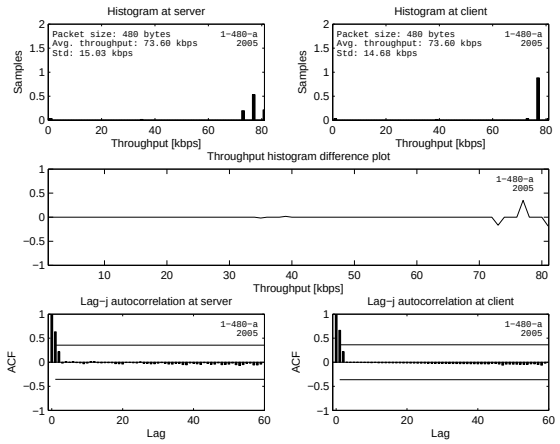
For the PIITSA traces we have the same problem as we have with the long trace of the MobiHealth project. Namely, when we look at the throughput plots (Figure 4.9a) we do not see that in this research project no time synchronization has been used, this problem is only seen by looking at the delay plot (Figure 4.9b).

Starting with Figure 4.9a we see in the throughput histogram difference plot that the network is showing a typical shaping bottleneck [7, 20]. A shaping bottleneck can be recognized by the 'W' shape; the sender is sending the packets with varying speeds in the throughput, whereas the receiver is receiving it almost with one speed.

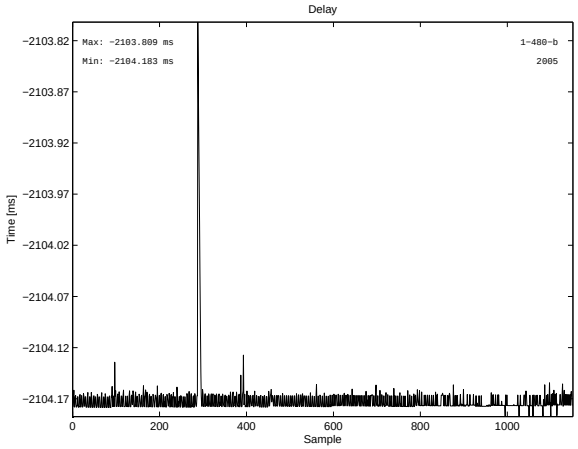
In the delay plot we see that most of the delays are around 2103 ms, but we can also see that these delays all have a negative value. It is a pity that we can not say exactly what the clock difference is between the sender and the receiver, even if it looks like that the receiver is about 2.1 seconds behind.

We made the same plots (Figure 4.10a and 4.10b) for sending data in the uplink direction. Here we see instead of a shaping bottleneck a sharing bottleneck; the sender is sending with one speed while the receiver is receiving it with variations in the throughput. In the delay plot we see the same behaviour as in the downlink direction, the only difference is that here we even have a bigger difference in the clock, namely about 4 seconds. And therefore, these PIITSA traces could not be used for the analysis on finding

the relationship between one-way delay and throughput.

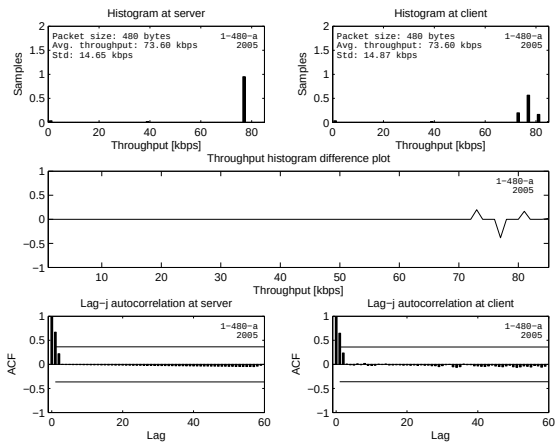



(a) Throughput histograms, throughput histogram difference and autocorrelation plots

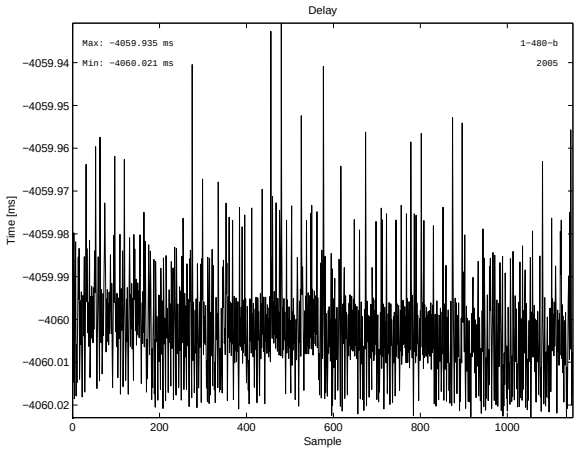



(b) Delay plot

Figure 4.9: PIITSA traces for UMTS downlink with an IPD of 50 ms




(a) Throughput histograms, throughput histogram difference and autocorrelation plots




(b) Delay plot

Figure 4.10: PIITSA traces for UMTS uplink with an IPD of 50 ms

## 4.4 Traces with a traffic shaper

The analysis of these traces are again mainly focused on the traces for a saturation factor of 0.9, hence sending 12 packets per second. There were traces for the saturation factors 0.5–0.9, but we haven chosen to work with saturation factor 0.9 because here the IPD

is the smallest of all traces and thus it is nice to see with this IPD how the network is behaving. Furthermore, it is good to see how the traffic shaper is working, because we can compare it to the results from before.

To do the analysis for this section we have made three different kinds of figures. In Figure 4.11 the throughput for both sender and receiver, the throughput histogram difference and the autocorrelation plots are given, this to see the difference between the sender and the receiver. Here we see that the average throughput for both sender and receiver are equal, even the standard deviation is approximately the same. This is also shown by the throughput histogram difference plot, where we see a horizontal line which means that the throughput at the input equals the throughput at the output. And in the last two plots in Figure 4.11 we see that both the sender and the receiver exhibit periodicity in the behaviour. If these results are compared to the results of the MobiHealth project itself we can conclude that the traffic shaper is doing its job. We can conclude that because for the previous traces from the MobiHealth project, the results at the receiver side were different from the sender side, but now the results are the same.

In Figure 4.12 we see that both the sender and the receiver have no outliers in the throughput, which we also saw in Figure 4.11. The minimum throughput for both is 46.112 kbps, which is 11 packets per second, and the maximum throughput is 50.304 kbps, which corresponds with 12 packets per second. And because it is at both endpoints we can say that the fact that there were sent only 11 packets per second is not caused by the network, but it is because there were just 11 packets per second send. Furthermore, we see in the delay plot (the middle one in Figure 4.12) that there is one packet which is really delayed, this is packet number 11123 with a delay of 274 ms, but if we have a look at the corresponding interval we do not see that this delayed packet is influencing the throughput. Looking at the IPD plots we see that this packet has a high delay because the IPD is high too, and the reason that it has no influence on the throughput can also be found in the IPD plot from the receiver, because the first packet after the packet with the high delay has an IPD of 0 ms, hence it is arriving together with the previous packet. This is also for the packets with a higher IPD before, first a packet has a higher IPD than average and the packet behind it has an IPD that compensates the difference (between the high IPD and the average IPD) in a negative way, so both packets are still arriving with an average IPD. In the plot of the IPD for the sender we also see some high IPDs, the reason that we do not see this IPDs at the same packet at the receiver is because of the traffic shaper, the traffic shaper controlled the packets in such a way that when the packets leave the traffic shaper the packets continue its way to the receiver with a stable interval and thus the outliers at the receiver are caused by the network between the traffic shaper and the moment that the packets are received by the application layer.

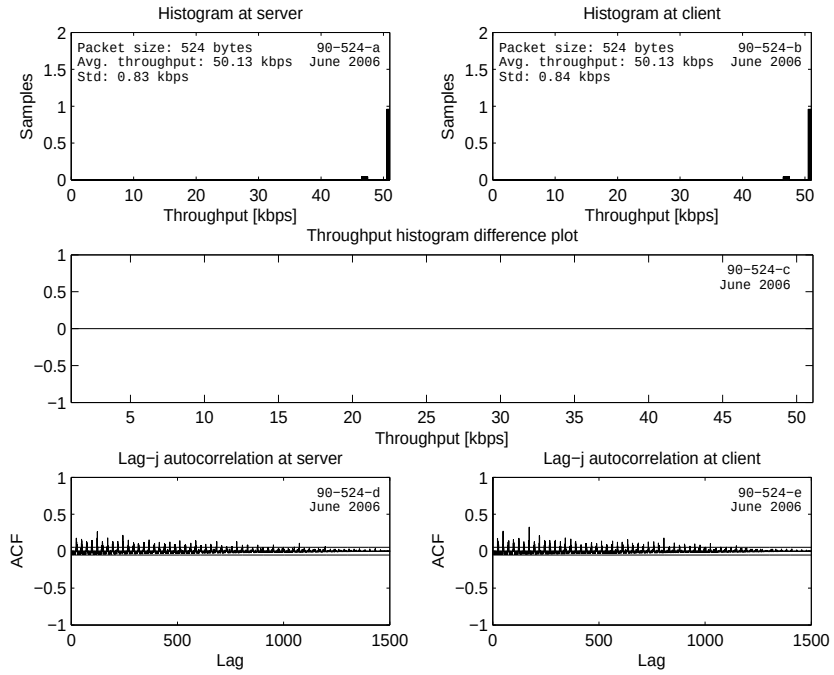



Figure 4.11: Throughput histograms, throughput histogram difference and autocorrelation plots of the traces with the traffic shaper

