# Performance evaluation of a Transport System supporting the MobiHealth BANip: Methodology and Assessment

# VOLUME 1

24 October 2004

Thesis for a Master of Science degree in Telematics, performed at the Architecture and Services of Network Applications chair, Department of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands

**Authors** : K.E. Wac MSc. & ing. R.G.A. Bults

**Supervisors:** Prof.dr.ir. D. Konstantas
Dr.ir. A.T. van Halteren (first)
Dr.ir. V.F. Nicola

**Timeframe:** January 2003 – October 2004

This page intentionally left blank [TPILB].

## Table of Contents

## Summary

MobiHealth is a European project (IST-2001-36006, www.mobihealth.org) that explores the possibilities of GPRS and UMTS broadband wireless networks to support emerging m-health services. The service delivered by the MobiHealth system is an instantiation of an m-health service.

The MobiHealth BANip is a TCP/IP based application protocol designed to transport m-health service related data (e.g. patient vital signs measurements data) over GPRS and UMTS networks (i.e. transport systems). The success of the MobiHealth service depends strongly on the ability of a transport system to support the BANip efficiently. Alternatively, understanding the behaviour and performance of a candidate transport system may influence the BANip design choices and therefore enables perfect tuning of this protocol. Hence, the necessity to perform a "Performance evaluation of a Transport System supporting the MobiHealth BANip: Methodology and Assessment".

Vodafone's (pre)commercial UMTS network (candidate transport system) in the Netherlands (V3GNL) became available May 1, 2003 to the MobiHealth project consortium members in the Netherlands. In addition, the Enschede region of the V3GNL network was dedicated (!) to the project. Since no other users were allowed to access this part of the network, background traffic has not influenced our measurements results. We selected this region of the V3GNL network for our performance evaluation activities and considered it as the (transport) system under test (SUT). The scope of these activities was to provide recommendations to the BANip protocol designer on the BANip service type (e.g. confirmed or unconfirmed), optimal BANip PDU size and PDU rate based on the performance evaluation of the V3GNL network.

We divided the performance evaluation activities in three major parts:

1) development of a generic measurements-based performance evaluation methodology (see table below)

| | |
|---|---|
| *1. State the Goals and System Definition* | |
| *2. List Services and their Outcomes* | |
| *3. Select Performance Criteria (i.e. Metrics)* | |
| *4. List System and Workload Parameters* | |
| *5. Select Factors and their Levels* | |
| *6. Select System and Workload Parameters* | |
| *7. Design and Execute the Experiments* | |
| *8. Analyse, Evaluate and Interpret the Data* | |
| | *a. Select Model Representation* |
| | *b. Parameterise the Model* |
| | *c. Validate and Verify the Model* |
| *9. Present the Results* | |

2) design and implementation of a distributed evaluation system containing workload generators and measurement functions
3) design and implementation of an elementary (simple) statistical application

The execution of our performance evaluation methodology is sequential; i.e. phase-by-phase. This approach stimulates the 'reflection' process (phases 1 to 6) that must result in a precise description of the measurements experiments leading to the performance evaluation of the SUT (i.e. V3GNL). Because of the scope of the SUT's performance evaluation activities, we selected speed as the performance criterion, and delay as the corresponding primary performance parameter. The methodology also involves the design of eleven reproducible experiments (phase 7). The first experiment is based on one set of system parameters and it contains 400 measurements, each executed with different workload parameters. This experiment enabled a detailed performance evaluation of the SUT, and is therefore considered the "benchmark" experiment for all other experiments. The remaining experiments aim to evaluate the performance influence of system parameters and multiple users (5 and 10) on the SUT.

The distributed evaluation system is instrumental for the execution of all experiments. It provides the functionality for controlled workload generation at one or more of the SUT's ingress point(s) and performs correlated measurements at one or more of the SUT's egress point(s). We used the elementary statistical application for (preliminarily) evaluation of the (raw) measurements data to ensure its correctness. The evaluation criterion was the proper timing of 'events' by the distributed evaluation system. The (correct) measurement data was used to develop, validate and verify a D/G/3 queuing model.

The main findings of the performance evaluation activities are based on the benchmark experiment and the developed queuing model:

**Capacity switching behaviour**: The capacity performance characteristics of the SUT are changing in time. We conclude from the measurements data, that the data size (i.e. SDU at TCP SAP) and rate offered to the SUT causes capacity changing behaviour. We specify the capacity switching behaviour as 'bearer assignment' or 'bearer switching'.

For an initial data transport request, the SUT assigns the common bearer (shared bearer with low capacity) by default to both: uplink and downlink directions. For additional data transport requests, it assigns a dedicated bearer (with a higher capacity) that 'matches' the required transmission rate and volume. There are two bearers for the uplink direction: common bearer 16 Kbps and dedicated bearer 1 of 64 Kbps. We distinguish four bearers in the downlink direction: common bearer 16 Kbps and dedicated bearers 1, 2 and 3 respectively: 64 Kbps, 128 Kbps and 384 Kbps (nominal capacities).

The bearer assignment behaviour is always gradual, i.e., the SUT assigns the adjacent bearer (to the current one) to both directions. The exact conditions, under which a bearer assignment takes place, became not clear to us. Therefore, we indicate different SUT behaviour regions in which there is a relation between the data size transported and the assigned bearer (see table below). The observed SUT goodput and delay variation performance parameters depend directly on the bearer assignment behaviour. The bearer assignment may result in an asymmetric communication channels.

| | Region 1 | Region 2 | Region 3 | Region 4 |
|---|---|---|---|---|
| uplink | data size <= 174 B | data size > 174 B | - | - |
| | common bearer | dedicated bearer 1 | - | - |
| downlink | data size <= 2096 B | 2096 < data size <= 10480 B | 10480 < data size <= 23056 B | data size > 23056 B |
| | common bearer / dedicated bearer 1 | dedicated bearer 1 / dedicated bearer 2 | dedicated bearer 2 / dedicated bearer 3 | dedicated bearer 3 |

**Goodput**: We typify the SUT as 'goodput bottleneck' system. For the uplink direction, the maximum goodput is 57 Kbps and for the downlink approximately 300 Kbps. The bottleneck in the downlink direction differs (83 Kbps) in case the Bluetooth option of the pre-commercial Nokia 6650 UMTS terminal (both are system parameters) is used.

**Delay variation**: We observed significant delay variation (i.e. jitter) in the SUT. This has a negative influence on the speed (i.e. delay) performance characteristics of the (MobiHealth) transport system as a whole; recall, the SUT is a subsystem of this (larger) transport system. We identify three possible sources of jitter: 1) SUT bearer assignment, 2) packet loss in one of the SUT's subsystems (causing UMTS or TCP retransmissions), and 3) resource problems in one of the SUT's subsystems.

**Influence of system parameters**: We considered the following system parameters of influence: communication system between the computer system and UMTS terminal (intra communication system), UMTS terminal, computer system, transport service and application buffer size, and inclusion of the Internet subsystem in the transport system's configuration. The default system parameters were notebook computer system, USB intra communication system, (pre-commercial) UMTS terminal, transport service and application buffer size each of 64 KBytes, and the UTnet subsystem included in the transport system configuration.

The SUT uplink behaviour is similar when changing the intra communication system from USB to Bluetooth (for the same pre-commercial UMTS terminal). The influence of the change is significant for the SUT downlink behaviour when using packet sizes larger than 524 Bytes. The Bluetooth instantiation of the intra communication system was identified as a goodput bottleneck. The maximum obtained goodput for downlink communication was 83 Kbps (nominal capacity of Bluetooth link is 115 Kbps).

Changing the other system parameters resulted in similar uplink and downlink behaviour of the SUT. However, increasing the transport service's buffer size (i.e. TCP send/receive buffers) resulted in higher average data transport delays. The SUT goodput (and efficiency) were not influenced for transport service buffer sizes of 32 KBytes and 64 KBytes. Changing the application protocol's buffer size (i.e. socket buffer) from 64 KBytes to 32 KBytes had no influence on the SUT's delay or goodput.

**Scalability characteristics**: We conducted indicative performance measurements to assess the scalability performance characteristics of the SUT. Generally, for a stable SUT (e.g. no loss of data), the delay, throughput and goodput per user are expected not to change significantly, when the number of users increases. However, we observe that if the SUT is concurrently used by 10 end-users in one (small) geographical location, an end-user experiences significant system performance degradation comparing to a single end-user scenario. In the uplink, the SUT performance degradation is observed as an increase (sometimes double) of delays and in addition, a significant increase of delay variation. Moreover, the observed SUT goodput decreases by 50%.

The MobiHealth project did not provide any delay requirements for the transport of BAN sensor-system data. Therefore, we assumed the end-user to raise the following questions: "What sampling frequency of the BAN sensor-system is supported by the MobiHealth system", and "Is real-time transmission of BAN sensor-system samples possible?" We translated these questions to "What service type to choose for the BANip application protocol?", and "What is the required throughput of a MobiHealth transport system to support the BANip PDU rate obtained from the end-user requirements?"

We conclude that an unconfirmed service type of BANip application protocol is the best choice to support MobiHealth SDU rates up to 83 SDUs/sec. The MobiHealth BAN sensor-system (Mobi4 3e1as) is configured for 256 samples (19 Bytes) per second. Real-time transport of sensor-system data (i.e. SDU) for this high sampling frequency is not supported by a UMTS based transport system (e.g. V3GNL). Despite a sample deflation factor of 52%, the transport of one sample plus the aggregated lower level protocol overhead (65 Bytes per sample) results in a gross bit rate of 154 Kbps, whereas the nominal uplink capacity of a UMTS network is 64 Kbps. The optimal BANip PDU size for the maximum supported SDU rate of 83 SDUs/sec is 39 Bytes. In addition, the end-user experiences a SDU delay of 12 ms. In contrast, the BANip PDU size of the current MobiHealth system is 1834 Bytes. The corresponding SDU rate and SDU delay are 1.3 SDUs/sec and 782 ms.

In addition, we conclude from the indicative scalability tests that the upper limit of SUT users per geographical location of 10 $m^2$ is 10; however, this is subject for future research. The area of a geographical location and amount of concurrent users supported at a single location must be investigated.

Finally, we recommend further study on the delay and jitter calculations performed by the elementary statistical application. This work must aim to neutralise the effect of the bearer assignments in statistical calculations of the delay. Moreover, we recommend further investigation of possible jitter sources by means of deploying measurements probes inside the SUT. Our last recommendation is to make the performance evaluation of a transport system dynamic instead of static. This involves automatic repetitive execution of the evaluation activity under predefined conditions to obtain the dynamic behaviour of the transport system used to transport BANip PDUs. The BANip can automatically adapt according to the changing characteristics of the transport system. Hence, the MobiHealth system performance automatically adapts to the changing characteristic of the supporting transport system.

## List of figures

**List of abbreviations and definitions**

| | | |
|---|---|---|
| APN | Access Point Network | Identifies a Packet Data Network (PDN) that is accessible from a GGSN node in a GPRS/UMTS network [TechWeb] |
| BAN | Body Area Network | Collection of (inter) communicating devices, which are worn on the body, providing an integrated set of personalized services to the user [BoV2001] |
| BANip | BAN interconnect protocol | Protocol for interaction between the MBU surrogate (object) and the MBU device [Doko2003] |
| CUS | Component Under Study | A specific part of the SUT of which the influence on the quantitative aspects of the SUT is studied [Hoek1997] |
| CVM | C Virtual Machine | Fully compliant Java virtual machine highly optimized for resource-constrained devices, e.g. consumer products, embedded devices [SunWeb] |
| E1 | | The European counterpart to T1, which transmits at 2.048 Mbits/sec [TechWeb] |
| GPRS | General Packet Radio Service | An enhancement to the GSM mobile communications system that supports data packets. GPRS enables continuous flows of IP data packets over the system [TechWeb] |
| IP | Internet Protocol | The network layer protocol in the TCP/IP communications protocol suite. IP contains a network address and allows messages to be routed to a different network or subnet. IP does not ensure delivery of a complete message |
| ISP | Internet Service Provider | An organization that provides access to the Internet [TechWeb] |
| ITU-T | International Telecommunication Union – T | ITU – Telecommunication Standardisation Sector |
| JVM | Java Virtual Machine | A Java interpreter. The Java Virtual Machine (JVM) is software that converts the Java intermediate language (byte code) into machine language and executes it. |
| kbps | Kilo bits per second | One thousand (1000) bits per second. |
| Kbps / KBps | Kilo Bits per second / Kilo Bytes per second | One thousand twenty four (1024) bits per second. Upper case "B" in KBps means kilo Bytes per second. |
| MBU | Mobile Base Unit | Hardware platform supporting the MobiHealth software architecture; e.g. programmable mobile phone or Personal Digital Assistant [MobiHeal] |
| MTU | Maximum Transmission Unit | The largest frame size that can be transmitted over the network. Known also as the Maximum Transfer Unit [TechWeb] |

| | | |
|---|---|---|
| NTP | Network Time Protocol | A protocol used to synchronise the real time clock in a computer system [NtpWeb] |
| PCMCIA | Personal Computer Memory Card International Association | Standard for connecting peripherals to portable computers [TechWeb] |
| PDA | Personal Digital Assistant | A handheld computer that serves as an organizer for personal information [TechWeb] |
| PDU | Protocol Data Unit | The technical name of a frame of data transmitted over the data link layer (layer 2) in a communications network [TechWeb] |
| PPP | Point-to-Point Protocol | The most popular method for transporting IP packets over a serial link between the user and the ISP [TechWeb] |
| SAP | Service Access Point | An interaction point between a service user and a service provider [Viss2000] |
| SE | Service Element | Logical grouping of Service Primitives with a prescribed temporal ordering [Viss2000] |
| SoD | System of Discourse | The object of measurement activity, distinguished from the system used to asses it (i.e. evaluation system) [Hoek1997] |
| SNTP | Simple Network Time Protocol | Adaptation of the NTP used to synchronise computer clocks in the Internet [TechWeb] |
| SP | Service Primitive | Interaction that can occur at a SAP, defined by stating its purpose and parameters [Viss2000] |
| SUT | System Under Test | That part of the SoD of which the quantitative aspects of behaviour are under study (keeping the qualitative aspects of the behaviour the same) [Hoek1997] |
| TCP | Transmission Control Protocol | The TCP part of the TCP/IP. TCP is one of the two transport protocols in TCP/IP. TCP ensures that a message is sent accurately and in its entirety [TechWeb] |
| TCP/IP | Transmission Control Protocol / Internet Protocol | A communications protocol developed under contract from the U.S. Department of Defence to internetwork dissimilar systems. Invented by Vinton Cerf and Bob Kahn, this de facto Unix standard is the protocol of the Internet and has become the global standard for communications [TechWeb] |
| TOS | Type Of Service | A field in an IP packet (IP datagram) that is used for quality of service (QoS) [TechWeb] |
| TTL | Time To Live | A set maximum amount of time a packet is allowed to propagate through the network before it is discarded [TechWeb] |

| | | |
|---|---|---|
| UDP | User Datagram Protocol | A protocol within the TCP/IP protocol suite that is used in place of TCP when a reliable delivery is not required. If UDP is used and a reliable delivery is required, packet sequence checking and error notification must be written into the applications [TechWeb] |
| UMTS | Universal Mobile Telecommunications System | The European implementation of the 3rd Generation (3G) wireless phone system. UMTS, which is part of IMT-2000, provides service in the 2GHz band and offers global roaming and personalized features. Designed as an evolutionary system for GSM network operators, multimedia data rates up to 2 Mbps are expected using the W-CDMA technology [TechWeb] |
| USB | Universal Serial Bus | A hardware interface for low-speed peripherals and telephony devices [TechWeb] |
| VLAN | Virtual Local Area Network | Logical subgroup within a local area network that is created via software rather than manually moving cables in the wiring closet [TechWeb] |
| V3GNL | Vodafone 3G Netherlands | Vodafone's (pre) commercial UMTS network in the Netherlands |

# 1 Introduction

The purpose of this chapter is twofold: 1) provide the underlying reasons for the research described in this thesis (section 1.1, 1.2 and 1.3), and 2) present the research questions and the approach towards the corresponding answers (section 1.4 and 1.5). The last section is devoted to the description of the thesis structure.