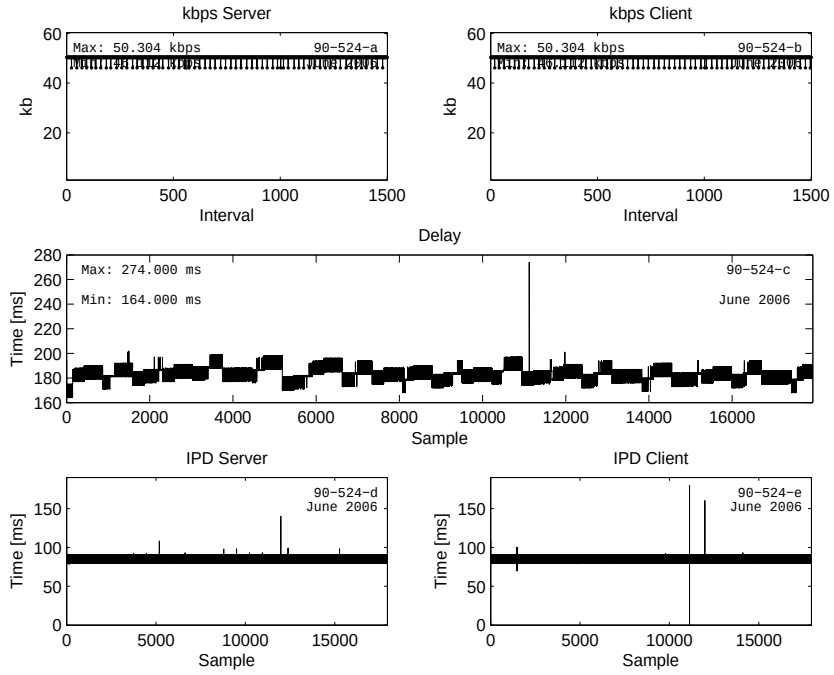



Figure 4.12: Throughput plots, delay plot and IPD plots of the traces with the traffic shaper

In Figure 4.13 we plotted the delay against the normal distribution to see if the delay has a normal distribution. The reason for this is that when we look at the shape of the delay histogram it seems to be a normal distribution and also because compared to the MobiHealth traces conducted in the Netherlands we see that with the traffic shaper the traces do not have a lot of outliers which makes it even more reasonable to think it is normal distributed. The parameters of the normal distribution are based on the delay data, hence the average of the delay is the $\mu$ in the normal distribution which is 183.23 ms and the $\sigma = 5.64$ ms.

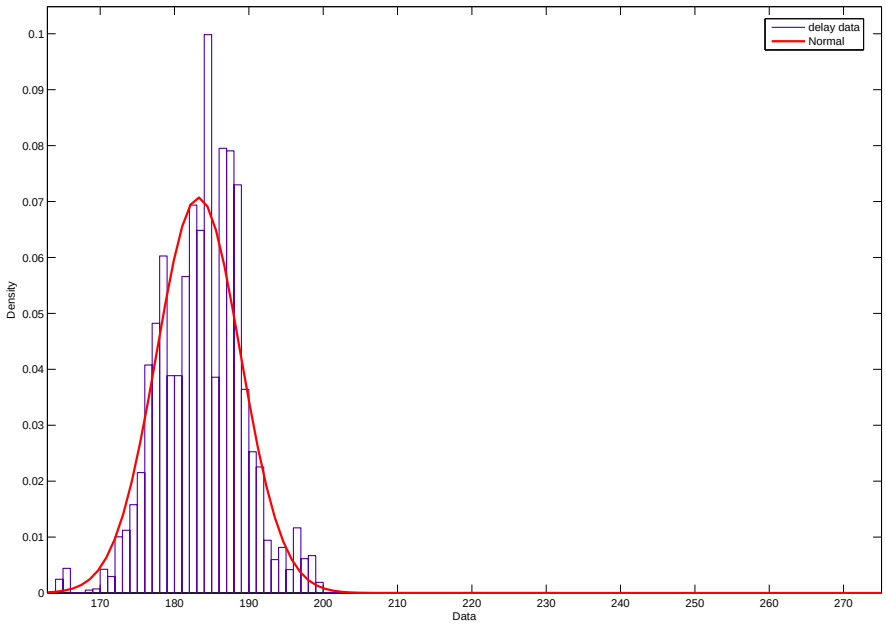



Figure 4.13: Delay histogram and Normal distribution

In the figure above we see that the delay does not show the normal behaviour as expected. This is especially the case for the values bigger than the average delay. Hence even if a traffic shaper is used the delay data is still not normally distributed.

## 4.5 Delay tail

In this section we attempt to find the relation of the tail in the delay histogram with the drift – the throughput difference at sender and receiver –, the throughput, and the throughput difference histogram for MobiHealth short traces for SAT 0.5. We base this analysis on a histogram of the inverse delay, such that we have the delay tail

at the left-hand side, which makes it easier to compare than to the right-hand side one. We did this for different time intervals $\Delta T$ over which we analyse the measurement data. More details about the different values of the time interval will be given in section 4.6.

The delay and throughput histograms are made with a Matlab script, written for the purpose of this thesis. For these histograms, we have the following definitions. To calculate the one-way delay ($d$) we subtracted the time stamp from the receiver $t^{out}$ of the one from the sender $t^{in}$:

$$d = t^{out} - t^{in}. \tag{4.1}$$

Where *out* stands for the receiver side and *in* for the sender side. To calculate the throughput the following formula (in Matlab) was used:

$$tp_i = \frac{\frac{a \cdot Packetsize \cdot 8}{1000}}{T}, \tag{4.2}$$

where $a$ is the number of packets in a certain interval $i$, the packet size is specified as 524 bytes, and $T$ is the time interval in seconds; for example $T = 0.25$ if we wanted a time interval of 250 ms.

The last thing we used is the drift. The drift is the difference in throughput between the sender and the receiver in a certain time interval $i$.

$$drift = tp_i^{out} - tp_i^{in}. \tag{4.3}$$

In Figure 4.14 the drift is not showing any tail, but the throughput histogram difference is. Furthermore, this figure is showing a small tendency to the lower values (seen from the average throughput).

The fact that the smaller time interval (IPD 83 ms, 1 packet each interval) is not showing a tail, is because when one packet arrives, the average throughput of 50 kbps is reached, but when no packet at all is arriving, there is no throughput. But there can be more than one packet in an interval, resulting in a tail towards the higher values. It is normal that if there would be a tail, that the tail would be towards the side of the higher values, because a negative throughput is not possible. For a time interval of 250 ms this is also the case, because intervals with no packets exists.

Since some plots in Figure 4.14 show a similar shape as the inverse of the delay, we will continue in the following sections with the inverse of the delay.

## 4.6 Variation of $\Delta$T

In this section, in our analysis we will vary $\Delta T$, the time interval over which we analyse the data, for the MobiHealth short traces for the saturation factor 0.9 (12 packets a

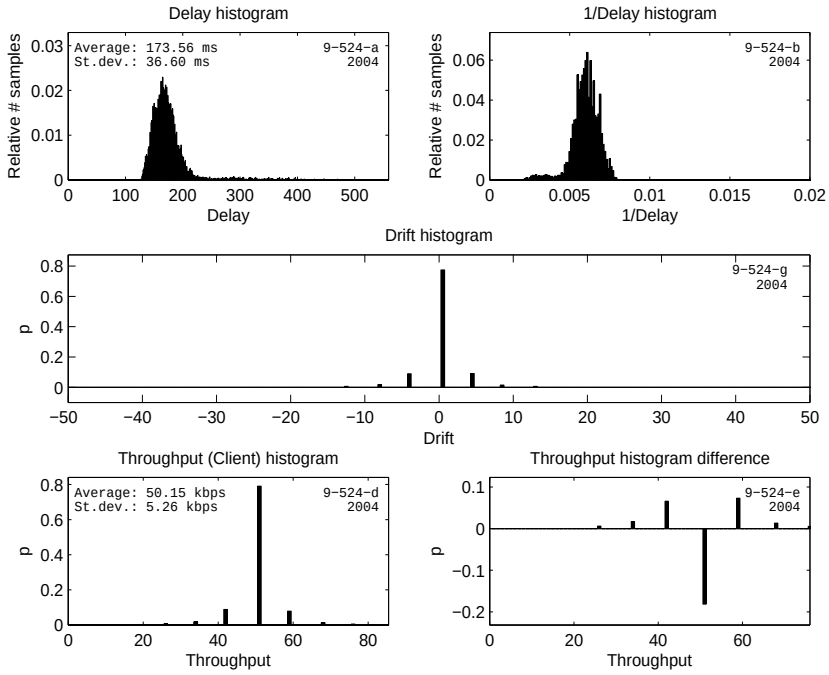



Figure 4.14: Delay, inverse delay, drift, throughput and throughput difference histograms for a time interval of ΔT 500 ms

second). We do it in order to have a better view on where exactly in the time interval the packets arrive at the receiver. With varying the $\Delta T$ the number of packets per interval will be different too. Because there are 12 packets a second, we selected $\Delta T$s of 83 ms (1 packet in an interval), 250 ms (3 packets), 500 ms (6 packets) and 1 second (12 packets). With changing the $\Delta T$, we could analyse the bandwidth (its maximum and minimum) and the maximum and minimum IPD in a certain interval. Selected results are presented in Tables 4.1–4.6, where each row corresponds to an observation interval. Based on these results, we attempt to find the relationship between the one-way delay and the throughput.

Already from Table 4.1, for ΔT of 1 second, it is clear that a high delay does not automatically mean that there is a change in the throughput. For example, the fifth interval presented in the table, there is a maximum delay of 429 ms, while the throughput is what is expected: 50.203 kbps. This is because even if there is a maximum delay of 429 ms, still 12 packets arrived in this interval. Furthermore, in interval 9 there is a significantly lower throughput (37.728 kbps), it is because there are less packets received in this interval. When we look at the next interval we see that there is a maximum delay of 446 ms, so the assumption is that that packet should have arrived in the previous interval, but because of the high delay it came into the next interval and it pushed all the packets behind it, into the next interval. And because the maximum delay in the fifth interval has no influence on the throughput, the packet with this delay is probably somewhere at the beginning or half-way of the interval. To see if this assumption is true,

we look at the data, assuming smaller $\Delta T$s, for which it may become clear if this is the case.

In Table 4.2, for $\Delta T$ 500 ms, we still see that in the maximum delay of 429 ms in interval 3, which has no influence on the number of packets in the interval and thus on the throughput, so the assumption from previous analysis holds. This packet is definitely not at the end of the interval, otherwise there would be probably some changes in throughput with this $\Delta T$. Furthermore, the packet with the high delay is also causing the problem for the change in throughput with this $\Delta T$.

In Table 4.3, where $\Delta T$ is 250 ms, we see the first time influence of the delay of 429 ms on the throughput. On the other hand, it is also clear that it has not a big influence, compared to the ninth and tenth interval where a whole interval is skipped, caused by the packet with the delay of 446 ms.

The best overview is presented in Table 4.4, for a $\Delta T$ of 83 ms. It is clear that the packet with 446 ms is at the end of the interval of 1 second and falls, because of the high delay, into the next interval (Table 4.4 and 4.5). This packet with a high one-way delay is pushing the two packets behind it, in the next second. It results in the situation, where in one interval, only 9 packets arrive whereas in the next interval the remaining three packets will arrive, and hence 15 packets in total arrive.

Hence from these tables it is clear that a high one-way delay does not mean that there is a change in the throughput. The question on the relationship between the one-way delay and throughput is still valid.

As stated before, the relationship between the delay and the throughput is not just based on a high delay. When going back to the analysis of the delay tail we pointed out that it is better to work with the inverse of the delay, because of the fact that the drift histogram, the throughput histogram, and the throughput difference histogram – and then mainly the last one – were showing a small tendency to the left, hence a left-hand sided tail for the delay would be better to work with then with a right-hand sided one. Therefore the analysis will be continued with the inverse of the delay in the next section.

### 4.6.1 Additional delay

In the previous section we saw that it is better to compare the drift, throughput and throughput difference histogram with the inverse of the delay. In the literature [21, 22], the writers were talking about "potential delay", which they also derive with the inverse delay, hence we come to the idea that the "additional delay" is based on the drift

$$d_{add} = \frac{1}{throughput\text{“}change\text{”}} \tag{4.4}$$

The $d_{add}$ is the additional delay besides the network-transport related one-way delay

(theoretical), observed (measured) per packet in the observation interval and resulting from the network behaviour variations. With the throughput "change" we mean the difference between the throughput ($tp_m$)with what we get out of the network (measured, practical) and the throughput ($tp_t$) which is expected to come out of the network (calculated, theoretical). The throughput is calculated by $tp = \frac{L}{\Delta T} \cdot a$, where $L$ is the packet size, $\Delta T$ is the time interval in milliseconds and # packets is the number of packets in the given time interval. For a saturation factor of 0.9 and a time interval of 1000 ms, we would have an expected throughput of $\frac{4192}{1000} \cdot 12 = 50.304$ kbps.

Before getting into the analysis of the additional delay, we will give some examples to illustrate it.

**Example 1**

The following is known:

$\Delta T = 1000$ ms $\rightarrow$ 12 packets a second

L = 4192 bits

In the first interval 9 packets arrive and in the second interval 15 packets arrive.

This will give us the following for 12 packets a second:

$tp_t = \frac{4192}{1000} \cdot 12 = 50.3040$ kbps

$\frac{1}{tp_t} = \frac{1}{50.3040} = 0.019879$ s/kb

First interval:

$tp_m = \frac{4192}{1000} \cdot 9 = 37.728$ kbps

$\frac{1}{tp_m} = \frac{1}{37.728} = 0.026506$ s/kb

$\rightarrow d_{add} = tp_m - tp_t = 0.26506 - 0.019879 = 0.00626$ s/kb

This gives the following additional delays for each packet in the interval:

$1^{st}$ : $0.006626 \cdot 1 \cdot 4192 = 27.778$ ms

$2^{nd}$ : $0.006626 \cdot 2 \cdot 4192 = 55.556$ ms

$3^{rd}$ : $0.006626 \cdot 3 \cdot 4192 = 83.334$ ms

$\vdots$

$9^{th}$ : $0.006626 \cdot 9 \cdot 4192 = 250$ ms

Second interval:

$tp_m = \frac{4192}{1000} \cdot 15 = 62.8800$ kbps

$\frac{1}{tp_m} = \frac{1}{62.8800} = 0.015903$ s/kb

$\rightarrow d_{add} = tp_m - tp_t = 0.015903 - 0.019879 = -0.003976$ s/kb

| kbps | number of packets | maximum delay | minimum delay |
|---|---|---|---|
| 50.304 | 12 | 196 | 151 |
| 50.304 | 12 | 167 | 139 |
| 50.304 | 12 | 179 | 146 |
| 46.112 | 11 | 194 | 148 |
| 50.304 | 12 | 429 | 147 |
| 50.304 | 12 | 178 | 145 |
| 50.304 | 12 | 205 | 162 |
| 50.304 | 12 | 364 | 147 |
| 37.728 | 9 | 194 | 149 |
| 62.880 | 15 | 446 | 148 |
| 50.304 | 12 | 198 | 147 |
| 50.304 | 12 | 196 | 147 |
| 50.304 | 12 | 277 | 152 |
| 50.304 | 12 | 180 | 150 |

Table 4.1: Arbitrary samples for the time interval of $\Delta T$ 1 second

| kbps | number of packets | maximum delay | minimum delay |
|---|---|---|---|
| 50.304 | 6 | 194 | 162 |
| 50.304 | 6 | 164 | 147 |
| 50.304 | 6 | 429 | 160 |
| 50.304 | 6 | 178 | 161 |
| 50.304 | 6 | 175 | 145 |
| 50.304 | 6 | 205 | 162 |
| 50.304 | 6 | 194 | 174 |
| 50.304 | 6 | 364 | 162 |
| 50.304 | 6 | 165 | 147 |
| 50.304 | 6 | 194 | 160 |
| 25.152 | 3 | 163 | 149 |
| 75.456 | 9 | 446 | 150 |
| 50.304 | 6 | 164 | 148 |
| 50.304 | 6 | 198 | 147 |

Table 4.2: Samples for the time interval of $\Delta T$ 500 millisecond

| kbps | number of packets | maximum delay | minimum delay |
|---|---|---|---|
| 16.768 | 1 | 160 | 160 |
| 83.840 | 5 | 429 | 177 |
| 50.304 | 3 | 194 | 176 |
| 50.304 | 3 | 178 | 174 |
| 16.768 | 1 | 162 | 162 |
| 83.840 | 5 | 364 | 163 |
| 50.304 | 3 | 166 | 162 |
| 50.304 | 3 | 163 | 149 |
| 0 | 0 | 0 | 0 |
| 100.61 | 6 | 446 | 167 |
| 50.304 | 3 | 164 | 150 |
| 50.304 | 3 | 162 | 148 |

Table 4.3: Random samples – the 429 ms and 446 ms included – for the time interval of 250 milliseconds

| kbps | number of packets | maximum delay | minimum delay |
|---|---|---|---|
| 100.61 | 2 | 161 | 149 |
| 50.304 | 1 | 147 | 147 |
| 0 | 0 | 0 | 0 |
| 50.304 | 1 | 161 | 161 |
| 50.304 | 1 | 164 | 164 |
| 50.304 | 1 | 162 | 162 |
| 50.304 | 1 | 160 | 160 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 201.22 | 4 | 429 | 195 |
| 50.304 | 1 | 177 | 177 |

Table 4.4: Sample – with the 429 ms of the 1 sec time interval – for the time interval of 83 millisecond

| kbps | number of packets | maximum delay | minimum delay |
|---|---|---|---|
| 50.304 | 1 | 160 | 160 |
| 50.304 | 1 | 180 | 180 |
| 50.304 | 1 | 194 | 194 |
| 50.304 | 1 | 166 | 166 |
| 50.304 | 1 | 164 | 164 |
| 50.304 | 1 | 162 | 162 |
| 50.304 | 1 | 149 | 149 |
| 50.304 | 1 | 163 | 163 |
| 50.304 | 1 | 161 | 161 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Table 4.5: Sample – before the 446 ms of the 1 sec time interval – for the time interval of 83 millisecond

| kbps | number of packets | maximum delay | minimum delay |
|---|---|---|---|
| 201.22 | 4 | 446 | 212 |
| 50.304 | 1 | 179 | 179 |
| 50.304 | 1 | 167 | 167 |
| 50.304 | 1 | 164 | 164 |
| 50.304 | 1 | 162 | 162 |
| 50.304 | 1 | 150 | 150 |
| 50.304 | 1 | 148 | 148 |
| 50.304 | 1 | 162 | 162 |
| 50.304 | 1 | 150 | 150 |
| 50.304 | 1 | 164 | 164 |
| 50.304 | 1 | 162 | 162 |
| 50.304 | 1 | 149 | 149 |

Table 4.6: Sample – with the 446 ms of the 1 sec time interval – for the time interval of 83 millisecond

This gives the following additional delays for each packet in the interval:

$1^{st}$ : $-0.003976 \cdot 1 \cdot 4192 = -16.67$ ms

$2^{nd}$ : $-0.003976 \cdot 2 \cdot 4192 = -33.33$ ms

$3^{rd}$ : $-0.003976 \cdot 3 \cdot 4192 = -50$ ms

$\vdots$

$15^{th}$ : $-0.003976 \cdot 15 \cdot 4192 = -250$ ms

**Example 2**

The following is known:

$\Delta T = 500$ ms $\rightarrow$ 6 packets a second

L = 4192 bits

In the first interval 3 packets arrive and in the second interval 9 packets arrive.