## 1.1 Assignment

This thesis presents a generic methodology for performance evaluation of transport systems that support the MobiHealth BANip (Body Area Network interconnect protocol). A performance methodology is outlined to perform measurements of a selected transport system. The obtained measurements are than used to derive a high-level performance model of the service delivered by this transport system. Both measurements and model are used for performance evaluation of the selected transport system, e.g. delay analysis, bottleneck analysis and scalability analysis.

The authors are graduate students of the faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) at the University of Twente. This thesis serves as the final report for the graduates' combined final project and is part of the MobiHealth technical evaluation deliverable. The assignment was done at the ASNA (Architecture and Services of Network Applications) chair in the context of the MobiHealth project.

## 1.2 Rationale

Patient monitoring outside a hospital is practically non-existing today. If a patient requires measurements of various vital signs (e.g. blood pressure, heart rate, blood glucose level) at regular intervals, this patient has to visit the hospital frequently. Even patients who are not at immediate risk are obliged to visit the hospital and stay there for some time to undergo a series of regular measurements. This may result in high costs for the hospital and healthcare insurers and loss of work hours and morale of the patient.

Remote monitoring of patients may be a solution to reduce health care costs, increase productivity of working patients and increase the quality of life of chronically ill patients. We use the term "m-health" in this context, but what is m-health? The answer of this question lies in the understanding of e-health. Immediately the question rises: What is e-health? People commonly use the term e-health (an abbreviation of electronic health), but it is difficult to find a clear generally accepted definition for this comparatively new term. A few years ago, this term was barely used and now seems to serve as a general "buzzword," used to characterize not only "Internet medicine", but also virtually everything related to Information and Communication Technology (ICT) and medicine. An Internet survey delivers many descriptions of e-health but only a few useful definitions. One of the leading Internet technology providers (Intel), for example, provides one description; they refer to e-health as "a concerted effort undertaken by leaders in health care and hi-tech industries to fully harness the benefits available through convergence of the Internet and health care". It is clear that the scope of e-health is

broader than the deployment of Internet related (ICT) technology in the healthcare domain. The definition presented by the Journal of Medical Internet Research applies to the broader scope:

> *E-health is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies. In a broader sense, the term characterizes not only a technical development, but also a state-of-mind, a way of thinking, an attitude, and a commitment for networked, global thinking, to improve health care locally, regionally, and worldwide by using information and communication technology* [Eyse2001].

The utilization of different modern information and (tele) communication technologies (ICT) will play a significant role in the realisation of any e-health service. A significant proportion of the health demands of next future e-health care models will be satisfied through (private or public) broadband wireless networks. The application of these networks and the related technology (e.g. mobile terminals: PDA's (Personal Digital Assistant), phones) and services can be categorized as mobile based health services, or in short m-health services. The following definition of m-health services is used in this document:

> *M-health services are e-health services that are based on the deployment of wireless mobile telecommunication technology to realise mobile-based health (care) services.*

M-health services are typically supported by telematics systems. One of the major characteristics of these systems is the geographical distribution of system components that are interconnected by a (data oriented) transport system (i.e. telecommunication system). The mobility aspect of the m-health service is usually supported by a wireless component with limited resources in the transport system. These transport system are typically hybrid systems that consist of wireless and wired communication technologies.

Telematics systems in the Internet domain generally use standardised Internet Protocol (IP) based communication protocols. Internet applications supported by telematics systems therefore use IP based application protocols to exchange application specific information (e.g. m-health service data) in a predefined way. Hence, the design of application protocols used by telematics systems that deliver m-health services to end-users must be optimised to the characteristics of the hybrid transport system. The protocol designer must make well-motivated decisions to balance the application protocol functionality with the resources offered by the transport system.

This prerequisite for a successful application of m-health services makes it necessary to understand the behaviour of the transport system. Performance evaluation and assessment of performance data obtained from measurements can contribute to the understanding of the behaviour of complex transport systems.

## 1.3 Context

MobiHealth is a European project (IST-2001-36006, www.mobihealth.org) that explores the possibilities of GPRS and UMTS broadband wireless networks to support emerging m-health services. The service delivered by the MobiHealth system is an instantiation of an m-health service, where standards based TCP/IP Internet networking protocols are used to transport the m-health service data over these next generation broadband wireless networks.

The MobiHealth service is based on a telematics system that consists of four major building blocks: 1) BAN (Body Area Network), 2) BEsys (Back-End system), 3) transport system (i.e. data communication infrastructure) and 4) PortiLab2 end-user application. The MobiHealth team has used the BAN as a healthcare-monitoring tool to measure vital sign data of ambulatory patients. BANs are associated with a BEsys that makes the vital sign data arriving from a particular BAN available to a (remote) end-user (e.g. medical doctor). Transportation of vital sign data and control data is supported by the BANip application protocol that in turn is supported by a transport system.

The success of the MobiHealth service depends strongly on the performance (e.g. speed[1], accuracy[2] and dependability[3]) of the transport service (i.e. service delivered by the transport system). Under the assumption that the MobiHealth service is used for patients with a critical condition, the performance of the transport service is of vital importance to the health of the patient. It is therefore necessary to study the behaviour and scalability of the transport service in a quantitative way to determine the impact of this service on the overall performance of the MobiHealth service. Hence, if the behaviour of the transport service can be predicted in relation to the m-health service data transported, the quality of the MobiHealth service delivered to the end-user can be predicted and adapted according to end-user (pre)defined quality of service profiles.

## 1.4 Research Questions

The goal of the MobiHealth project was to design, implement and trial an m-health service instantiation that uses GPRS and UMTS wireless network infrastructures. The MobiHealth system architecture was designed in May 2002. Although GPRS network infrastructures were commercially available, the characteristics and behaviour of these networks in relation to transportation of large volumes of vital sign data was unknown. Furthermore, the characteristics and behaviour of GPRS networks differ significantly between operators! A pre commercial UMTS network became available the 1st of May 2003 to the MobiHealth project consortium members in the Netherlands. No empirical data was available about the behaviour of this UMTS network in relation to the support of m-health services.

[1] Time interval used to transport data from a source to a destination.
[2] The degree of correctness with which a service is performed.
[3] The degree of certainty with which the service can be used regardless of speed or accuracy.

The MobiHealth system architects used "second source" (practical or theoretical) information about characteristics and behaviour of GPRS and UMTS network infrastructures. Consequently, assumptions were made about the behaviour of the GPRS and UMTS network infrastructures used during the lifetime of the MobiHealth project. The design of the MobiHealth BANip was one of the crucial steps in the MobiHealth system design. One of the major concerns was the selection of the packet size for this application protocol: Fixed packet size or variable packet size? An ad hoc decision was made: variable packet sizes in range of 300 to 1800 Bytes.

The authors had access to Vodafone's commercial GPRS network and pre commercial UMTS network infrastructures in the Netherlands from May 1, 2003 until April 1, 2004 to answer the following research questions:

1) What performance evaluation methodology can be applied to assess a transport system supporting the MobiHealth BANip?

2) What are the performance criteria that should be considered for measurements, modelling and evaluation of the transport system?

3) What model can be used to represent the transport system and what are the conditions under which the model is valid?

4) Consider the MobiHealth BANip as a user confirmed application protocol. What is the optimal PDU size of a UMTS based transport system?

5) Consider MobiHealth BANip as a user unconfirmed application protocol. What is the optimal PDU size and PDU transmission rate of a UMTS based transport system?

6) What are the relevant performance issues related to the transport system?

## 1.5 Approach

To give an answer to the research questions an approach is followed that consists of six activities:

1) Description of system architecture and protocols
2) Selection and further development of a performance evaluation methodology
3) Specification of the transport service characteristics and description of the environment
4) Performance measurement instrumentation
5) Measurements execution and processing
6) Performance evaluation of the transport system for confirmed and unconfirmed application protocols following the selected performance evaluation methodology (ies)

In the first activity background research is performed on the MobiHealth system architecture with a focus on the MobiHealth BANip. Part of this activity is also to develop a basic understanding of the GPRS and UMTS wireless networks operated by Vodafone in the Netherlands.

To derive the performance characteristics of the transport system, a performance evaluation methodology's selection or development activity (2) will be performed. This methodology is a recipe for performance evaluation of the transport system and must include the following steps: objectives determination, planning, instrumentation, execution and evaluation activities. These five steps form the basic ingredients for remaining activities.

In activity 3, the transport service characteristics are specified. For hybrid transport systems that include GPRS or UMTS networks, the transport service offered may have in general asymmetrical characteristics (i.e. different uplink and downlink transport capacity). Furthermore, the transport service is assumed to be reliable (i.e. based on TCP Internet protocol). The description of the transport system environment specifies how the transport service users are using the transport system. For the MobiHealth BANip the uplink interaction between the MobiHealth BAN and the transport system is of vital importance: vital sign data is sent from a MobiHealth BAN over the transport system to the BEsys.

The instrumentation of the performance measurements is the focus of activity 4. It is based on available or to be developed tools and/or equipments. The specification of the transport service characteristics and description of the environment in activity 3 will be used as a starting point for this activity.

Activity 5 focuses on the performance measurements execution and processing. The objectives are: 1) to visualize the (raw) data obtained from each individual measurement activity in an extensive set of graphs, and 2) to generate the statistical data based on the (raw) data obtained from each individual measurement activity.

The final activity (6) focuses on the last two research questions. The performance of a UMTS transport system will be assessed for confirmed and unconfirmed application protocols. The selected performance evaluation methodology (ies) in activity 2 will be used for this purpose. A high-level abstract model is derived by the authors of this report based on activity 3 and validated/verified by means of performance measurements. The transport system abstract model verification aims to check whether it fulfils the quantitative requirements. The model is used for delay, bottleneck and scalability analysis of the transport system.

## 1.6 Thesis Structure

This thesis is structured into seven chapters. For each chapter only the major topics are described here. The chapter's structure is described in the introduction section of that particular chapter.

Chapter 1 introduces this thesis. It provides the underlying reasons for the research described in this thesis, presents the research questions and the approach towards the corresponding answers.

Chapter 2 describes the MobiHealth system and its services. The focal point is the MobiHealth BANip and its implementation aspects.

Chapter 3 introduces the measurement and modelling methodologies used for a transport system performance's evaluation, and the major activities that have to be undertaken in each methodology are described. Moreover, the integration of these methodologies into a single (repeatable) methodology is provided.

Chapter 4 describes the first phase (collection of activities) of the measurements methodology. The goals of the measurements are stated and a conceptual model of a selected transport system and its services is derived. In addition, the performance criteria and a mechanism to apply "measurement data" in a controlled way to the transport system are described.

Chapter 5 presents the second phase of the measurements methodology. This comprises the identification of distinct measurement experiments, their design (i.e. measurement instrumentation) and execution.

Chapter 6 describes the performance modelling and evaluation of the transport system. A simple mathematical model is derived based on the measurements results, which is used as a supporting tool for the performance evaluation process.

Chapter 7 presents the main achievements and conclusions of this thesis. In addition, it presents our recommendations for the existing MobiHealth system (version 3.1.1) and discusses potential subjects of interest for future research.

Figure 1.1 provides a mapping of the defined activities in section 1.5 to seven chapters of this thesis:

| activity | chapter |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 5 |
| 6 | 6,7 |

Figure 1.1 Activity to chapter mapping

# 2 MobiHealth System

This chapter introduces the MobiHealth system. Section 2.1 presents the system overview and section 2.2 presents the system components. Section 2.3 provides details on the functionality of the MobiHealth BANip application protocol.

## 2.1 System Overview

The MobiHealth system delivers an m-health service to patients and healthcare professionals (e.g. medical doctor). The service is based on a telematics system that consists of four major building blocks: 1) a Body Area Network (BAN), 2) a Back-End system (BEsys), 3) the PortiLab2 end-user application running on the End-User Host, and 4) a transport system (i.e. data communication infrastructure).

The Body Area Network is defined as [BoV2001]:

> *Collection of (inter) communicating devices, which are worn on the body, providing an integrated set of personalized services to the user*

Figure 2.1 presents the high-level schematic overview of the MobiHealth system distributed interaction model [Viss2000]. It presents the MobiHealth system's functional building blocks, and they interact by means of interaction points (denoted as *ip*). Particularly the MobiHealth team developed a monitoring service with the BAN and BEsys as the system's core components (denoted in yellow in figure).

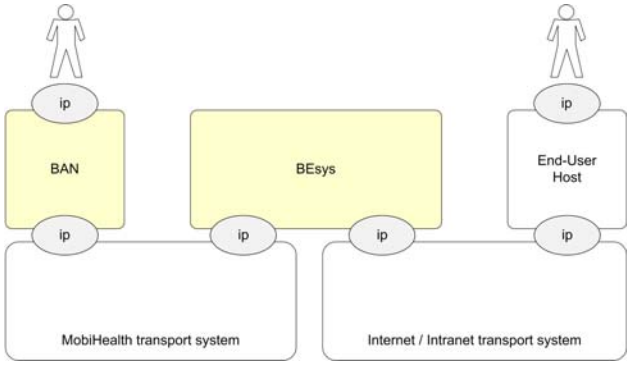



Figure 2.1 MobiHealth system overview

Before the MobiHealth BAN end-user can use the MobiHealth system, the BAN needs to associate with the BEsys. After this initial operation, the BAN can be used as a healthcare monitoring tool for ambulatory patients. This tool measures and transmits the patient's vital sign data (i.e. BAN data) to the BEsys. Transmission of the BAN data is supported by the MobiHealth BAN interconnect protocol (BANip) application protocol. The MobiHealth transport system supports the BANip.

The BEsys makes the BAN data available to a (remote) MobiHealth system end-user (e.g. medical doctor or a patient himself). The end-user uses the End-User Host and an end-user application to access the BEsys online (i.e. the BAN data is transmitted real-time to the BEsys) or offline (i.e. the BAN data is already stored at the BEsys). An underlying Internet (or) Intranet transport system supports this access. End-users do not interact directly with the BEsys.

The following section provides details on the MobiHealth system components as available in version 3.1.1 of the system. [Bult2004] provides more details on MobiHealth system and services.

## 2.2 System Components

Body Area Network

Figure 2.2 presents a high-level overview of the MobiHealth BAN architecture. Sensors measure the patients' vital signs. Processing and communication facilities (i.e. Front-End, Mobile Base Unit and optionally a 2.5/3G terminal) support the transmission of vital sign data. There are two different MobiHealth BAN architectures (Figure 2.2 A and B) depending on the implementation of the interaction between the Mobile Base Unit (MBU) and MobiHealth transport system. The following paragraphs provide description of the MobiHealth BAN functional building blocks.

Sensors are devices that convert a physical phenomenon, for example a BAN end-user's vital sign (e.g. movement, muscle activity, blood flow), into an electrical signal. Therefore, there is always an interaction point between an end-user and sensor(s). Examples of sensors are: a temperature sensor, a pulse oximeter sensor, an alarm sensor, a drop sensor, a respiration band, electrocardiogram (ECG) and electroencephalograph (EEG) sensors.

The sensor's electrical signal is very weak and needs amplification before further processing. A sensor is always wired to the Front-End, and therefore transmits a signal directly to the Front-End. A Front-End has the following functionality:

1) power supply to a sensor
2) amplification of a sensor signal
3) sensor signal digitization into sensor data
4) filtering and compression of the sensor data
5) external transmission of the sensor data (from the Front-End to the MobiHealth MBU)

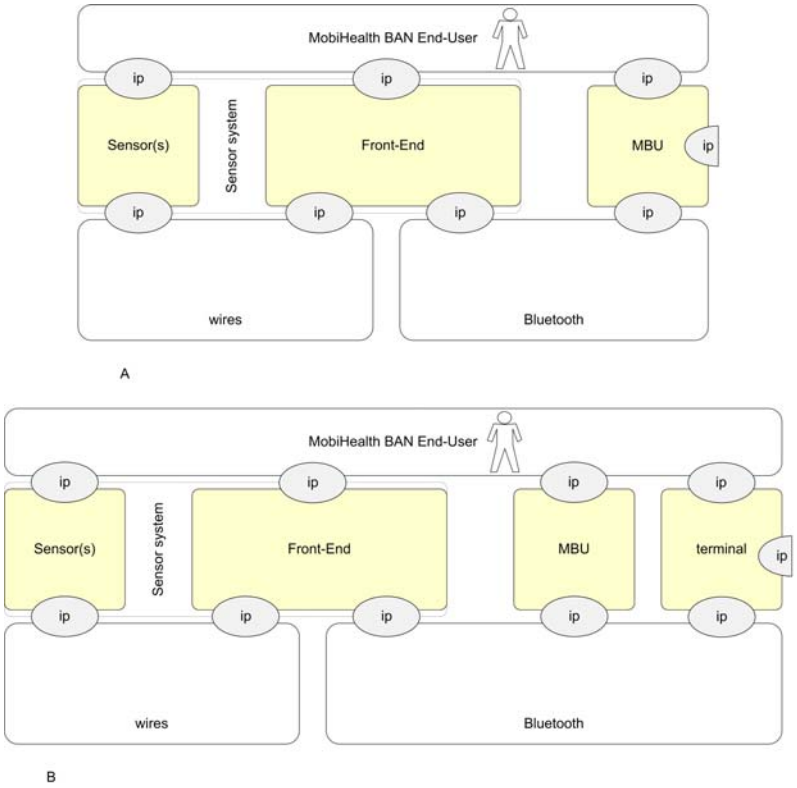



Figure 2.2 MobiHealth BAN architecture

The end-user interacts with the Front-End by means of an on/off button and attaching/detaching sensors.

*Note: The MobiHealth BAN architecture supports sensors and actuators, i.e., devices that influence end-user's behaviour or vital signs. Actuators are not implemented in the current version of the BAN. Potentially, the Front-End would be responsible for control of the actuators.*

A sensor system consists of a sensor(s) and a front-end and is responsible for the acquisition process of the sensor data. The sensor system is highly customized to match the requirements of the monitored end-user. If the sensor system consists of more than one sensor, all sensors operate on the same Front-End clock. The sensor data is acquired in synchronous time intervals from all the sensors. The Front-End multiplexes the sensors' data and transmits it, as sensor system data, to the MBU. The Front-End interacts with the MBU by means of Bluetooth.

The MBU is the MobiHealth BAN's core component and has several functions within the BAN. The MBU controls the BAN, particularly; it starts the sensor system measurements. The MBU acquires the sensor system data from the Front-End and processes this data for further transportation. Sensor system data processed by the MBU is then denoted as BAN data. The MBU initializes the external (i.e. from the MBU to the BEsys) transmission of the BAN data (via the MobiHealth transport system).

A Sensor Viewer component is available on the MBU. This software component enables viewing of the sensor system data on the MBU display. In addition, a manual input component for entering manual observations by the BAN end-user is deployed on the MBU. Manual input is then digitized and multiplexed together with other BAN data.

Communication between entities within a BAN, i.e. between the MBU and Front-End is called intra-BAN communication. For the BAN data transmission there is an external communication called extra-BAN communication. The MBU component facilitates the intra-BAN and extra-BAN communication. The intra-BAN communication is always realized by means of short-range wireless technology. The realization of the extra-BAN communication depends on the way the MBU interacts with the MobiHealth transport system. It is important to note that the MBU may interact with the MobiHealth transport system by means of 2.5/3G wireless technology.

The technical realization of the MobiHealth intra and extra-BAN communication is as follows. The MobiHealth MBU is realized by means of a HP iPAQ H3870. The intra-BAN communication is always realized by means of Bluetooth. For the extra-BAN communication, there are two possibilities: 1) the MBU can be equipped with a 2.5G (i.e. GPRS) jacket, or 2) the MBU communicates via Bluetooth to a Nokia 6650 2.5/3G terminal. In the first case, as Figure 2.2 A denotes, the jacket is considered as a part of the MBU. Therefore, the MBU is directly interacting with the MobiHealth transport system. Figure 2.2 B denotes the BAN architecture in case the MBU communicates via Bluetooth to the terminal. The terminal is then interacting with the MobiHealth transport system. The BAN end-user may interact with the terminal by means of the terminal's keyboard (e.g. changing the connection parameters).

Figure 2.3 presents the MobiHealth system implementation with Front-End and sensors (ECG and an alarm sensors), configured for GPRS communication. Figure 2.4 shows the UMTS configuration of the MobiHealth system (excluding the sensors).

Figures 2.5 and 2.6 provide an example demonstration of both BANs. The woman in Figure 2.5 wears a MobiHealth pregnancy BAN that consists of ECG sensors attached to her belly (hidden under her clothes). The man in Figure 2.6 wears a MobiHealth trauma BAN that consists of ECG sensors (hidden under his shirt), a pulse oximeter sensor (black finger clip on his right hand) and a respiration band (white band around his chest).

Figures 2.7 to 2.9 provide an example of the sensor viewer screen on the MBU for the respiratory (Figure 2.7) and pulse oximeter (Figures 2.8 and 2.9) sensors.

Figure 2.3 MobiHealth GPRS Pregnancy BAN (incl. sensors)

Figure 2.4 MobiHealth UMTS Trauma BAN (excl. sensors)

Figure 2.5 MobiHealth Pregnancy BAN demonstration

Figure 2.6 MobiHealth Trauma BAN demonstration

Figure 2.7 Sensor viewer display - Respiration

Figure 2.8 Sensor viewer display - Plethysmogram

Figure 2.9 Sensor viewer display – oxygen Saturation

Back-End System

The Back-End System (BEsys) is a computer system that provides the MobiHealth BAN security, control, management and the BAN data storage functionalities. Due to these functionalities, a BEsys is usually located in the network of the healthcare provider but can be also located anywhere on the Internet.

Because in this research project we consider the BEsys as a whole, we provide only a high-level description of the BEsys subsystems and their functionality. How these systems interact is less relevant for the purpose of this thesis. However, the interested reader finds a detailed description of the BEsys sub systems, their functionality and the way they interact in [Knop2003].

The BEsys consists of the following subsystems: the SurrogateHost (SH), the LookUp Service (LUS), the BAN Data Repository (BDR) and BAN Streaming Service (BSS) subsystems, which together provide the MobiHealth BAN management and configuration services. Additionally, the Wireless Service Broker (WSB) subsystem provides the security services. The functionality of these subsystems is as follows:

1) The SH is the gateway that manages the MobiHealth BANs 'activity'. The SH knows which BANs are online and what the properties are of these BANs (e.g. how many sensors are in the BAN's sensor system).
The SH is also responsible for gathering the BAN data from the MobiHealth MBU.

2) The LUS is responsible for the discovery and advertisement of the available services of a particular MobiHealth BAN.

3) The BDR is responsible for the collection and storage of the BAN data. It stores each measurement session of a BAN separately. The BDR offers an interface to the MobiHealth end-user willing to retrieve the BAN data via the End-User Host offline.

4) The BSS offers an interface to the MobiHealth end-user willing to retrieve the BAN data via the End-User Host online.

5) The WSB provides the security services (e.g. authentication, authorization) to the MobiHealth BAN. Every online BAN is always connected via the WSB to the SH subsystem (i.e. a firewall inhibits direct connections between the BAN and SH).

*Note: The WSB is denoted in [Doko2003] as a proxy*

End-User Host and the End-User Application

The End-User Host is a networked host (e.g. notebook) usually placed within the healthcare provider network. A healthcare professional, or a patient himself, can connect from the End-User Host via the Internet or Intranet transport system to the BEsys to control the behaviour of the BAN and access the measured BAN data. However, in order to visualize the BAN data, there is the MobiHealth End-User Application named Portilab2. Figure 2.10 shows the screen of the Portilab2 application.

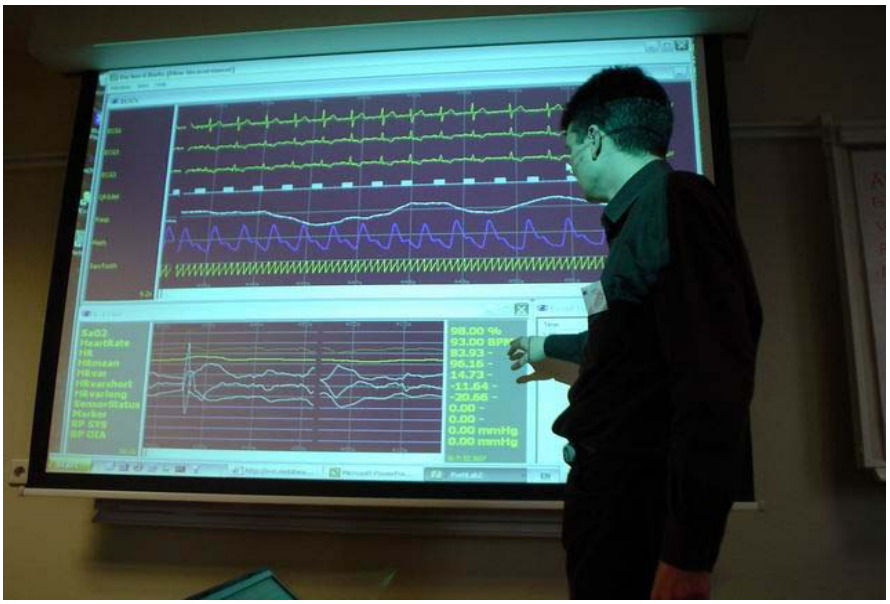


Figure 2.10 MobiHealth End-User Application (PortiLab2)

Transport Systems

The MobiHealth system uses two transport systems: 1) the MobiHealth transport system and 2) the Internet/Intranet transport system. These transport systems provide the transport service of the BAN data. The transport service is not delivered by one transport system belonging to one organizational entity, but by different transport providers. In general, the transport system consists of three (interacting) transport systems belonging to at least three organizations. It is therefore necessary to identify these systems through decomposition of the transport system. If each sub-system is responsible for a section of the 'end-to-end' communication path used between MobiHealth system components, then the 'end-to-end' communication path can be decomposed according to Figure 2.11.

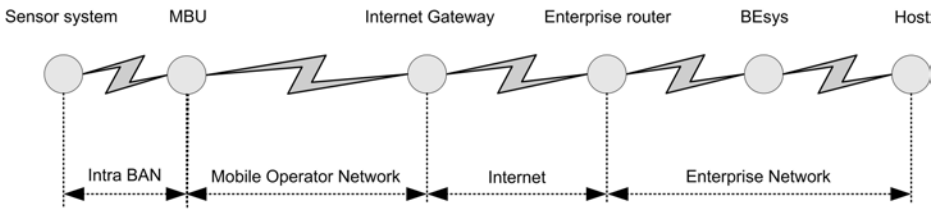



Figure 2.11 MobiHealth system 'end-to-end' communication path

## 2.3 BAN Interconnect Protocol

The MobiHealth system consists of a collection of sub-systems (e.g. BAN, BEsys, end-user Host) with specific tasks. This system is known as the MobiHealth service platform, providing mobile healthcare services to its end-users (e.g. patient, medical practitioners). The internal communication between components of the service platform depend on a special purpose application protocol to support communication between large numbers of wireless and (small) mobile healthcare devices (i.e. BAN) and a wired (back-end) system. The BAN interconnect protocol (BANip) implements this application protocol to support next generation m-health services.

One of the particular goals of this research project is to find the optimal size of a MobiHealth BANip Protocol Data Unit (PDU) for a UMTS based transport system (research question 4 and 5). An exhaustive analysis of the design and implementation of the BANip is therefore outside the scope of this project. However, the interested reader finds a detailed description of the BANip design and implementation issues in [Doko2003].