This will give us the following for 6 packets a second:

$tp_t = \frac{4192}{500} \cdot 6 = 50.3040$ kbps

$\frac{1}{tp_t} = \frac{1}{50.3040} = 0.019879$ s/kb

First interval:

$tp_m = \frac{4192}{500} \cdot 3 = 25.1520$ kbps

$\frac{1}{tp_m} = \frac{1}{25.1520} = 0.03976$ s/kb

$\rightarrow d_{add} = tp_m - tp_t = 0.03976 - 0.019879 = 0.019879$ s/kb

This gives the following additional delays for each packet in the interval:

$1^{st}$ : $0.019879 \cdot 1 \cdot 4192 = 83.33$ ms

$2^{nd}$ : $0.019879 \cdot 2 \cdot 4192 = 166.67$ ms

$3^{rd}$ : $0.019879 \cdot 3 \cdot 4192 = 250$ ms

Second interval:

$tp_m = \frac{4192}{500} \cdot 9 = 75.4560$ kbps

$\frac{1}{tp_m} = \frac{1}{75.4560} = 0.01333$ s/kb

$\rightarrow d_{add} = tp_m - tp_t = 0.01333 - 0.019879 = -0.00667$ s/kb

This gives the following additional delays for each packet in the interval:

$1^{st}$ : $-0.00667 \cdot 1 \cdot 4192 = -27.78$ ms

$2^{nd}$ : $-0.00667 \cdot 2 \cdot 4192 = -55.56$ ms

$3^{rd}$ : $-0.00667 \cdot 3 \cdot 4192 = -83.34$ ms

$\vdots$

$9^{th}$ : $-0.00667 \cdot 9 \cdot 4192 = -250$ ms

**Example 3**

The following is known:

$\Delta T = 250$ ms $\rightarrow$ 3 packets a second

L = 4192 bits

In the first interval 0 packets arrive and in the second interval 6 packets arrive.

This will give us the following for 3 packets a second:

$tp_t = \frac{4192}{250} \cdot 3 = 50.3040$ kbps

$\frac{1}{tp_t} = \frac{1}{50.3040} = 0.019879$ s/kb

First interval:

No packets arrived in this interval, hence a whole interval of 250 ms is skipped. Which means that the packet has an additional delay of at least 250 ms.

Second interval:

$tp_m = \frac{4192}{250} \cdot 6 = 100.6080$ kbps

$\frac{1}{tp_m} = \frac{1}{100.6080} = 0.0099$ s/kb

$\rightarrow d_{add} = tp_m - tp_t = 0.0099 - 0.019879 = -0.0099$ s/kb

This gives the following additional delays for each packet in the interval:

$1^{st} : -0.0099 \cdot 1 \cdot 4192 = -41.67$ ms

$2^{nd} : -0.0099 \cdot 2 \cdot 4192 = -83.33$ ms

$3^{rd} : -0.0099 \cdot 3 \cdot 4192 = -125$ ms

$\vdots$

$6^{th} : -0.0099 \cdot 6 \cdot 4192 = -250$ ms

Hence, we conclude from the following examples that if the additional delay is about 250 ms $\pm$ 10 ms; the average delay of the network is about 180 ms $\pm$ 10 ms. If we add those two, we get the delay of around 446 ms. It is not exactly the value we indicated before because of the $\pm$ 10 ms clock resolution we have in the traces of the MobiHealth project.

In the Figure 4.15, all the 'additional' delays between two consecutive time intervals are illustrated for the different time intervals $\Delta T$. The black dotted line indicates sending 12 packets per second per time interval of 1000 ms. But as we see in this figure only 9 packets arrived in the first interval whereas 15 packets in the next interval, hence the additional delay of the ninth packet in the first interval is 250 ms. The green line gives a better view of the additional delay of 250 ms, because here we have a time interval of 250 ms and hence the 3 packets which should arrive in the first interval all arrived in the

second interval together with the other 3 packets.

It is clear that most of the additional delays, which are reached in an interval are solved in the next interval, the $d_{add}$ value is always getting back to zero. But this is not always the case, for example, when packets are lost and TCP needs to transmit, then it might be that in a certain interval, as illustrated by the red dashed line in Figure 4.15, 8 packets are received and in the next interval 15 packets (with in mind that we still work with 12 packets a second). It is clear, that in that case we expected 16 packets per second, hence one packet is missing in the interval of 15 packets. Several problems could come from this, one of them is that for example only 11 packets were sent, so if you send one packet less, you also receive one packet less. The other reason is shown in the figure (red line). One of the packets arrived in the interval before and hence the additional delay starts with a negative value. But after the three intervals, packet rate is back to normal. The reason for this is that already at the start of this MobiHealth short trace something was not aligned. We saw that in the first interval only 12 packets were sent, but 13 packets arrived, which is not possible. This could be explained by of a small synchronization error, and it could be analysed with, for example, the equivalent bottleneck method which is described in [7].

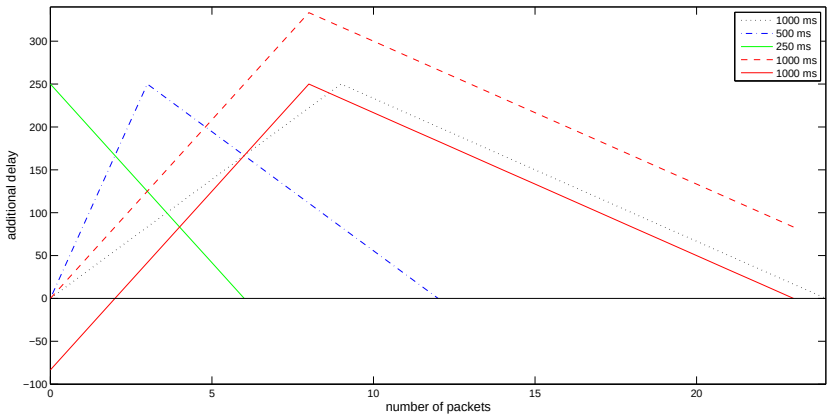



Figure 4.15: Additional delay per time interval

The analysed additional delay is also shown in Figure 4.16. The black arrows mark the arrivals of the packets as supposed for the network behaving correctly. The blue arrows shows the arrivals of the packets with the additional delay. As one can see, the ninth packet was supposed to arrive around 747 ms, but it arrived just before the second past. The 250 ms of additional delay is also illustrated here.

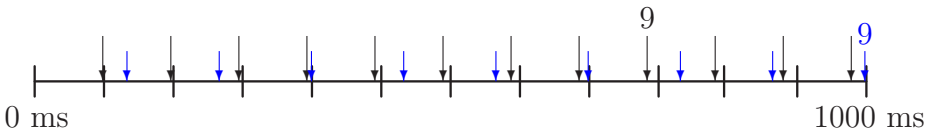



Figure 4.16: Arrival of packets in a time interval of 1 second

With the above analysis we came to the following relationships:
Throughput in interval $i = tp_i = \frac{L}{\Delta T} \cdot a$, where
$L$ is the packet size in bits
$\Delta T$ is the time interval in milliseconds
$a^{out}$ is the number of packets received in the time interval $i$

$$D_{add} \approx \left( \left( tp_i^{out} \right)^{-1} - \left( tp_i^{in} \right)^{-1} \right) \cdot L \cdot a^{out} \tag{4.5}$$

This formula can be simplified to:

$$D_{add} \approx \Delta T - \frac{\Delta T \cdot a^{out}}{a^{in}} \tag{4.6}$$

With this formula, we succeed to find the relationship between the one-way delay and throughput in a UMTS network.

### 4.6.2 Validation

In the previous sections it became clear that the traces of the PIITSA project could not be used for the validation, for the reason that in this project no time synchronization is used and thus we can not rely on the measured one-way delay values between the sender and the receiver. The traces of the cooperation between the PIITSA and MobiHealth project also could not be used, because the traffic shaper controlled the traffic perfectly and thus there are no time intervals where the packets are delayed and thus pushed into the next interval. Therefore the validation of our formula for the one-way delay throughput relationship will be done using selected MobiHealth measurement data with saturation factors different than for which this relationship has been derived. The measurement data for saturation factors which are lower than the SAT 0.9 (used before) will be used for validation purposes. This is because for saturation factors bigger than 0.9 we deal with a stressed network behaviour, which is not really representative as a network behaviour.

For validation purposes, we will use data for saturation factor 0.7, which means 9 packets per second. For this saturation factor, we will first show when using the formula of the additional delay, what will happen according to us when certain numbers of packets arrive, and after that, we will verify these results with what we see from the measurement traces.

For a saturation factor of 0.7 the sender will send 9 packets per second and thus for most of the time it is expected that 9 packets arrive at the receiver, but this could, as we saw in the traces of saturation factor 0.9, also be another numbers of packets. In a certain interval we could receive 5–13 packets, and the less packets we receive the higher the additional delay is, but of course it is also assumable that this will happen often. We will use simplified formula (Formula 4.6) to illustrate what we will expect as the additional delays and the total delays, where we take $\Delta T = 1000$ ms as the time interval, $a^{in}$ equals 9 and $a^{out}$ is the expected number of packets which arrive at the receiver.