We focus solely on the BANip sensor data exchange (i.e. *Sensor-Data* private interconnect messages) between the MBU of a BAN and the BEsys[4]. In addition, we consider the management data exchange (i.e. *Registration* lifecycle messages and *RequestSubscription* externally initiated messages) of minor importance to investigate because of the low data volume and less stringent timing requirements on the transport system.

The rest of this section provides a detailed description of the data format used by the BANip to send sensor data from the MBU to the BEsys, and the exchange mechanism of BANip data. We assume that the reader is familiar with the concepts of protocol design described in [Viss2000].

BANip PDU format and exchange

The BANip protocol typically defines the cooperation between BANip protocol entities running on both the MBU and the BEsys. The two BANip protocol entities use PDUs to exchange Protocol Control Information (PCI) as well as an end-user Service Data Unit (SDU). In general, the PCI and SDU specify the two PDU parameters. The PCI consists of end-user specific service data (e.g. BAN sensor identifier) and/or data exclusively used by the protocol to enable communication between peer protocol entities (e.g. PDU sequence numbers). The SDU contains the actual end-user data (e.g. sensor data).

The BANip PDU uses the PCI only for peer protocol communication and contains three fields: 1) BANip protocol version, 2) message type and 3) user data length. As mentioned before, we focus solely on the BANip *Sensor-Data* private interconnect message type; this particular message is represented by type "00 00 00 01". The BANip user data field contains the MobiHealth SDU. The assembly of this SDU is discussed in the paragraph "MobiHealth SDU assembly"). Figure 2.12 denotes the construction of BANip PDU.

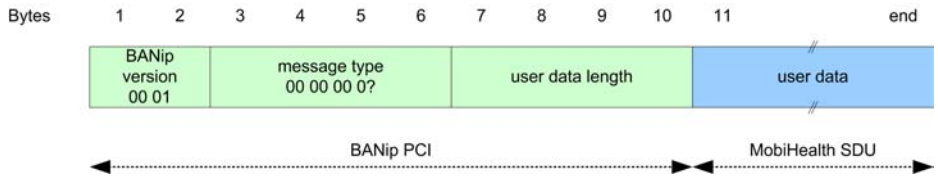



Figure 2.12 BANip PDU format

According to [Doko2003], the HyperText Transmission Protocol (HTTP) encapsulates the BANip protocol; HTTP acts as a lower level service for BANip PDU transportation. MobiHealth system version 3.1.1 uses HTTP 1.1 to encapsulate BANip PDUs. HTTP 1.1 provides two important features for the MobiHealth system [HTTPfaq]:

[4] In reality, the BANip supports data exchange between the MBU and the Surrogate Host component of the BEsys. This level of detail is however not needed for the purpose of this section.

1) HTTP 1.1 supports chunked encoding, which allows HTTP messages to be broken up into several parts. Chunking is most often used by the server (i.e. BAN) for responses, but clients (i.e. BEsys) can also chunk large requests.
2) HTTP 1.1 supports persistent connections. It is very important on wireless networks (e.g. GPRS and UMTS) to do everything possible to avoid latency problems. Persistent connections, which enable transportation of multiple consecutive HTTP requests over a single TCP/IP connection, are one way to reduce network latency by eliminating the overhead of creating a new connection for every transaction. If every HTTP request required the connection to tear down and set up again, performance would suffer greatly.

*Note: [RFC2616] provides extensive information about HTTP 1.1.*

The BANip PDUs are encapsulated in HTTP 1.1 PDUs (i.e. chunks), and transported over a persistent HTTP 1.1 connection. The encapsulation of BANip PDUs (Figure 2.13) adds seven bytes[5] of HTTP chunking overhead to the BANip exchange. A lower level service provider transports the HTTP PDUs. The MobiHealth system typically uses the TCP protocol to implement the lower level service. Appendix A provides an Ethereal dump (captured at the BEsys) of MobiHealth SDU exchanges between the MBU and BEsys, where an IP based transport system supporting TCP implements the lower level service.

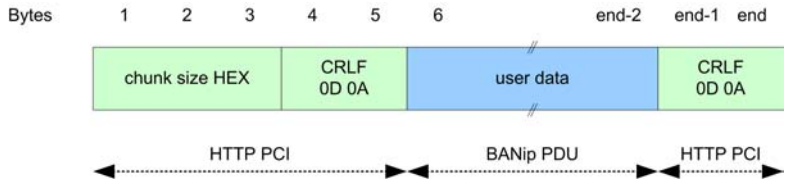



Figure 2.13 HTTP encapsulated BANip PDU

Figure 2.14 denotes an abstract view of SDU and PDU exchange between service and protocol peers. Service user 1 and 2 represent the MBU (sensor data server) and the BEsys (sensor data client) respectively. The Service Access Points (SAP) are the (only) interaction points between the protocol layers within the system.

PDU exchanges in general, depend on either a confirmed service or unconfirmed service. For example, the lower level service depicted in Figure 2.14 may provide a confirmed service. This means that the transmission of an HTTP PDU performed by the sending protocol entity, is succeeded by a transmission of a receive confirmation message from the receiving protocol entity.

[5] The chunk-size header can be smaller than three bytes. In the case of a TMSI Mobi4 3e1as sensor system with all sensors attached and connected to the patient, the chunk-size is presented with three bytes.

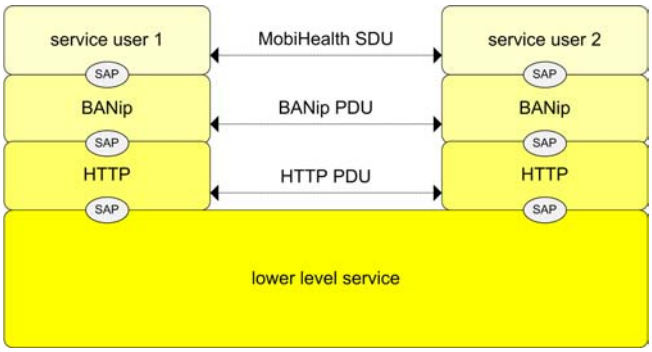



Figure 2.14 BANip protocol stack

[Viss2000] uses a Service Primitive (SPs) to model an interaction between a layer N service user and layer N-1 service provider; a PDU is a representation of a Service Primitive. Service Primitives with a logical relation are grouped into a Service Elements (SE). The confirmed SE consists of a send and receive Service Primitive. The unconfirmed SE only consists of a send Service Primitive. Figure 2.15 denotes the two different Service Elements.

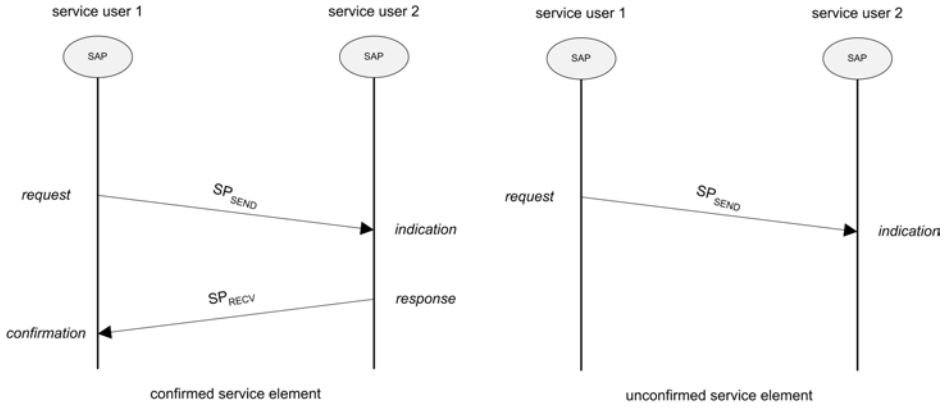



Figure 2.15 Service element type

Based on the previous explanation of the BANip and considering the fact that this protocol is encapsulated by the HTTP application protocol, it is logical to 'unify' both application protocols for reasons of simplicity. In chunking mode, the HTTP protocol is nothing more than a simple 'envelop' for a BANip PDU. Hence, BANip PDU exchange is similar to HTTP PDU exchange encapsulating a BANip PDU. We use BANip PDU and HTTP PDU exchange interchangeably unless explicitly stated otherwise.

The BANip PDU (and HTTP PDU) exchange is based on an unconfirmed service concept. The receiving BANip protocol entity does not confirm the receipt of a BANip PDU. In order to guarantee that the BANip and HTTP PDUs are transmitted and received in good order (e.g. no data modification, correct sequence of PDUs), the HTTP PDU exchange is based on a reliable lower level transport service. As mentioned before, the TCP protocol implements this reliable service.

MobiHealth SDU assembly

Recall that the MobiHealth SDU contains the actual end-user service data; i.e. BAN sensor data. In MobiHealth version 3.1.1, a TMSI Mobi4 sensor system (blue box in Figure 2.4) provides accumulated sensor system data for one or more (physically) attached sensors. Within the MobiHealth project, five different types of TMSI sensor system were used to execute nine different trials. The method used to accumulate the sensor data is generic for all TMSI sensor systems. However, the volume and content differ for each type of sensor system. We focus on the TMSI Mobi4 used for the Dutch trauma trial (type 3e1as) to describe the MobiHealth SDU assembly.

Appendix B provides the XML description of the TMSI Mobi4 trauma-3e1as sensor system as used in the MobiHealth project. Figure 2.16 is deducted from this description and indicates the channel types and corresponding resolution. The Mobi4 sampling frequency is 256 samples per second. Hence, every 1/256 second one sample of 19 bytes accumulated sensor data is generated by the TMSI Mobi4 trauma-3e1as sensor system.

*Note: We ignore the 'manual input' data send together with the sensor sample data by assuming that the compression factor of this data is close to 100%. The manual input is send very rarely by the MobiHealth system end-user.*

| **Mobi4 3e1as** | |
|---|---|
| channel | no bytes |
| ECG | 3 |
| ECG | 3 |
| ECG | 3 |
| resp | 3 |
| SaO2 | 1 |
| pleth | 1 |
| HR | 1 |
| sensor status | 1 |
| marker | 1 |
| sawtooth | 1 |
| subtotal | 18 bytes |
| device_id | 1 |
| total | **19 bytes** |

Figure 2.16 Mobi 3e1as sensor data sample

The MobiHealth SDU assembly is part of the BANip transcoding chain (section 2.5 [Knop2003]). The functionality of this chain is defined in the line "`<SensorSet ID="" transcodingChain="Filter;Packet;EDR;Deflate">`" of the XML description. The *Packet*

and *Deflate* (section 5.5.2 [Knop2003] describes the mechanism details) option are of importance to the SDU assembly. These options indicate the need for packaging of samples into a SDU and the requirement to deflate (i.e. compress) the samples in the SDU.

The *Packet* option activates the packetizer in the transcoding chain. It aggregates a set of 200[6] samples and concatenates them into one temporary SDU. The *Deflate* option activates the *DeflateEncoder* that compresses the temporary SDU into the final SDU. The average compression factor for a temporary SDU is 52% (Appendix C provides details on how this percentage was obtained).

All relevant information is now available to calculate the average size of a MobiHealth SDU:

One sample is equal to 19 Bytes (Figure 2.15).

Packetizer aggregates 200 samples into one $MobiHealth_{SDUtemp}$: $P_{agg} = 200$.

$MobiHealth_{SDUtemp} = P_{agg} * sample$
$= 200 * 19$
$= 3800$ Bytes

$MobiHealth_{SDU} = DeflateEncoder_{52\%} * MobiHealth_{SDUtemp}$
$= (1-0,52) * 3800$
$= 1824$ Bytes

BANip PDU size

Recall that the BANip PCI length is 10 Bytes, and the average length of a MobiHealth SDU (valid under the conditions provided in the previous paragraph) is 1824 Bytes. The average BANip PDU size is calculated by:

$BANip_{PDU} = BANip_{pci} + MobiHealth_{SDU}$
$BANip_{PDU} = 10 + 1824 = 1834$ Bytes (average)

We provide the calculation of the HTTP PDU size below as important background information for chapter 7. An HTTP 1.1 PDU encapsulates a BANip PDU, and the *chunk-size* [RFC2616] of the HTTP PDU represents the average BANip PDU length. The average BANip PDU size is 1834 Bytes and represented by three ASCII characters. These characters represent the hexadecimal value (e.g. 72A HEX) of the HTTP chunk payload (i.e. the BANip PDU). The chunk has two *CRLF* delimiters (two Bytes each). Hence, the $HTTP_{pci}$ length is seven Bytes. The average HTTP PDU size is calculated by:

$HTTP_{PDU} = HTTP_{pci} + BANip_{PDU}$
$HTTP_{PDU} = 7 + 1834 = 1841$ Bytes (average)

[6] Number must be in the range of 1-255. The number 200 is chosen arbitrarily by the BANip designers.

# 3 Transport System Performance Evaluation

In the first few months of the MobiHealth project, the system architects were designing the MobiHealth BANip application protocol. They used "second source" (practical or theoretical) information about characteristics and behaviour of GPRS and UMTS network infrastructures. There was practically no "first hand" public information available about the behaviour of these network infrastructures in relation to the transport of m-health related data. Hence, the system architects had to base the BANip design on assumptions rather then facts about the real-world behaviour and performance of GPRS and UMTS networks; these networks are the MobiHealth transport systems of interest.

Obtaining facts about the real-world behaviour and performance of a GPRS and/or UMTS based transport system is a complex and daunting activity. The "art of performance evaluation" of transport systems (e.g. data communication networks) is in many cases based on mathematical (i.e. analytical) models (e.g. queuing models, Petri Nets and state machines [Have1998]) and simulation of these systems. Performance evaluation of real-world transport systems by means of measurements is an alternative method. [Hoek1997] presents a systematic approach towards performance evaluation of these systems by means of measurements.

The rest of this chapter is organised in four sections. Section 3.1 discusses the selection of the performance evaluation method(s). Section 3.2 and 3.3 describe the selected measurement methodology and modelling methodology for the performance evaluation process. Section 3.4 presents the approach towards the performance evaluation methodology of one alternative MobiHealth transport system.

## 3.1 Performance Evaluation Method Selection

The two methods (i.e. mathematical model plus simulation and measurements) presented for performance evaluation of transport systems can be used in conjunction, but circumstances may influence the order in which these methods are used, or how these methods are used. Given the fact that we had the availability over Vodafone's commercial GPRS network and its (pre) commercial UMTS network from July 2003 until 1 April 2004, we made the choice to use measurements for the performance evaluation of one alternative MobiHealth transport system. We believe that performance evaluation of a transport system by means of measurements can only be successful if a methodical approach towards the execution of the measurements activity is used.

The requirements for such a performance evaluation methodology focus on two major points:

1) decomposition of the measurements activity as a whole into a systematic order of (sub) activities with clearly defined goals

The (sub) activities defined must comprise four phases:

a) preparation:
defines the (transport) system of interest (or part of it), defines the measurements goal and state the performance metrics

b) 'non-invasive' execution:
measurements are executed at the boundary[7] of a transport system

c) verification and interpretation:
measurements data must be sound and accurate before it can be used as a data set for (performance) interpretation purposes; the process of verification comprises measurements data analysis and evaluation

d) presentation of results:
the measurements results are presented in a easy to interpret manner (e.g. graphs), discussed and conclusion are drawn

A mathematical model can support the analysis and evaluation phase to provide estimations in case measurements results are not available.

2) the results of the measurements activity must allow for the derivation of a (crude) analytical model of the transport system

We knew about the existence of the document "Measurements, A Methodical Approach to Performance Measurement Experiments: Measure and Measurement Specification" written by F.W. Hoeksema et al [Hoek1997]. It presents a precise and systematic approach towards performance evaluation of telecommunication networks (i.e. transport systems) by means of measurements. They base their approach on the methodical approach for performance evaluation of computer systems by [Jain1991].

Our requirements for the performance evaluation methodology presented above match the methodical phases discussed in [Hoek1997] and [Jain1991]. [Hoek1997] focuses in detail on the preparation activity of the first requirement. [Jain1991] focuses on the first and second requirement including the defined activities. Hence, we adopt the methodologies of [Hoek1997] and [Jain1991] and use them to describe the applied measurement methodology (section 3.2) for the performance evaluation process.

Deriving a (crude) mathematical model (section 3.3) can support the analysis and evaluation phase to provide estimations in case measurements results are not available. It also provides an interesting opportunity to use it as a supporting tool for the overall performance evaluation of the transport system.

[7] Operators of (pre) commercial GPRS and/or UMTS networks may not be willing to install measurements probes inside their networks.

## 3.2 Measurement Methodology

A performance measurement activity aims to find estimates for transport system performance parameters based on reproducible experiments. These estimates form the basis for a quantitative and/or qualitative analysis of the behaviour of the transport system. This section presents a methodical approach to the performance measurement activity.

The performance evaluation methodology of computer systems by means of measurements consists of the following phases (i.e. sub activities) [Jain1991]:

| |
|---|
| *1. State the Goals and System Definition* |
| *2. List Services and their Outcomes* |
| *3. Select Performance Criteria (i.e. Metrics)* |
| *4. List System and Workload Parameters* |
| *5. Select Factors and their Levels* |
| *6. Select System and Workload Parameters* |
| *7. Design and Execute the Experiments* |
| *8. Analyse, Evaluate and Interpret the Data* |
| *9. Present the Results* |

Figure 3.1 Measurement methodology

These phases also apply to the performance evaluation methodology of telecommunication systems, but need adaptation and/or refinement according to [Hoek1997]. Hence, the description of the nine phases is a combination of the measurement methodology description provided by [Hoek1997] (phase 1-6) and [Jain1991] (phase 7-9).

*Note: in the text below, "system" refers to a (telecommunications) transport system.*

1. *State the Goals and System Definition*

The first phase of the measurement activity is answering the question what system performance characteristics are of importance and why (i.e. what is the objective of the study) and what is the system as an object of the study (i.e. what are the system boundaries). A description of the system should be provided as far as it is known and relevant to the stated objectives. For complex systems, that consists of many subsystems, this activity can imply the system decomposition process, such as the subsystem of interest is identified, or even the particular component of the system is highlighted as of interest. The objective of the study is the key consideration while system delineation. Moreover, the goal set strongly affects the kind of accuracy required for a performance study.

For example, when considering the goal of speed determination of the PC, the goals and system definitions stated by the end-user and the system architect will be different. Namely, the end-user would usually state the interest in the speed of particular application running on this PC, while the system architect will scale the goal down to the speed determination of the (particular) processor, as a component of this complex system.

2. *List services and their outcomes*

The next phase is to answer the question on what services the system offers and what their respective outcomes are. In the particular case where a system offers many services a selection of services of interest and their respective outcomes may be necessary. Generally, these outcomes are divided into three categories: the system may perform the service correctly, incorrectly or refuse to perform the service. Some of these outcomes are desirable and some are not. If the system performs the service incorrectly, an error has occurred.

For example, the database system provides searching and sorting services. The service of interest may be that database responds to a searching query. It may answer to the query correctly, incorrectly (e.g. due to its inconsistency) or not at all (e.g. due to deadlocks). The outcome of interest may be the correct one.

3. *Select Performance Criteria (i.e. Metrics)*

The next phase is to identify, select and define the performance criteria (i.e. metrics) of the transport system. Phase 1 defined the system and phase 2 the services and their respective outcomes. Performance metrics are associated with the three possible service outcomes (i.e. successful service delivery, error, and service unavailability) and relate to speed, accuracy and availability of the services offered by the system.

For example, the performance criteria of the transport system can be availability, speed (e.g. delay, goodput) and reliability (e.g. error rate of the transported data). For each service offered by the system, there are a number of speed metrics, a number of reliability metrics and a number of availability of metrics. For the system that offers more than one service, the number of metrics grows proportionately.

Let us consider the example of the database system from the previous phase. If the system answers to the query correctly, its performance is measured by: the elapse time to perform this service (responsiveness), the rate at which this service is performed (productivity) and the resources consumed by this service (utilization).

The desired accuracy of the evaluated performance parameters is of the vital importance. The question: “What level of detail is required for the estimation of particular parameter?” is raised. According to the desired accuracy level, the mean values, variances or complete distribution of measures of interest need to be obtained.

4. *List System and Workload Parameters*

In this phase, the main concern is the set of parameters that affect the system performance. There are system-specific parameters and workload parameters. System parameters include both the hardware and software parameters, which generally do not vary among various installations of the system. Workload parameters are characteristics of user’s requests, which vary from one installation to the next. For example, the buffer sizes (system parameter) and volume of data being transported (workload parameter) affect the transport system performance.

5. *Select Factors and their Levels*

The list of identified parameters can be divided into these that will be constant during the measurements and these that will be varied. The latter group of parameters are called factors and their values are called levels. The selection criteria of parameters as system factors should be: a) the influence level of particular parameters on the system performance, and b) the feasibility of changing these parameters. Regarding the transport system, the important factors can be size and arrival rate of the transported packets.

6. *Select System and Workload Parameters*

The workload consists of the list of service requests and their arrival intensities. For particular service request, the values of workload and system parameters are determined. The list of service requests is a representative of the system usage in real life or the planned system usage. For measurements, the workload consists of user executable scripts on the system. The measurements done on the transport system need to explore the variety of possibilities of transport service requests.

7. *Design and Execute the Experiments*

The decision on what experiments and in which order to perform these experiments to obtain the maximum information with minimal effort, is taken. The equipment and/or additional code needed for experiments and their instrumentation needs to be listed. The report on when, where, how and by whom the experiments should be executed follows.

8. *Analyse, Evaluate and Interpret the Data*

At this phase, the (raw) measurement results need to be collected and (roughly) validated. In addition, the generation of the statistical data based on the (raw) data obtained from each individual experiment is assumed. Data interpretation phase can follow only if the obtained results are explainable and accepted, otherwise the performance evaluation measurement process needs to be revised and, if necessary, repeated (i.e. start from phase 1).

9. *Present the Results*

At this point, the obtained data is analysed, presented and discussed. It is important that the results are presented in an easy understandable manner (e.g. in graphic form). Often at this point, the knowledge gained by the study may require going back and reconsideration of some of the decisions made in previous phases. For example, redefining the system boundaries or inclusion of other factors and performance metrics that were not considered before. In this case, another cycle through all the steps is required.

## 3.3 Modelling Methodology

Based on the measurement results, a high-level abstract performance model of the transport system can be derived. This model is a tool that can be used for further performance analysis of the system. The model intends to reflect the system performance characteristics obtained in the measurement phase. For example, if the focus of the measurements activity is to obtain the speed-related characteristics (e.g. system delay, goodput) of the transport system, then a queuing model is the most appropriate for quantitative approximation of performance system characteristics.

| |
|---|
| *1. State the Goals and System Definition* |
| *2. List Services and their Outcomes* |
| *3. Select Performance Criteria (i.e. Metrics)* |
| *4. List System and Workload Parameters* |
| *5. Select Factors and their Levels* |
| *6. Select System and Workload Parameters* |
| *7. Select Model Representation* |
| *8. Parameterise the Model* |
| *9. Validate and Verify the Model* |

Figure 3.2 Modelling methodology

Transport system modelling methodology is a recipe for the performance evaluation of the system by means of system modelling and it consists of the following phases:

Phases 1 to 6 are already defined and executed in the Measurement Methodology. The concepts used while modelling the transport system have to be consistent with the ideas provided while executing the measurement methodology.

7. *Select Model Representation*:

The first question to answer is: "What model yields the desired metrics the best?" In general, there are three possible abstract representations of a system: queuing models, Petri nets and abstract state machine models [Have1998]. After answering this question the identification of the system specific parameters (e.g. servers, queues, service disciplines, for a queuing model) and workload-specific parameters (e.g. arrival process, service time) should follow.

8. *Parameterise the Model*:

The data obtained from the measurements needs to be carefully interpreted and processed to estimate the model parameters (e.g. workload parameters, service times, etc.).

9. *Validate and Verify the Model*:

The measurements data can be used to validate that the selected model is an appropriate representation of the system. In addition, the data can be used to verify the accuracy of this model. Model verification amounts to check whether it fulfils the qualitative requirements that have been identified in the selection. This is done by comparing the measurements data with the mathematical outcomes of the model. As a result, model reselection (repeat phases 7-9) or refinement (repeat phases 8-9) is needed.

## 3.4 Performance Evaluation Methodology Approach

This section presents our approach towards the performance evaluation methodology of one alternative MobiHealth transport system. It is a combination of the measurement and modelling methodologies presented in section 3.2 and section 3.3. Figure 3.3 shows the nine phases of our methodology. We copied phases 1-6 and 9 from the measurement methodology and extended phase 8 of this methodology with phases 7-9 of the modelling methodology to enable the use of the model as a tool for evaluation purposes.

| | |
|---|---|
| *1. State the Goals and System Definition* | |
| *2. List Services and their Outcomes* | |
| *3. Select Performance Criteria (i.e. Metrics)* | |
| *4. List System and Workload Parameters* | |
| *5. Select Factors and their Levels* | |
| *6. Select System and Workload Parameters* | |
| *7. Design and Execute the Experiments* | |
| *8. Analyse, Evaluate and Interpret the Data* | |
| | *a. Select Model Representation* |
| | *b. Parameterise the Model* |
| | *c. Validate and Verify the Model* |
| *9. Present the Results* | |

Figure 3.3 Performance evaluation methodology

The presented methodology allows for repetition of any phase. However, in case of a carefully prepared measurements action (phases 1-6), the probability of repetition is the highest for phases 7-9.

The rest of this report follows the successive phases presented in our methodology. Chapter 4 describes the "Transport System Services Modelling" and relates to phases 1-5. Chapter 5 describes the "Performance Measurements Design" and implements phases 6-8 (phase 8, preliminary evaluation of measurements data). Phase 8 is discussed again (detailed evaluation) in chapter 6 "Performance Modelling and Evaluation". It addresses the analysis, evaluation and interpretation of the measurements data prior to the execution of the modelling activities (phases 8a-8c). In addition, chapter 6 describes phase 9, which presents the results of the transport system's performance evaluation activity. Chapter 7 "Conclusions, Recommendations and Future Work" completes this report.

# 4 Transport System Services Modelling

This chapter focuses on the first phase of the methodology presented in section 3.4. It covers phase 1-5 of the methodology (Figure 4.1).

| | |
|---|---|
| ***1. State the Goals and System Definition*** | |
| ***2. List Services and their Outcomes*** | |
| ***3. Select Performance Criteria (i.e. Metrics)*** | |
| ***4. List System and Workload Parameters*** | |
| ***5. Select Factors and their Levels*** | |
| *6. Select System and Workload Parameters* | |
| *7. Design and Execute the Experiments* | |
| *8. Analyse, Evaluate and Interpret the Data* | |
| | *a. Select Model Representation* |
| | *b. Parameterise the Model* |
| | *c. Validate and Verify the Model* |
| *9. Present the Results* | |

Figure 4.1 Performance Evaluation Methodology Phase 1

The performance evaluation goals state the objectives of the measurement activity for a particular system. Section 4.1 provides the answer to the question: "Why is a performance evaluation of the transport system needed?"

Section 4.2 provides a definition (i.e. description) of the transport system (as a whole) and its decomposition into major sub-systems. The "system under test" concept is introduced and used to delineate (setting the boundaries) the part of the transport system that is subject to the performance measurement activity. Sections 4.1 and 4.2 cover the first methodology phase.

Section 4.3 uses the [Viss2000] service concept to describe the services provided by the transport system and "system under test". These services are identified and their possible outcomes analysed by means of service decomposition into "service layers" (methodology phase 2).

Controlled generation of specified messages destined for the "system under test" and measurement of the correlated effects are important functions in the performance measurements activity. An evaluation system must provide these functions. Section 4.4 describes the functional relation between an evaluation system and the "system under test".

When the context of the performance measurements activity is clear, an answer to the following question: "What are the performance criteria[8] of interest?" can be provided. Section 4.5 introduces the "3 x 3 matrix approach" and applies it to describe the "system under test" performance criteria of interest (methodology phase 3).

System parameters (e.g. transport link capacity) may influence performance measurements. In section 4.6, these "system parameters of influence" are classified into "descriptive" and "usage" related parameters. This section concludes with a selection and abstract description of the parameters used in the performance measurements.

The last section (4.6) of this chapter specifies the system ("descriptive") and workload ("usage") parameters of influence. Values will be assigned to selected system and workload parameters in section 4.7. Sections 4.6 and 4.7 cover methodology phase 4 and 5.

## 4.1 Objectives

Performance evaluation of a system based on measurements can be a very difficult and time-consuming activity; systems may be very complex (i.e. consist of many sub systems) and there are potentially many performance parameters (e.g. speed, data transport accuracy, connection setup denial probability) that can be measured. In the case of unlimited resources, performance parameters can be measured exhaustively and a 'perfect' understanding of the system's behaviour under many different circumstances can be obtained. Resources are in general limited and there is in most cases no need to measure performance parameters exhaustively, but only the ones that are of interest. It is therefore important to state very clearly, what the objectives of the performance evaluation are, that is by answering the question:

Why is a system performance evaluation of the transport system needed?