Additional delay for 5 packets in an interval:
$1000 - \frac{1000 \cdot 5}{9} = 444.44$ ms
Additional delay for 6 packets in an interval:
$1000 - \frac{1000 \cdot 6}{9} = 333.33$ ms
Additional delay for 7 packets in an interval:
$1000 - \frac{1000 \cdot 7}{9} = 222.22$ ms
Additional delay for 8 packets in an interval:
$1000 - \frac{1000 \cdot 8}{9} = 111.11$ ms

For more than 9 packets in an interval we get the negative values for the additional delay from the above calculations. Knowing that we should get these additional delays, we will try to get the delays for the same intervals. For saturation factor 0.9 (12 packets per second) we had an average delay of about 180 ms, now we are sending only 9 packets in each time interval, hence the average delay is expected to be lower than this but because of the accuracy of the time stamps we take the one-way delay of 180 ms as an upper bound. Thus we would get a maximum delay of 620 ms, 510 ms, 400 ms, and 290 ms for, respectively, 5, 6, 7, and 8 packets in an interval.

In Table 4.7 we present measurement data obtained from the measurement traces conducted in the MobiHealth project in 2004. From the table we see some samples with a high delay and the throughput for that interval. If the measured one-way delay and throughput values correspond to the ones, we calculated above, we can say that our model for the additional delay is correct and otherwise we have to find some reasons why it is different. Moreover the question raises if the model is valid for all the saturation factors and thus for all the measurements (time synchronized) which are done in a UMTS network, or only for the selected measurement data.

In this table we see just one interval with only 6 packets. So we can conclude that for this saturation factor the network does not have that many problems with sending and receiving the right number of packets

| kbps | number of packets | maximum delay | minimum delay |
|---|---|---|---|
| 37.728 | 9 | 181 | 146 |
| 25.152 | 6 | 161 | 134 |
| 50.304 | 12 | 415 | 133 |
| 37.728 | 9 | 163 | 133 |
| 37.728 | 9 | 432 | 137 |
| 37.728 | 9 | 181 | 136 |
| 37.728 | 9 | 165 | 138 |

Table 4.7: Samples for a saturation factor of 0.7

When we have a look at the table we see two intervals with a high delay, but what we also see is that, just like for saturation factor 0.9, a high delay does not mean a throughput much different than expected. For a saturation factor of 0.7 (9 packets per second) we expect a throughput of $\frac{4192}{1000} \cdot 9 = 37.728$, but in the table we see that the first interval with a delay above 400 ms we have a higher throughput, whereas in the second interval with a delay above 400 ms we get the expected throughput. As explained before, this is caused by the additional delay.

**Example for SAT 0.7**

The following is known:

$\Delta T = 1000$ ms $\rightarrow$ 9 packets a second

L = 4192 bits

In the first interval 6 packets arrive and in the second interval 12 packets arrive.

This will give us the following for 9 packets a second:

$tp_t = \frac{4192}{1000} \cdot 9 = 37.728$ kbps

$\frac{1}{tp_t} = \frac{1}{37.728} = 0.02651$ s/kb

First interval:

$tp_m = \frac{4192}{1000} \cdot 6 = 25.152$ kbps

$\frac{1}{tp_m} = \frac{1}{25.152} = 0.03976$ s/kb

$\rightarrow d_{add} = tp_m - tp_t = 0.03976 - 0.02651 = 0.01325$ s/kb

This gives the following additional delays for each packet in the interval:

$1^{st} : 0.01325 \cdot 1 \cdot 4192 = 55.56$ ms

$2^{nd} : 0.01325 \cdot 2 \cdot 4192 = 111.11$ ms

$3^{rd} : 0.01325 \cdot 3 \cdot 4192 = 166.67$ ms

$\vdots$

$6^{th}$ : $0.01325 \cdot 6 \cdot 4192 = 333.33$ ms

Second interval:

$tp_m = \frac{4192}{1000} \cdot 12 = 50.304$ kbps

$\frac{1}{tp_m} = \frac{1}{50.304} = 0.01987$ s/kb

$\rightarrow d_{add} = tp_m - tp_t = 0.019879 - 0.02651 = -0.0066$ s/kb

This gives the following additional delays for each packet in the interval:

$1^{st}$ : $-0.0066 \cdot 1 \cdot 4192 = -27.78$ ms

$2^{nd}$ : $-0.0066 \cdot 2 \cdot 4192 = -55.55$ ms

$3^{rd}$ : $-0.0066 \cdot 3 \cdot 4192 = -83.33$ ms

$\vdots$

$12^{th}$: $-0.0066 \cdot 12 \cdot 4192 = -333.33$ ms

Hence we get an additional delay of about 333 ms when 3 packets less are received than expected, but also for these traces we see that the additional delay of the first time interval (333.33 ms) is undone by the second interval (−333.33 ms). The average measured one-way delay is 157 ms, and as expected it is lower than the 180 ms which we get with sending 12 packets per second, hence we would get a total delay of 490 ms and with the accuracy of the time stamps we would get a range of [470;510] ms and again we see that the maximum total delay is what we expected for 6 packets in an interval.

Unfortunately, also for these MobiHealth traces there are some parts of the traces which are having problems with time synchronization. It is because, just like for the traces with the saturation factor of 0.9, there are parts of the traces where in the first interval more packets are arriving than that there are send, which make them unusable.

## 4.7 Additional delay method

In the last paragraphs we have introduced the additional delay method, because from the examples it was clear that we could not base the bottleneck analysis only on the delay values obtained from the measurement traces analysis.

The additional delay is the extra delay experienced by a packet transported over a network on top of the expected one-way network delay, and it is defined as follows:

$$D_{add} \approx \left( (tp_i^{out})^{-1} - (tp_i^{in})^{-1} \right) \cdot L \cdot a^{out}.$$

From this formula we see that we can base the additional delay on the network throughput changes (drift) and the number of packets which are getting out of the network in a given time interval $i$. So if we have a measurement trace from which these parameters van be

derived, we can identify a bottleneck in a network and its severity is based on the size of the additional delay; the bigger the change in the throughput the bigger the additional delay, the more severe is the network bottleneck condition.

The additional delay can also be defined as:

$$D_{add} \approx \Delta T - \frac{\Delta T \cdot a^{out}}{a^{in}}$$

In this definition we see that the additional delay is also based on the time interval ($\Delta T$) and the number of packets at the sender ($a^{in}$) and the receiver ($a^{out}$).

So if we look at this definition we could ask ourself what the best time interval is to find the bottleneck condition. From the previous analysis (showed in Figure 4.15) we see that with a time interval of 250 ms we automatically get the additional delay of 250 ms, which we also found with the other time intervals. Hence it is better to work with a small time interval, but do not take it too small, i.e., that only 1 packet should arrive in each interval, because that would probably not give a good overview of the additional delay. Our advice is to start with a time interval of 250 ms and based on those results what the next interval, larger or smaller, should be.

The best way to find clear bottlenecks is to work with the IPD. When a sender sends 10 packets per second, the packets have an IPD of 100 ms and when only 8 packets arrived at the receiver we see that 2 packets were delayed. Working with a time interval of the two packets, which are missing in the certain time interval at the receiver, and the IPD is best.

### 4.7.1 Algorithm

In the preceding section, we have the additional delay and how to choose the time interval explained. How to have a clear view of the bottleneck in the network we get to the algorithm below.

- Start with a quite large time interval ($\Delta T$), like 1 second or more, and calculate the drift ($drift_i = tp_i^{in} - tp_i^{out}$) time series for each interval $i$.
- As soon as there are drifts unequal to zero, we get an indication of the order of magnitude of the additional delays.
- Reduce $\Delta T$ from now on – now the output throughput ($tp_i^{out}$) will be interesting. When $tp_i^{out}$ reaches zero for a certain interval $i$, while $tp_i^{in}$ was not zero for the same interval, we find that the additional delay is at least equal to $\Delta T$

## 4.8 Summary

In this chapter, we started with the analysis of the traces from the MobiHealth project, where different loads were used to send data over the UMTS network and our analysis aimed in presentation of how the network was behaving for these different loads. We concluded that it is better for the application to use loads smaller than the ones corresponding to the behaviour for a fully saturated network (SAT 1.0; 14 packets per second). And from all the loads, smaller than the 14 packets per second, we have chosen the measurement data corresponding to the saturation factor with the largest number of packets send through the network without showing any problems, and with this data we continued the analysis with the traces obtained in the MobiHealth project.

Besides the short traces obtained in the MobiHealth project, we also had a long trace of about 15 minutes. However, this trace exposed some synchronization problems and hence it was not useful for the rest of our analysis.

Besides the traces from the MobiHealth project we also had a look at the traces obtained by the PIITSA project. Unfortunately no time synchronization was used in this research project, which could also be seen in the different plots of the PIITSA traces, thus these traces could not be used for the purpose of this thesis.

The PIITSA traces were followed by the analysis of the MobiHealth traces which were conducted in Sweden in the summer of 2006. The main difference between the MobiHealth traces conducted in Sweden and the ones conducted in the Netherlands in 2004 is the traffic shaper which was included in the measurement setup in Sweden as replacement for a life UMTS network segment. In the results from the traces it is obvious that the traffic shaper represents ideal constant network delay conditions, packets do not have delay variations as without the traffic shaper. Because of the fact that packets did not experienced jitter and they were not delayed, such that they did not arrived in other time intervals other than supposed to, these MobiHealth traces could also not be used for the purpose of this thesis, which is an analysis of the relationship between the one-way delay and the throughput. These traces were only useful in the sense, that they showed that when packets arrive in time, there is no additional delay.

After the analysis of all the traces, we showed that it is better to work with the inverse of the delay, instead of the delay itself. It is because of the fact that from the inverse delay we can get a left-hand sided delay tail which corresponds to the drift and throughput difference histograms, which also show a small tendency to the lower values.

At the end of our analysis, we have derived the relationship between the one-way delay and throughput, which is the additional delay. The additional delay is given by the following formula:

$$D_{add} \approx \left( (tp_i^{out})^{-1} - (tp_i^{in})^{-1} \right) \cdot L \cdot a^{out}$$

Where $tp_i^{out}$ and $tp_i^{in}$, respectively, are the throughput at the receiver and sender side. The packet size is defined by $L$ (in bytes) and $a^{out}$ is the number of packets at the receiver in time interval $i$.