Context

The MobiHealth system is a complex system, which consists of several sub-systems. The transport system is one of the sub-systems that provide a (lower level) transport service to the MobiHealth system. One of the major success factors of the MobiHealth service is the performance (e.g. speed, dependability) delivered by the transport system. Under the assumption that patients with a critical condition will be the main users of the MobiHealth service, the performance of the transport service is of vital importance to the health of these patients. It is therefore necessary to study the behaviour and scalability of the transport service in a quantitative and then qualitative way to determine the impact of this service on the overall performance of the MobiHealth service. Hence, if the behaviour of the transport service can be predicted in relation to the m-health service data transported, the quality of the MobiHealth service delivered to the end-user can be predicted and adapted according to end-user (pre)defined quality of service profiles.

[8] distinguishing qualities

Transport system of interest

The MobiHealth system design abstracts from a transport system's constitution[9] and character[10] for transportation of BANip application protocol data. The only prerequisite of the transport system is to support the transport of IP protocol datagrams. Assuming that a MobiHealth transport system consists of multiple sub-systems, we choose to limit the performance evaluation activity to one transport (sub) system of interest:

> Vodafone's (pre) commercial[11] UMTS network in the Netherlands (V3GNL).

MobiHealth's BANip - transport system interaction

The BANip protocol is responsible for the transportation of BAN data (i.e. management data and patient vital signs data) on a real time[12] basis. The current implementation of the BANip requires the transport system to provide a reliable transport service (section 2.3). The transport system can only fulfil the real-time requirement if it can deliver the BANip packets (a.k.a. Protocol Data Units) within a maximum delivery time.

The BANip packet size and the transport system's delivery time may be correlated; the size of a packet has an effect on the delivery time of that packet. Under the assumption that the delivery time of small packet sizes is different from larger packet sizes, the following question can be raised: "What would be the maximum BANip packet size and packet rate for a specified (maximum) delivery time?" The answer to this question provides a BANip packet size that is optimised for delivery time. However, this packet size may result in lower net transportation of BAN data per time unit (a.k.a. goodput) at the application level due to increasing transport system overhead. Hence, reformulating the previous question into:

> What would be the optimal BANip packet size and packet rate for a specified (maximum) delivery time?

Considering the current MobiHealth BANip implementation (version 3.1.1) uses packet sizes in range of 300 to 1800 Bytes, the answer to the previous question will also provide the answer to the next question:

> Are the packet sizes of the current MobiHealth BANip implementation adequate?

[9] Transport system that consists of multiple interacting transport systems acting as a whole.
[10] Interacting transport system may be implemented in a different way; i.e. use different types of data communication technologies. This results in a hybrid transport system used by the MobiHealth BANip application protocol.
[11] Vodafone announced on February 12, 2004 the launch of it's commercial UMTS service on February 16, 2004h
[12] As fast as required. A realtime system must respond to a signal, event or request fast enough to satisfy some requirement [TechWeb]. The response time requirement for the MobiHealth system is defined in chapter 2.

Objectives

Now that the context is clear, the first question regarding our objectives is put into this context and answered accordingly:

> Consider V3GNL as a MobiHealth transport system for the BANip application protocol:
>
> Why is a performance evaluation of this system needed?
>
> 1) To characterize the quantitative behaviour of this system as a MobiHealth transport service
> 2) To optimise the maximum BANip packet size and packet rate for a specified (maximum) delivery time
> 3) To determine if the packet sizes of the current BANip implementation are chosen adequately

## 4.2 System Description and Boundaries

The MobiHealth system architecture discussed in chapter 2 presented a transport system as a "cloud" of one or more (interacting) transport systems. Recall that BANs and the BEsys are geographically dispersed, patients wearing a BAN are roaming and the BEsys is typically located in an enterprise network (LAN) of a (healthcare) service provider. The probability of having one transport system provider delivering the transport service to the MobiHealth system is therefore low. In general, a MobiHealth transport system consists of at least three (transport) sub-systems controlled by three (or more) organizations: Mobile Operator Network (e.g. UMTS network), Internet, and enterprise network, respectively, controlled by a mobile operator, Internet Service Provider (ISP) and healthcare service provider.

### 4.2.1 System Decomposition

The transport system of interest is Vodafone's (pre) commercial UMTS network in the Netherlands (V3GNL). Vodafone acts as the transport service provider (i.e. mobile operator) for mobile or roaming users; in case of MobiHealth the mobile users are roaming patients wearing a BAN. Assume the BEsys is located at the campus network (i.e. enterprise network) of the University of Twente (UTnet) and the university fulfils the role of a health care service provider that operates the MobiHealth service. Somehow, a connection has to be made between V3GNL and the UTnet for the MobiHealth service to come into operation. A third party acts as an Internet service provider, offering a connection service to Vodafone and the University of Twente. Hence, the whole MobiHealth transport system has a hybrid character and consists in this example of three interacting sub-transport systems: V3GNL, Internet and UTnet.

The MobiHealth transport system can be considered as a whole, and presented as an abstract design model to master the complexity of the system. However, it is necessary to decompose this system into sub-systems to evaluate the contribution of each individual sub-system (in particular the transport system of interest) to the overall performance of the transport system. If each sub-system is responsible for a section of the 'end-to-end' IP communication path between the IP-capable MobiHealth system components, then the 'end-to-end' communication path can be decomposed according to Figure 4.2.

*Note: Recall that the MobiHealth BANip application protocol is based on the IP protocol (section 2.3), therefore the MobiHealth transport system (and its sub-systems) must support the transport of IP datagrams (i.e. provide an IP service).*

In version 3.1.1 of the MobiHealth system, the MBU is the only IP entity in the BAN able to communicate with the BEsys, therefore the intra BAN communication is considered outside the scope of the performance measurements. The MBU is directly connected to the V3GNL[13]; i.e. the transport system of interest and the BEsys will be connected to the UTnet.

*Note: The intra enterprise communication between the host of the end-user and the BEsys is outside the of the performance measurements.*

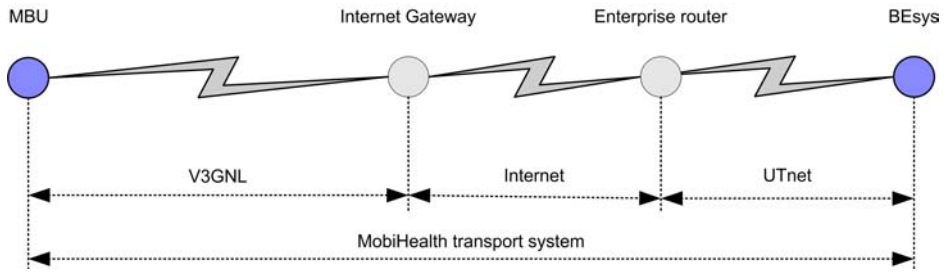



Figure 4.2 Decomposed abstract design model of the MobiHealth transport system

[13] The V3GNL transport system actually delivers a UMTS (i.e. link layer service) service to mobile system users and an IP service to the Internet. For simplicity reasons we assume that the V3GNL transport system delivers a UMTS based IP service to the MBU. Section 4.3 provides an argumentation for this assumption.

### 4.2.2 System under Test

This section discusses a conceptual framework that will be used to delineate the MobiHealth transport system and its sub-systems that are important to the performance measurement activity. The conceptual framework consists of a system of discourse[14] containing a "system under test" (i.e. the transport system of interest) and is based on the "black box – white box" principle. According to [Hoek1997] the conceptual framework consists also of an evaluation system. Section 4.4 describes this system.

Black box – white box principle

The "black box – white box" principle is applied to the decomposed abstract design model of the MobiHealth transport system (Figure 4.2) to generate a conceptual service oriented view[15] of this system. The MobiHealth transport system as a whole can be represented as a black box that provides an IP transport service. From a transport service user perspective it is of no interest how the transport system delivers this service, as long as it delivers the transport service at a certain (sometimes pre defined) quality level. Hence, the transport system is a "black box" providing specified service types to the user.

For performance evaluation purposes the black box model is in many cases not useable. This depends of course largely on the objectives of the performance evaluation. Our objective is to evaluate the performance of the V3GNL sub-system, which is part of the transport system as a whole. Furthermore, Figure 4.2 provides knowledge about other sub-systems of the transport system and therefore discloses the "black box", because it presents the inner working or one possible way of implementation of the "black box". Hence, the MobiHealth transport system as a "black box" is converted to a "white box" (Figure 4.3). A "black box" represents each the transport system's sub-systems. For our performance evaluation objectives the internals of the sub-systems are of no interest[16]. We are solely interested in the IP service delivered by these systems. The interactions between the systems (includes MBU and BEsys) take place at the boundary of each system by means of an IP Service Access Point (IP_SAP); i.e. IP protocol datagrams can only be exchanged between systems by means of Service Primitive[17] (red arrows in IP_SAP) exchange at their individual IP_SAPs.

[14] System of Discourse: an inclusive class of entities that is tacitly implied or explicitly delineated as the subject of a statement, discourse, or theory [Webster]
[15] Based on [Viss2000].
[16] Vodafone kindly refused to disclose their V3GNL network for commercial reasons.
[17] Interaction that can occur at a SAP, defined by stating its purpose and parameters [Viss2000].

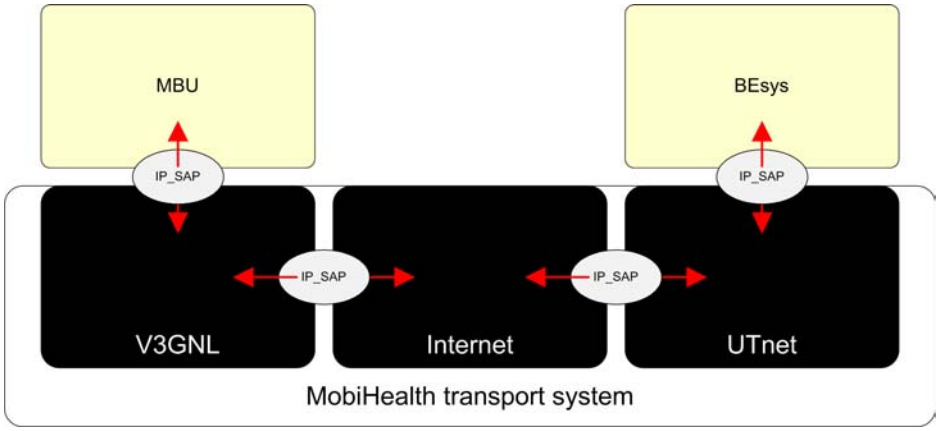



Figure 4.3 "Black box – white box" model of the MobiHealth transport system

System of Discourse and System under Test

The MobiHealth transport system is considered the system of discourse (SoD). Inside this SoD the subject of the measurement activities is identified as the System Under Test (SUT). Figure 4.4 provides an abstract representation of the SoD including the SUT.

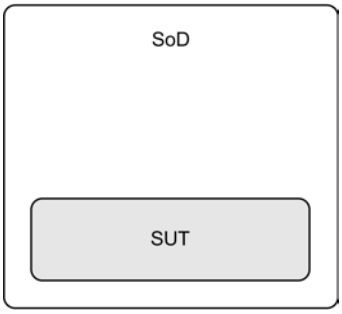



Figure 4.4 Abstract representation of the SoD including the SUT

The SUT is defined as:

> *That part of the system of discourse of which the quantitative aspects of behaviour are under study (keeping the qualitative aspects of the behaviour the same)* [Hoek1997].

The non-SUT part of the SoD may influence the performance measures of the SUT. Hence, a description of the non-SUT part is relevant. The non-SUT part can also play a role in the workload generation[18]. This possibility will be used in the description and instrumentation of the evaluation system (section 5.1). Recall that our objective is to evaluate the performance of the V3GNL transport system. Hence:

The V3GNL transport system is defined as the SUT.

The "black box – white box" model of the MobiHealth transport system (Figure 4.3) is now integrated with the SoD conceptual framework and presented in Figure 4.5. The MBU and BEsys in Figure 4.3 are respectively replaced by an abstract "service user 1" and "service user 2".

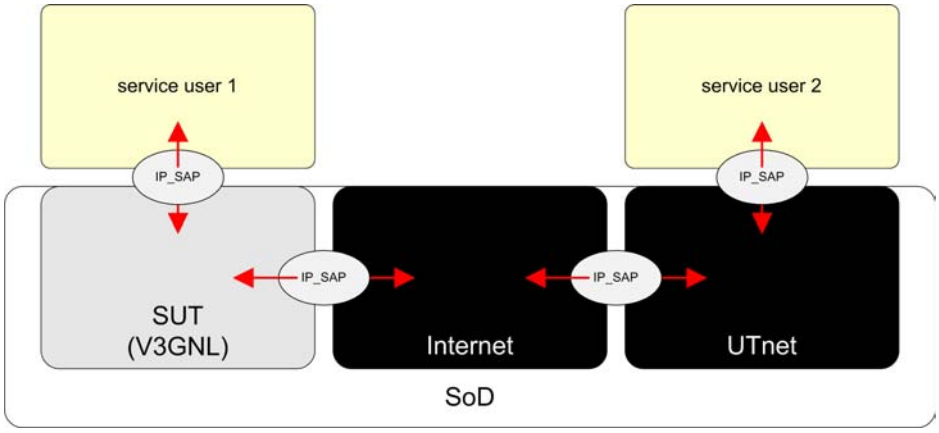



Figure 4.5 SoD and SUT

## 4.3 Service Description

In general, system users are interested in the particular service(s) delivered by this system. They are not interested in the implementation aspects of the system. From a performance evaluation perspective, we are interested in the service(s)[19] of the SUT. The external observable behaviour of the SUT defines these services. Figure 4.5 shows how the SUT is embedded in the SoD. The SUT delivers an IP service to another system embedded in the SoD and to (external) service user 1 of the SoD. Figure 4.5 abstracts from the service interaction mechanism between the SUT and the SoD. Since the non-SUT part of the SoD may influence the performance measures of the SUT, a description of the service(s) delivered by this part is considered relevant.

[18] The reader is invited to read sections 4.2.1 and 7.2 of [Hoek1997 ] for the argumentation.
[19] This is valid for now. In section 4.6 we show that the particular system parameters that influence the behaviour of the SUT are also of interest.

### 4.3.1 Service Decomposition

Recall that the SoD as the MobiHealth transport system delivers a reliable transport service (section 2.3) to the MobiHealth MBU and BEsys sub-systems. We assume that the Transmission Control Protocol (TCP) realises this service. TCP uses the (lower level) datagram transport service (i.e. IP service) provided by the IP protocol (see section 1.7.2 [Kuro2001]). The IP service on its turn uses a "lower level service" to transport the IP datagrams. The IP service level is the service level of our interest, therefore we abstract for now from the "lower level service". Figure 4.6 presents the decomposition of the SoD service according to the Internet Protocol Stack.