For the additional delay it is shown how the additional delay is calculated and what the results are by showing the difference between arrivals expected to be observed for a 'normal' network behaviour and in a network with a presence of a jitter behaviour.

We have validated this relationship. Our intention was to validate it with the PI-ITSA traces, but unfortunately these traces could not be used for that because of the fact that the clocks at the sender and the receiver were not synchronized, hence the verification was done with MobiHealth data captured for another saturation factor(SAT 0.7), than the one for which the relationship has been derived (SAT 0.9). The validation shows that our model for the additional delay can capture delay bounds within particular accuracy.

And to identify the bottlenecks in the network we have introduced an algorithm. Where we first start with quite a large time interval and calculate the drift, if this drift is unequal to zero than we take smaller time intervals to get the bottleneck, which is at least the same as the time interval.

# CHAPTER 5

# Summary and Future Work

When starting this thesis a pool of research questions were raised to be answered. Unfortunately, not all the questions were managed to be answered in this thesis and thus these questions are a subject of future work.

## 5.1 Summary of methodologies comparison

The first part of this thesis consisted of the comparison of the methodologies used in the two research projects, MobiHealth and PIITSA. We started this part with some general background information, which is followed by the differences between the two projects.

1. Measurement setup. For the PIITSA project the sender was always sending the data over the wireless UMTS link and the receiver was receiving the data over the wired link. While in the MobiHealth project there was one computer which was sending and receiving the data over a wireless UMTS link and another computer was always connected to the wired University network. Furthermore, the MobiHealth project was using a third computer for the time synchronization issue.

2. Performance parameters of interest. The PIITSA project was interested in the throughput, whereas the MobiHealth was interested in the one-way delay and derived the throughput from that. Hence the two research projects measured a different performance parameter.

3. Workload parameter. Just one message size, 480 bytes for a UMTS network, has

been used in the PIITSA project. The MobiHealth project on the other hand used different message sizes, in total 400 combinations could be used in this project.

4. Transport protocols. UDP has been used by the PIITSA project, while TCP has been used by the MobiHealth project. The reason for this is that the MobiHealth project wanted to be sure that the data was received by the receiver, and this is a characteristic from the TCP transport protocol.

5. Software implementation. The PIITSA project chose to work with C# and the MobiHealth chose for Java.

6. Workload generation and the time stamping methods. The PIITSA project tried to send the UDP packets as regular as possible, this is done by sending the packets with a constant length and sequence number with a certain interval (IPD). This is to get a certain load on the network. Furthermore, the data flow has been time stamped on small time scales, typically 1 second. In the MobiHealth all the packets were time stamped and the packets were generated for each second. Hence in this project a specified number of packets were send per second, while in the PIITSA project the time interval between the packets was specified.

7. Time synchronization. It is because the MobiHealth wanted to measure the one-way delay and to have an one-way delay which is useful one need to synchronize the clocks at both the sender and the receiver in order that the times can be compared with each other. The PIITSA project was only interested in the throughput and thus no time synchronization was needed in the setup.

8. Output. Like more of the difference, also this difference is caused by the intention of the projects, this is because the PIITSA project was focused on the throughput and thus the results are also focused on the throughput. Whereas the MobiHealth project measured the one-way delay and derived the throughput from that, the results were focused on these performance parameters.

9. Initial delay vs the bearer changing discoveries. The PIITSA project discovered that the first packet can have a higher delay than the following packets. Whereas the MobiHealth discovered the bearer changing policy.

## 5.2 Summary of analysis

In the second part of this thesis the following research goal is answered.

> *To develop and evaluate a method for the identification of bottleneck conditions in a heterogeneous networking environment based on the relationships between measured one-way delay and throughput performance metrics.*

Summarizing the research conducted in the frame of this thesis, the relationship between the performance parameters one-way delay and throughput is captured by, what we called the *additional delay* ($D_{add}$). It is defined individually for each packet, and it reflects the extra delay on top of the expected observed one-way delay of a packet transported from a sender to a receiver in the heterogeneous networking environment. Therefore, the additional delay is defined as follows:

$$D_{add} = \left(\frac{1}{tp_i^{out}} - \frac{1}{tp_i^{in}}\right) \cdot L \cdot a^{out} = \Delta T - \frac{\Delta T \cdot a^{out}}{a^{in}}$$

Where $tp_i^{out}$ and $tp_i^{in}$, respectively, are the throughput at the receiver and sender side. The packet size is defined by $L$, $a^{out}$ ($a^{in}$) is the number of packets at the receiver (sender) and $\Delta T$ is the time interval.

To get the bottleneck an algorithm is introduced. This algorithm has a couple of steps, the first one is to calculate the drift for each time interval, where these intervals are quite large (i.e., one second or larger). When the drift is smaller or larger than zero, we have an indication of the order of magnitude of the additional delays. Finally by reducing the time interval the output throughput becomes interesting. Because when the output throughput reaches zero, while the input throughput is unequal to zero, we find the additional delay at least equal to the time interval.

From the analysis of the additional delay we can say that if the number of packets per second (or i.e., per $\Delta T$) at the receiver side is lower than the number of packets for the same second at the sender, we have a positive additional delay and thus we get a lower throughput in that relevant second, which may indicate the bottleneck. Analogously, we get a higher throughput for the time interval when we get more packets at the receiver side than that were sent in that interval. It is interesting to notice that in the first expression of the additional delay formula the relationship between the one-way delay and throughput is much more visible than in the second expression. It is because from the second part we just see that the additional delay is based on the time interval ($\Delta T$) and the number of packets in that defined time interval $\Delta T$, at both the sender and the receiver.

## 5.3 Future Work

As mentioned before, we had a lot of research questions which we wanted to answer in this thesis, but unfortunately this was not possible due to time constraints and therefore these questions become recommendations for future work.

One of the possible future work area is using measurement-based performance evaluation methodologies to facilitate the development of **self-organizing** mobile applications and services.

Moreover, of interest is also defining application-specific performance thresholds for issuing performance “alarms”.

Finally, developing a run-time end-to-end performance monitoring methodology and define API to a mobile application would be an ultimate goal of the research following the one performed for the purpose of this thesis.

# APPENDIX A

# Graphs for different saturation factors for the individual traces of the MobiHealth project

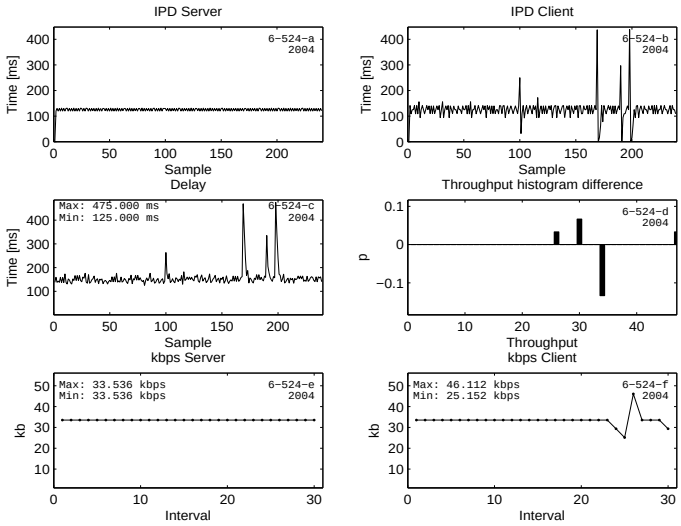



Figure A.1: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 0.6

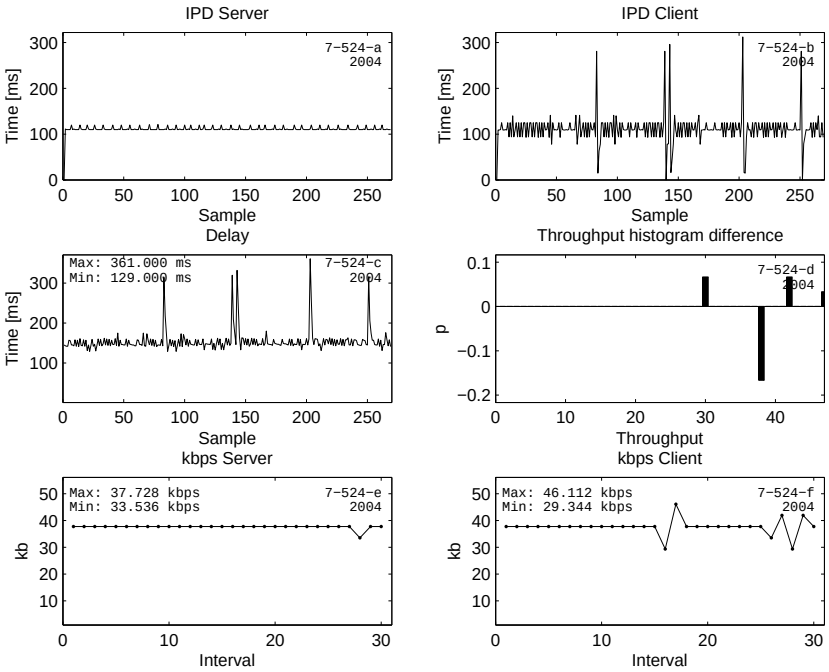



Figure A.2: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 0.7

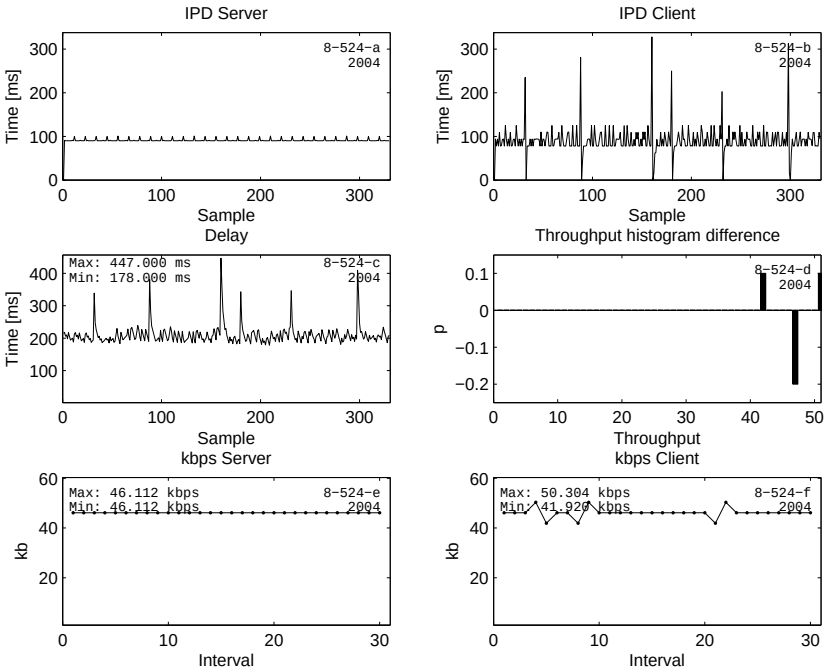



Figure A.3: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 0.8

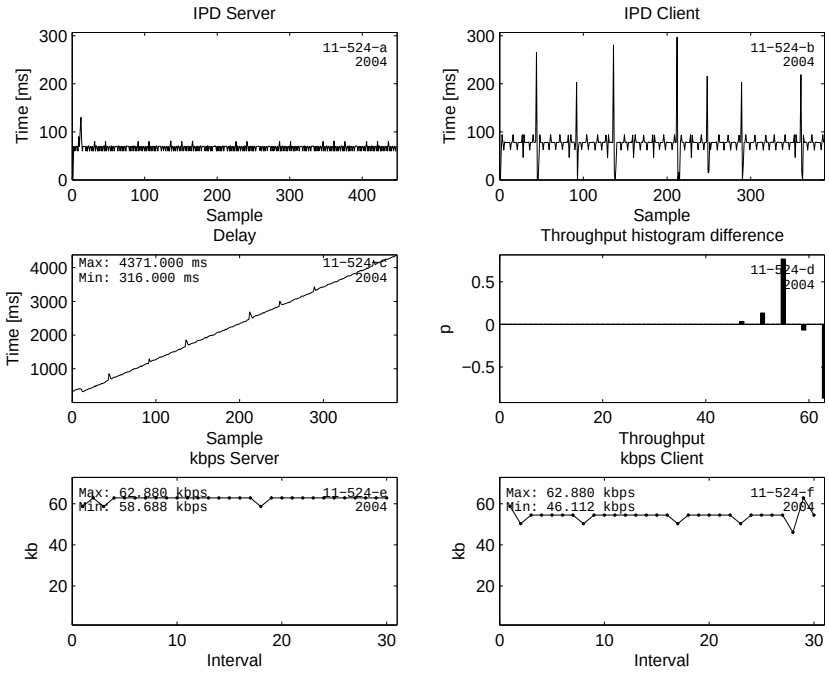



Figure A.4: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 1.1

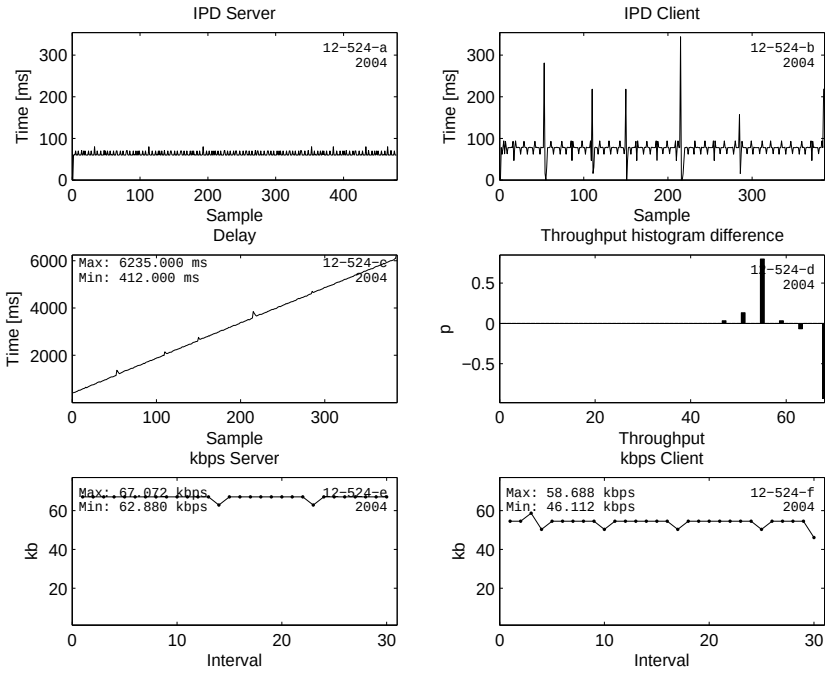



Figure A.5: IPD plots, delay plot, throughput difference histogram and bandwidth plots for saturation factor 1.2

# APPENDIX B

# Abbreviations

| | |
|---|---|
| BAN | Body Area Network |
| BEsys | Back-End system |
| BTH | Blekinge Tekniska Högskola |
| DoD | Department of Defense |
| GPRS | General Packet Radio Service |
| IP | Internet Protocol |
| IPD | Inter Packet Delay |
| ITS | Intelligent Transport Systems |
| kbps | One thousand twenty four bits per second (1024 bps) |
| MBU | Mobile Base Unit |
| NSB | Network Selection Box |
| NTP | Network Time Protocol |
| OS | Operating System |
| OSI | Open System Interconnection |
| PIITSA | Personal Information in Intelligent Transport systems through Seamless communication and Autonomous decisions |
| QoS | Quality of Service |
| RTT | Round Trip Time |
| SAT | Saturation factor |
| TCP | Transmission Control Protocol |
| TS | Time stamp |
| UDP | User Datagram Protocol |
| UMTS | Universal Mobile Telecommunications System |
| WLAN | Wireless Local Area Network |

# APPENDIX C

# Glossary

| | |
|---|---|
| Application-perceived throughput | Reflects the perspective of an application, i.e. captures the behaviour of all communication stacks in between a sender and a receiver |
| Goodput | Refers to the measurement of actual application data successfully transported from a sender to a receiver through the transport system |
| Initial delay | The time interval it takes before the first packet which is sent arrives at the receiver, usually this delay is more than an average one-way delay |
| Jitter | Variable part of the delay, due to queueing |
| kernel32.dll | OS library, that handles memory management, input/output operations, and interrupts |
| One-way delay | The time interval a packet takes to get from a sender to a receiver |
| QueryPerformance-Counter | C# mechanism that retrieves the current value of the high-resolution performance counter |

| | |
|---|---|
| QueryPerformance-Frequency | C# mechanism that retrieves the frequency of the high-resolution performance counter, if one exists |
| Round Trip Time | The time interval a packet takes to get from sender to receiver and back |
| Seamless communication | The situation in which the user is not aware of which wireless connection it is using; the connection will be changed when it is needed without notifying the user. |

# APPENDIX D

# Matlab program

The Matlab program, which is written to do the analysis, consists of five programs, but a function is written where one has to set the parameters to choose the right program for the analysis. This function will be illustrated in section D.1 and after that the main lines of the five programs will be given (section D.2)

## D.1 Function Choose Program

In this function one has to change the following parameters:

- The *project* parameter to choose the MobiHealth or the PIITSA project traces;
- The right *file* for the right trace;
  - for the short traces from the MobiHealth project this is SAT 0.5.txt until SAT 2.0.txt;
  - for the long trace from the MobiHealth project this is SAT 900.txt;
  - for the MobiHealth-PIITSA trace SAT 9.txt;
  - for the PIITSA project this is PIITSA.txt;
- The *rader* parameter;
- The time interval, which is defined as $T$.

```
function[]=Choose_program();

close all
clear all
clc;
start_x = 0.02;         % To put the text in the graphs
start_x2 = 0.97;
start_y = 0.9;
start_y2 = 0.95;
delta_1 = 0.1;          % The space between the text
delta_2 = 0.05;
w_delta = 1;
t1=[];
p1=[];
result_time1=[];        % Sender
result_number1=[];
result_time2=[];        % Receiver
result_number2=[];
resultkbps_server = [];
resultkbps_client = [];
result_ipd_server = [];
result_ipd_client = [];
packets_per_interval_client = [];
packets_per_interval_server = [];
delay = [];
somserver = 0;
somclient = 0;
n = 31;
format short;
warning off MATLAB:colon:operandsNotRealScalar;
warning off all;

FS = 10;   % Font size
FS2 = 8;

%#########################################################
% Choose project = 1 for the MobiHealt project
% Choose project = 2 for the PIITSA project
%#########################################################

project=2

if(project==1)
```

```
    [saturationfactor,sample,packetsize,start_sample,time_limit,
    pv13,pv14,pv23,pv24,pv33,pv34] = textread('SAT 1.0.txt','%d
    %d %d %d %d %d %d %d %d %d %d','headerlines',1);
else
    [saturationfactor,Link,packetsize,InterPacketDelay,start_sample,
    time_limit,pv13,pv14,pv23,pv24,pv33,pv34] = textread('PIITSA.txt'
    ,'%d %s %d %d %d %d %d %d %d %d %d %d','headerlines',1);
end

%#########################################################
% MobiHealth short traces
% Use different saturation factors: 0.5-2.0
% Choose rader = 1 to get the single traces
% Choose rader = 1:30 to get the whole trace

% MobiHealth long trace
% SAT 900.txt, because the trace is 900sec
% Choose rader = 1, because no repetitions

% MobiHealth_PIITSA trace
% SAT 9.txt
% Choose rader = 1, to get the first part of the trace
% Choose rader = 2, to get the second part of the trace

% PIITSA trace
% Choose rader = 1 to get the downlink trace with IPD 50ms
% Choose rader = 2 to get the downlink trace with IPD 60ms
% Choose rader = 3 to get the uplink trace with IPD 50ms
% Choose rader = 4 to get the uplink trace with IPD 60ms
%#########################################################

rader = [1];

%################################################
% T is the time interval
% Example (for SAT0.9):
% Choose T = 1 for 1 second
% Choose T = 0.5 for 500ms
% Choose T = 0.25 for 250ms
% Choose T = 0.083333 for 83ms
%################################################

T=1
```

```
if(saturationfactor==1) %1, because PIITSA uses full capacity
    disp('PIITSA trace')
    PIITSA
else if(saturationfactor==900) %900, because MH long trace is 900s
        disp('MH_long_trace')
        MH_long_trace
    else if(saturationfactor==90)
            disp('MH_PIITSA')
            MH_PIITSA
        else if(size(rader,2)==1);
                disp('MH_single_samples')
                MH_single_samples
            else
                disp('MH_all_samples_together')
                MH_all_samples_together
            end
        end
    end
end
```