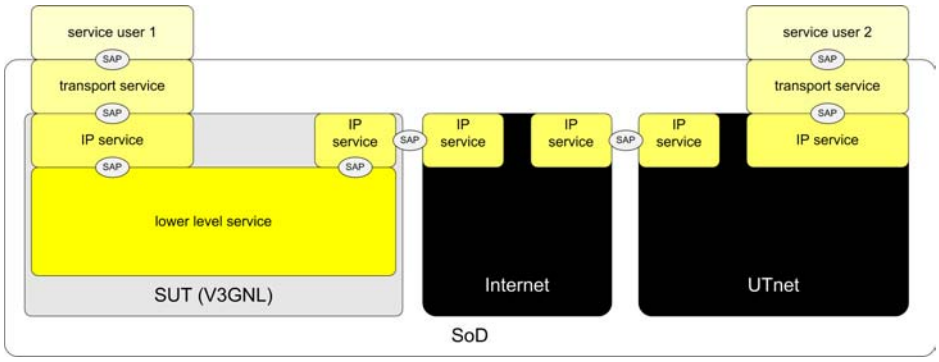



Figure 4.6 SoD service decomposition

*Note: A layer in the Internet Protocol Stack model provides a service (i.e. a function). Services are represented by rounded rectangles. The ellipse "SAP" (Service Access Point) between services represents a point of interaction. Interactions between the IP protocol entities (providing an IP service) inside the Internet and UTnet sub-systems are considered to be irrelevant and therefore not modelled.*

In section 4.2.1, we assume for simplicity reasons that the SUT (i.e. V3GNL) delivers an IP service. Closer study of the SUT reveals that this is not the case. The service delivered to the mobile user is a UMTS link layer service that supports higher layer services (e.g. IP)[20]. The service delivered to the Internet remains the IP service. Figure 4.7 provides the SoD – SUT service decomposition that closer matches reality. The focus of the SUT performance evaluation is on the BANip application protocol. The reliable transport service of the SoD supports the BANip. It is possible to consider the transport service, IP service and host-to-network service[21] as part of the SoD or SUT.

[20] An intermediate link layer service (e.g. PPP) is required to support the IP service.
[21] The host-to-network service represents the combined services of the link layer and physical layer of the Internet Protocol Stack.

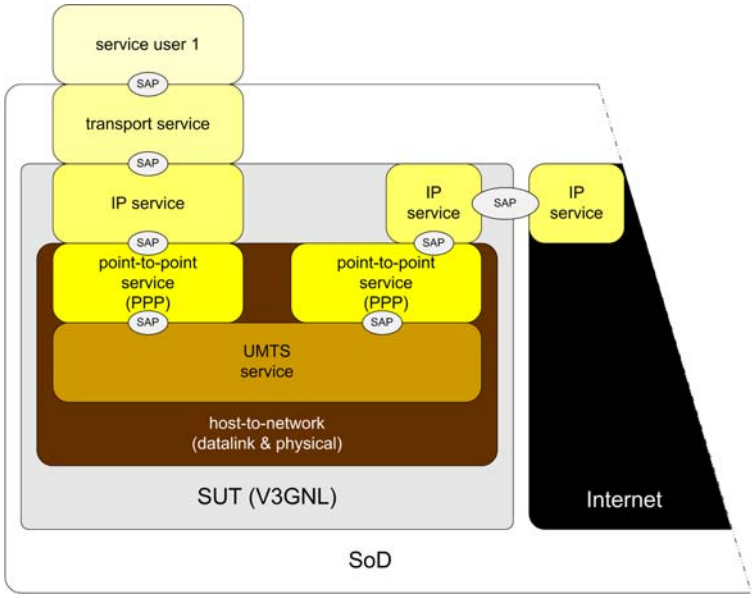



Figure 4.7 SoD – SUT service decomposition

We prefer to separate the transport service from the other services and model it as part of the SoD (as presented in Figure 4.6). This provides the possibility to have alternative implementations of the transport service (e.g. UDP protocol)[22]. Furthermore, we prefer the SUT to deliver one type of service at its boundary that is as close as possible to the actual type of service delivered by the V3GNL network (i.e. the network-to-host service and IP service). Therefore, the IP service and host-to-network service are modelled as part of the SUT. To elaborate on this choice, we have to leap forward to the instrumentation activity:

> *The host-to-network service is implemented on pre commercial UMTS terminals provided by Vodafone. No other UMTS terminals were available during the performance evaluation of V3GNL. Vodafone guaranteed that the UMTS terminals were interoperable with the V3GNL network. Hence, there was an indissoluble relation between the terminal and the SUT, and modelling of the host-to-network service (implemented in the UMTS terminal) as part of the SUT seems logical.*

[22] This rules out the transport service being modelled as a CUS inside the SUT.

*The IP service is not implemented on the UMTS terminals but on the computer systems that interact with the terminals. It seems therefore logical to model it as part of the SoD. We consider the IP service implementation in the operating system running on the computer systems as transparent; i.e. of no influence on the behaviour of the SoD. Therefore, the location of the IP service in the SoD – SUT service decomposition model is not bound to the SoD. We prefer the SUT to deliver one type of service to its users: the IP service. Hence, the IP service is modelled as part of the SUT (assumed to be implemented by an Access Point Network device).*

### 4.3.2 Service Characteristics Analysis

The service decomposition of the SoD and SUT described in the previous section abstracts from the detailed services characteristics. These characteristics are relevant for the performance criteria of interest selection (section 4.5). The SoD may hide the service characteristics of the SUT for the service user. For example, the IP service delivered by the SUT does not guarantee in sequence delivery of IP datagrams (packet). The TCP service of the SoD hides this characteristic by reordering the received IP datagrams before delivering them as a segment (message or part of a message) to the service user.

Figure 4.6 is used as the reference model to analyse the service characteristics of the SoD and the SUT. It is inevitable that some of the SUT service characteristics are hidden by using the TCP based transport service between the SUT and the service user. Recall, the choice for TCP as a transport service is the consequence of the fact that version 3.1.1 of the BANip application protocol is based on TCP.

*Note: If future implementations of the BANip are based on the UDP transport service, the characteristics of the SUT will be disclosed. UDP is a "light weight" protocol providing a connectionless service that enables the transmission of independent packets of data from one system to the other, without a delivery guarantee.*

The service characteristics analysis of the SUT is based on publicly available information of the V3GNL transport system. Furthermore, we consider performance measurements as the focal point of the analysis and therefore assume certain characteristics of the SUT and SoD as presented below.

SUT

1) IP datagram service. A connectionless service that delivers IP datagrams from the ingress SAP (service user side) to the egress SAP (SoD sub-system side) and vice versa, without guarantee of delivery (assuming that this is implemented by an APN).

2) Asymmetrical service with different uplink and downlink transport capacity. The ingress SAP receives uplink datagrams from the service user and the egress SAP receives downlink datagrams form a SoD sub-system (e.g. the Internet). Figure 4.8 presents the maximum transport capacity of the SUT's uplink (64 Kbps) and downlink (384 Kbps) links.

3) Transport service capacity correlates to uplink and downlink datagrams. The SUT reacts to changes in the volume and rate of the datagrams offered at its SAPs. It supports two "UMTS bearers" for uplink and four for downlink. These are identified as coloured rectangles in Figure 4.8. The uplink link has bearer capacities for 0-16 Kbps and 16-64 Kbps, and the downlink for: 0-16 Kbps, 16-64 Kbps, 64-128 Kbps and 128-384 Kbps[23]. The number given as an upper bound of the capacity for a particular bearer is a *nominal capacity* of this bearer [Jain1991].

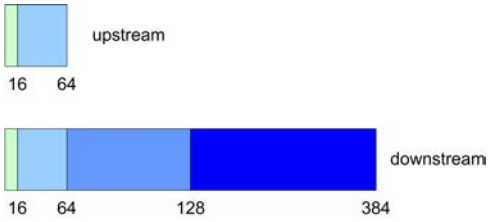



Figure 4.8 SUT uplink and downlink transport capacity

SoD

1) TCP service at the system boundaries (SAPs). Connection oriented service that enables TCP segment (service user message or part of a message) exchange between transport service SAPs in a reliable (i.e. no segment loss) and in sequence fashion[24] (the order of segments send is identical at the receiving end)

2) Service is available. The SoD and its sub-systems are always available for performance measurements independent of the type of measurements.

3) Service is reliable. The SoD performs its function as a transport service in a reliable manner. No loss of data is induced by any of the SoD sub-systems service providers (e.g. due to maintenance).

4) Service execution is correct. The probability of data corruption induced by the SoD sub-systems service providers is zero.

[23] Source Vodafone NL.
[24] See section 3.5 of [Kuro2001] for a detailed discussion of the TCP protocol.

## 4.4 Evaluation System

The SoD is distinguished from an evaluation system that is used to study the quantitative aspects of (parts of) its behaviour. The evaluation system stimulates the SoD to "reveal" its behaviour and in parallel measures this behaviour. Figure 4.9 presents a functional model of the evaluation system. It consists of two functional building blocks[25]: a workload generator and measurement function.

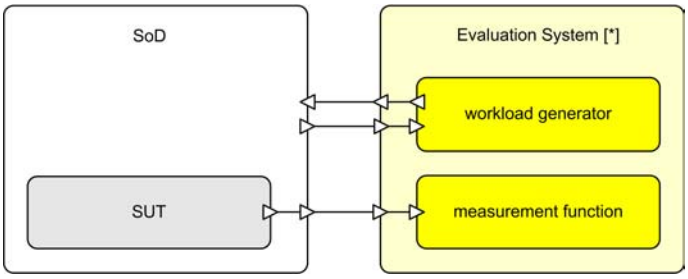



Figure 4.9 Functional view SoD and evaluation system

*Note: The SoD, SUT and evaluation system each provide a particular function. Functions are represented by rounded rectangles. Arrows between functions denote the flow of information. Flows may split or join which is represented by a white diamond and a black square respectively. They enter or exit functions via triangular input and output interfaces.*

The workload generator has an interaction relation with the SoD. It influences the behaviour of the SoD by generation of stimuli and is also able to react to stimuli from the SoD. The measurement function measures the reaction to generated stimuli. It has no interaction relation with the SUT and should influence the behaviour of the SUT as little as possible. The fact that the workload generator and the measurement function have different relations to the SoD implies that they cannot interact (i.e. co-operate) with each other.

*Note: If the measurement function is able to interact with the workload generator, therefore influencing the behaviour of this generator, the measurement function is able to influence the behaviour of the SUT indirectly via the workload generator. This should be avoided!*

[25] These functional building blocks are also denoted as sub-systems.

Above the implicit assumption was made of the existence of a one-to-one relation between the SoD and the evaluation system. This is indeed the case from a high abstract view; the evaluation system is considered as a whole, but may consist of many interacting sub-systems. [Hoek1997] section 4.2.1 provides only a high-level abstract view on the relation between the SoD and the evaluation system. We assume that multiple instances ([*] symbol in Figure 4.9) of evaluation systems may be related to one SoD.

Performance measurements of a SoD with a distributed character (i.e. consists of geographically dispersed components) may require more than one evaluation system. For example: performance measurements with a scalability objective (How many nodes can be supported by the SUT?) may use only one SoD but many instances of the evaluation system. Each workload generator of an evaluation system instance is competing for SoD resources.

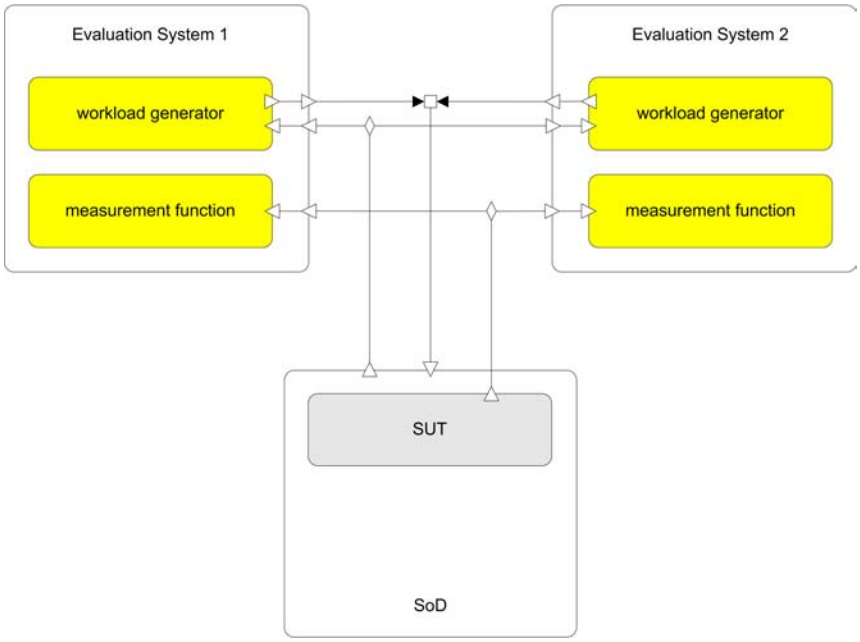



Figure 4.10 Functional view of SoD and two evaluation systems

Figure 4.10 presents an example scenario for two evaluation systems and one SoD, and abstracts from the SoD service access points (SAPs). Having more than one evaluation system creates interesting instrumentation problems:

> How to synchronise the behaviour of the evaluation system instances to create co-ordinated actions?
>
> How to synchronise the functional building blocks of the evaluation system instances in time?

We deal with these questions in section 5.1.3.

The workload generators of the evaluation system instances may interact with each other by means of mutually generated but correlated stimuli for the SoD, for example:

1) workload generator 1 generates stimulus 1 for the SoD
2) the corresponding behaviour of the SoD is to:
   a) forward a data item to workload generator 2, and
   b) generate stimulus 1 for workload generator 2
3) workload generator 2 reacts to the SoD stimulus 1 by generating stimulus 2 for the SoD
4) the corresponding behaviour of the SoD is similar to step 2.

The instances of measurement functions may interact between each other, but are not allowed to interact with any of the workload generator instances (recall previous note).

Recall the functional view of the evaluation system as presented in Figure 4.9. This model is converted to a service oriented view. The interaction points between the SoD and the evaluation system are modelled as SAPs (Service Access Points). In between the SAPs and the SUT a function block offering a transport service is modelled.