## D.2 Programs for analysis

After setting the right parameters and running the function Choose Program, a certain program will start.

All of the five different programs are twofold; the first part of the program consists of selecting the data and handling the data and in the second part of the program the figures are made.

The main Matlab lines will be given in this thesis, to give an overview of what is happening in the programs.

```
Packetsize = packetsize(rad,1);
ss = start_sample(rad,1);
TL = time_limit(rad,1);
TL2 = time_limit(rad,1)*time_limit(rad,1);

s1 = strcat(num2str(saturationfactor(rad,1)),'-',num2str(sample(rad,1))
,'-','Server','-',num2str(Packetsize));
s2 = strcat(num2str(saturationfactor(rad,1)),'-',num2str(sample(rad,1))
,'-','Client','-',num2str(Packetsize));
```

```
filename1 = strcat(s1,'.txt');      % Sender time stamps
filename2 = strcat(s2,'.txt');      % Receiver time stamps

[packetnumber1,time1] = textread(filename1,'%d %f','headerlines',0);
[r1 c1] = size(packetnumber1);
[packetnumber2,time2] = textread(filename2,'%d %f','headerlines',0);
[r2 c2] = size(packetnumber2);

%Select the right sending packet for starting
%Continue until the Time Limit is passed
for i = 1:r1
    if (packetnumber1(i,1)-1 >= ss) % Start at 0
        if (packetnumber1(i,1)-1 == ss)
            timestart = time1(i,1);
        end
        if ((time1(i,1) - timestart)/1000 < TL)
            p1(j,1) = packetnumber1(i,1)-1;
            t1(j,1) = time1(i,1);
            j = j + 1;
        end
    end
end
result_time1=[a3;t1];
result_number1=[b3;p1];

%Select the receiving packets
for i = 1:r2
    if ((packetnumber2(i,1)-1 >= ss) & (packetnumber2(i,1)-1 < e))
        p2(packetnumber2(i,1)-ss,1) = packetnumber2(i,1)-1;
        t2(packetnumber2(i,1)-ss,1) = time2(i,1);
    end
end
result_time2=[a4;t2];
result_number2=[b4;p2];


%######################## Data 1 ####################
y11 = (t1-t1(1))/1000; % y2 [s]
[n11,out1] =  histc(y11,[0:T:TL]);
n11 = n11(1:end-1,1);
server = n11;
packets_per_interval_server = [e1;server];
%######################## Data 2 ####################
```

```
y21 = (t2-t2(1))/1000; % y2 [s]
[n21,out2] =  histc(y21,[0:T:TL]);
n21 = n21(1:end-1,1);
client = n21;
packets_per_interval_client = [d1;client];
%#####################################################

kbps_server=(n11*Packetsize*8/1000/T);
resultkbps_server = [a1;kbps_server];
kbps_client=(n21*Packetsize*8/1000/T);
resultkbps_client = [b1;kbps_client];

if(ceil(max(kbps_server))>ceil(max(kbps_client)))
    kbps_limit = ceil(max(kbps_server));
else
    kbps_limit = ceil(max(kbps_client));
end

for i = 2:sum(server);                    %Inter Packet Delays Sender
    ipd_server(i,1) = t1(i,1)-t1(i-1,1);
end
result_ipd_server=[f1;ipd_server];

for i = 2:sum(client);                    %Inter Packet Delays Receiver
    ipd_client(i,1) = t2(i,1)-t2(i-1,1);
end
result_ipd_client=[g1;ipd_client];

for i=1:sum(client)                       %Delays
    if((t1(i,1)==0)||(t2(i,1)==0))
        d(i,1)=0;
    else
        d(i,1)=t2(i,1)-t1(i,1);
    end
end
delay=[h1;d];
```

This is the Matlab code from the MobiHealth individual short traces. For the other programs the code is at some places a bit different, but it all ends up with the same: getting the right time stamps to work with for both sender and receiver and calculate the throughput per time interval, the delay and IPD per packet and plotting some figures after this.

The figures which are plotted in most of the programs are:

- Throughput histograms for both Sender and Receiver, throughput difference histogram and autocorrelation plots;
- Delay plot;
- Throughput plot per time interval for both Sender and Receiver;
- Delay histogram and Cumulative Distribution Function;
- Inverse Delay histogram and Cumulative Distribution Function;
- Drift histogram and Cumulative Distribution Function;
- Maximum and Minimum delay per time interval for both Sender and Receiver;
- IPDs for both Sender and Receiver.

Furthermore, there are some other figures, but these figures differ for each program. These missing figures are based on the figures which are mentioned above and are made to put in this thesis.

# BIBLIOGRAPHY


[1] Blekinge Tekniska Högskola. URL: http://www.bth.se. Last checked: 4 October 2006.

[2] ITU-T. Rec. I.350, General aspects of Quality of Service and network performance in digital networks, including ISDNs, 1993.

[3] MobiHealth. URL: http://www.mobihealth.org. Last checked: 4 October 2006.

[4] R. Bults, K. Wac, A. van Halteren, and et al. Goodput analysis of 3G wireless networks supporting m-health services. In *8th International Conference on Telecommunications (ConTEL05)*, Croatia, 2005.

[5] PIITSA. URL: http://www.aerotechtelub.se/piitsa. Last checked: 4 October 2006.

[6] K. E. Wac, R. Bults, A. van Halteren, D. Konstantas, and V. F. Nicola. Measurements-based performance evaluation of 3G wireless networks supporting m-health services. In S. Chandra and N. Venkatasubramanian, editors, *Multimedia Computing and Networking 2005. Edited by Chandra, Surendar; Venkatasubramanian, Nalini. Proceedings of the SPIE, Volume 5680, pp. 176-187 (2004).*, pages 176–187, December 2004.

[7] M. Fiedler, L. Isaksson, S. Chevul, J. Karlsson, and P. Lindberg. Measurement and analysis of application-perceived throughput via mobile links, Tutorial. In *Performance modelling and evaluation of Heterogeneous Networks*, UK, 2005.

[8] J. Walrand. *Communication networks: a first course.* Richard D. Irwin, Inc., Burr Ridge, IL, USA, 1991.

[9] C. Hunt. *TCP/IP network administration.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1992.

[10] P. Arlos. *On the Quality of Computer Network Measurements*. PhD thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 05:2005.

[11] J.F. Kurose and K.W. Ross. *Computer networking: a top-down approach featuring the Internet*. Addison-Wesley, Reading, MA, USA, 2001.

[12] K. Wac and R. Bults. *Performance evaluation of a Transport System supporting the MobiHealth BANip: Methodology and Assessment*. MSc, University of Twente, 2004.

[13] B. Ertman. Niet slechts een keuze. Een vergelijk: Microsoft .NET en J2EE, October 2005.

[14] K. Wac, P. Arlos, M. Fiedler, S. Chevul, L. Isaksson, and R. Bults. Accuracy evaluation of application-level performance measurements. on 15th October 2006 submitted to: 8th Passive and Active Measurement (PAM) conference, 2007, 2006.

[15] Microsoft Corporation. URL: http://msdn.microsoft.com/. Last checked: 28 November 2006.

[16] Sun Microsystems, Inc. URL: http://java.sun.com. Last checked: 28 November 2006.

[17] Tardis 2000 software. URL: http://www.kaska.demon.co.uk. Last checked: 24 November 2006.

[18] C. Class. Synchronization issues in distributed applications: Definitions, problems, and quality of synchronization. Technical report, Technical Report No. 31, TIK, Swiss Federal Institute of Technology, Zürich, Switzerland, December 1997.

[19] T. Hoßfeld, K. Tutschku, and F. Andersen. Mapping of file-sharing onto mobile environments: Enhancement by UMTS. In *PerCom Workshops*, pages 43–49. IEEE Computer Society, 2005.

[20] S. Chevul. *On application-perceived Quality of Service in wireless networks*. Licentiate thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 11:2006.

[21] L. Massoulie and J. Roberts. Bandwidth sharing: Objectives and algorithms. In *INFOCOM (3)*, pages 1395–1403, 1999.

[22] B. Kauffmann, F. Baccelli, A. Chaintreau, K. Papagiannaki, and C. Diot. Self organization of interfering 802.11 wireless access networks. 2005.