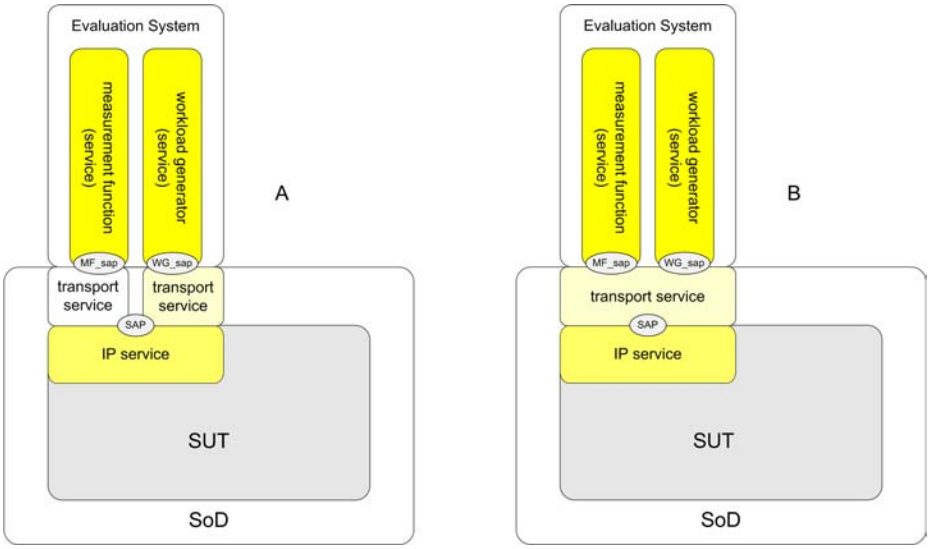



Figure 4.11 Service view SoD and evaluation systems

Figure 4.11-A presents the service oriented model. The transport service between the evaluation system's measurement function and the SUT provides an "ideal" service; i.e. transports information between the measurement function SAP (MF_sap) and the IP service SAP with no influence to the performance measurements. The transport service between the workload generator SAP (WG_sap) and the IP service SAP is implemented by the TCP protocol entity (determined in section 4.3.2).

We prefer the evaluation system's measurement function to interact at the same "level" as its workload generator. To explain this choice, we have to think about the instrumentation of the evaluation system and its functions:

> *The SoD service decomposition presented in section 4.3.1 describes the TCP transport service SAP as the (only) interaction point between the SoD and its service users. The evaluation system can be seen as a service user. This means that the workload generator and measurement function have to interact with the SoD by means of TCP SAPs.*
>
> *As a side effect, the instrumentation of the evaluation system may be simplified. There is no need for the measurement function to interact at the IP SAP. This interaction is considered more difficult to realise than an interaction with a TCP SAP (i.e. tcp_socket). For example, consider the scenario where multiple instances of an evaluation system are interacting with the SoD and SUT. The measurement function of each instance needs to filter inbound and outbound information on the IP service SAP to determine which information belongs to its instance.*
>
> *The disadvantage of using the SoD TCP SAP for measurements is the possible influence of the TCP protocol entity on the accuracy of the performance measurements. This will be discussed in section 4.6.*

Figure 4.11-B presents the revised interaction between the evaluation system and the SoD. The measurement function and the workload generator interact with the SoD by means of their "own" SAP, the MF_sap and the WG_sap, respectively.

The functional view in Figure 4.10 presents two evaluation systems interacting with the SoD. Based on the service oriented model presented in Figure 4.10-B, a service view with the complete SoD can be constructed (Figure 4.12). Each workload generator and measurement function of an evaluation system instance is interacting with the SoD via its "own" SAP. The common resource of the workload generators is the SoD.

Workload Generator

The primary function of the workload generator is to stimulate the SUT by providing a "workload" to the SUT SAP. It can take the initiative (generates a stimulus), or responds to a stimulus received form the SUT SAP (multiple instances of evaluation system scenario). A workload generator can be envisioned as a service user and therefore many different workloads can be offered to the SUT.

In principal, the workload generator should generate SUT-service related interactions at the SUT_sap to "reveal" the behaviour of the SUT. These interactions consist of exchange of service primitives (SP). If SP's are logically grouped with a prescribed temporal ordering, they are considered as a service element (SE). Interactions are in this case SE exchanges. Since this is the most common representation of the service related interaction, SE interactions will be used in the rest of this chapter. If it does not generate SE interactions, there are no events to be measured! The workload provided to the SUT must at least consist of multiple[26] executions of the SUT related SE's to which the measure pertains. It seems therefore natural to specify the workload as a sequence of particular SUT related SE's issued to the SUT.

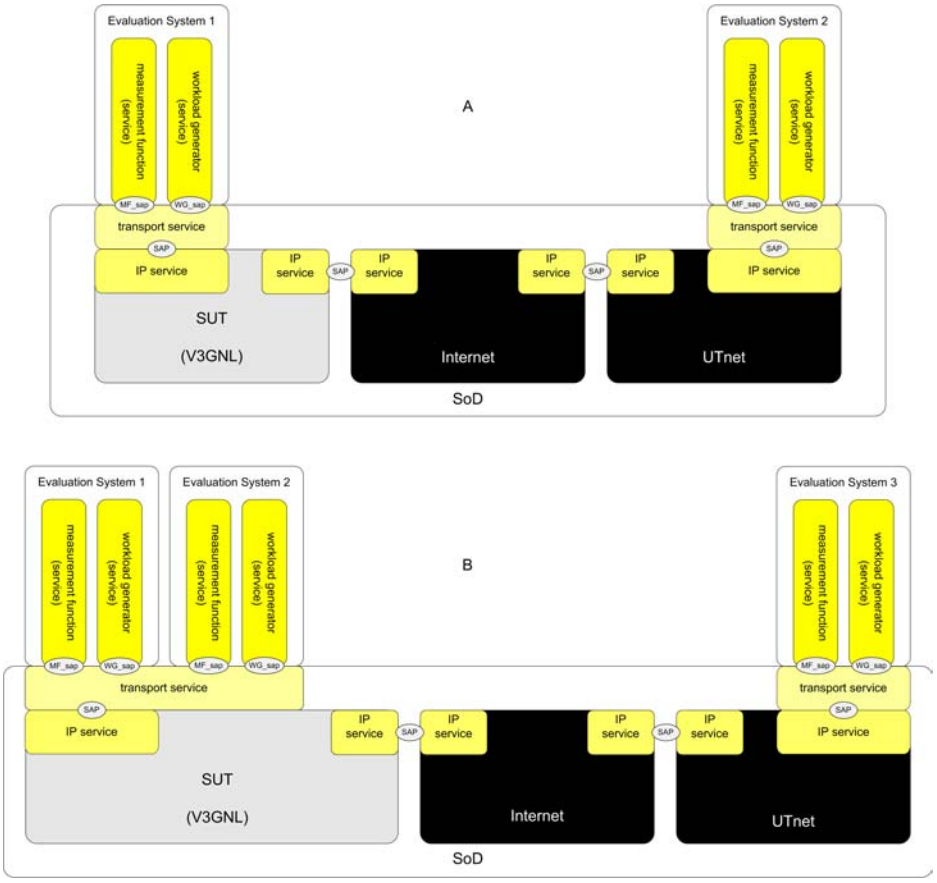



Figure 4.12 Examples of SoD and evaluation systems

[26] The number depends on the (anticipated) behaviour of the SUT and the accuracy of the statistical data to be obtained.

Ideally, the workload generator interacts directly at the SUT SAP (i.e. SUT boundary). It may also interact indirectly by using the capabilities of the non-SUT part of the SoD (see Figure 4.10-B). For example, it may not be possible to "get access" to the SUT SAP due to instrumentation issues (section 5.1). The workload generator interacts with the SUT by means of the WG_sap defined at the SoD boundary. Therefore, the behaviour of the SoD in response to the generated workload may influence the behaviour of the SUT. This may result in a decrease of the overall performance measurements accuracy. The workload generator may also (optionally) introduce background load into the SUT. We are not using this option in our performance measurements.

The workload offered to the SUT can be parameterized. A particular choice of a workload is defined by assigning values to the so called "workload parameters". They can be described as characteristics of user requests (e.g. SE request, SE rate, size of SP) which vary from one SoD and SUT instantiation to the next. Identification of the workload parameters requires knowledge about the usage of the SoD and SUT.

SoD and SUT instantiations can also be parameterised by "system parameters" (a.k.a. "system parameters of influence, section 4.6). These include both hardware (e.g. transport link capacity) and software (e.g. queue depth or buffer size) parameters that generally do not vary among various instantiations of the system. A particular workload choice is called a "workload instance". A performance measurement action can only take place if both a system and a workload are selected and parameterised.

Measurement Function

The measurement function of the evaluation system solely operates at the SoD or SUT SAP level. Figure 4.10-B showed that the measurement function is interacting only with the SoD through its dedicated MF_sap. Hence, the non-SUT part plays a role in the measurement action. The measurement action should be performed passively; i.e. the measurement function "listens" on the MF_sap for events. An event is considered to be the arrival of a SP or SE. The primary goal of the measurement function is to measure events that "happen" at the MF_sap but find their origin at a workload generator's WG_sap.

## 4.5 Performance Criteria of Interest

A performance criterion of a particular SUT can be defined as a characterizing mark or distinguishing quality of that system. In general, a performance criterion is an abstract collection of performance parameters that relate to some property (e.g. the "speed"). The ITU-T provides a systematic method of identifying system performance criteria and their related performance parameters for digital networks (including ISDNs). This method is known as the 3 x 3 matrix approach [ITU1993]. Although we have the impression that the focus is aimed at circuit switched networks, it can and will be used to select the performance criteria of interest for the SUT. We start with a brief overview of the 3 x 3 matrix[27] before selecting the criteria and their parameters of interest.

[27] The description of the matrix approach is copied from [ITU1993] document and modified for our purposes.

### 4.5.1 ITU-T 3 x 3 matrix approach

In this approach, the following definitions are used:

Primary performance parameter:

*A parameter or a measure of a parameter determined on the basis of direct observations of events at service access points or connection element boundaries.*

Derived performance parameter:

*A parameter or a measure of a parameter determined on the basis of observed values of one or more relevant primary performance parameters and decision thresholds for each relevant primary performance parameter.*

The 3 x 3 matrix approach is presented in Figure 4.13. The main features are:

1) Each row represents one of the three basic and distinct communication functions.

   *Note: The access function represents the connectionless as well as connection-oriented services that are possible with transport systems. We will focus only on the connection-oriented service implemented by the TCP protocol.*

2) Each column represents one of the three mutually exclusive[28] outcomes possible when a function is attempted.

3) The $3 \times 3$ matrix parameters are defined on the basis of events at the SoD/SUT boundaries (i.e. SAP) and are termed "primary performance parameters". "Derived performance parameters" are defined on the basis of a functional relationship of primary performance parameters.

4) Transport system primary performance parameters should be defined so as to be measurable at the SAPs of the SoD/SUT to which they apply.

5) Availability is a derived performance parameter. Decisions on the appropriate primary performance parameters, outage (the decision) threshold and algorithms for its definition require detailed study.

[28] Outcomes being related such that each excludes or precludes the other [Webster].

| performance criterion / function | speed | accuracy | dependability |
|---|---|---|---|
| access | | | |
| user information transfer | | | |
| disengagement | | | |

Figure 4.13 3 x 3 matrix approach

Description of the basic communication functions

*access*

The access function begins upon issuing an access request signal or its implied equivalent at the interface between a user and the communication network. It ends when either:

1) a ready for data or equivalent signal is issued to the calling users; or
2) at least one bit of user information is input to the network (after connection establishment in connection oriented services).

It includes all activities traditionally associated with physical circuit establishment (e.g. dialling, switching, and ringing) as well as any activities performed at higher protocol layers.

*user information transfer*

The user information transfer begins on completion of the access function, and ends when the "disengagement request" terminating a communication session is issued. It includes all formatting, transmission, storage, and error control and media conversion operations performed on the user information during this period, including necessary retransmission within the network.

*disengagement*

There is a disengagement function associated with each participant in a communication session: each disengagement function begins on issuing a disengagement request signal. The disengagement function ends, for each user, when the network resources dedicated to that user's participation in the communication session have been released. Disengagement includes both physical circuit disconnection (when required) and higher-level protocol termination activities.

Description of the performance criteria

*speed*

Speed is the performance criterion that describes the time interval used to perform the function or the rate at which the function is performed. (The function may or may not be performed with the desired accuracy.)

*accuracy*

Accuracy is the performance criterion that describes the degree of correctness (e.g. error probability) with which the function is performed. (The function may or may not be performed with the desired speed.)

*dependability*

Dependability is the performance criterion that describes the degree of certainty (e.g. blocking probability) with which the function is performed regardless of speed or accuracy, but within a given observation interval.

### 4.5.2 Selection of Performance Criteria and Parameters

Recall the performance evaluation objectives presented in section 4.1:

1) Characterize the behaviour of the SUT (i.e. V3GNL) as a MobiHealth transport service,
2) determine the optimal BANip packet size and packet rate for a specified (maximum) delivery time, and
3) determine if the packet sizes of the current BANip implementation are chosen adequately.

The focal point of the objectives is aimed at the SUT's communication function *user information transfer* and its performance criterion *speed*. The workload generator of the evaluation system will act as the transport service user, and substitutes the BANip application protocol mentioned earlier in the objectives. The [ITU1993] description of performance criterion *speed* can be specified to match our objectives:

> *Speed is the performance criterion that describes the delivery time that is used to successfully perform a transfer function and the rate at which this transfer is performed. The transfer function realises the transfer of a SE generated by a workload generator at a WG_sap1 (source) to another WG_sap2 (destination).The measurement function "listens" at its MF_sap for arriving SEs destined for WG_sap2.*

The *speed* criterion is linked to the primary performance parameter *delay* and derived performance parameters *jitter* and *goodput*. Figure 4.14 presents the selected performance criterion and the performance parameters.

| performance parameter / performance criterion | delay | jitter | goodput |
|---|---|---|---|
| speed | *primary* | *derived* | *derived* |

Figure 4.14 Performance criterion and parameters of interest

*delay*

Primary performance parameter that is a measure for the time interval used to transfer a SE from:

a) WG_sap1 (source) to another WG_sap2 (destination), or
b) WG_sap1 (source) to another WG_sap2 (destination) and vice versa.

Figure 4.15 presents a time sequence diagram for the transfer of a "user confirmed" SE. The time interval used to transfer the SE is the sum of the *request-indication* time interval and the *response-confirmation* time interval; in formulae:

$$SE_{user\_confirmed}\_delay = SE\_delay, uplink + SE\_delay, downlink$$

$$SE_{user\_confirmed}\_delay = (T_{RECV,\ WG\_sap2/MF\_sap2} - T_{SEND,\ WG\_sap1/MF\_sap1}) + (T_{RECV,\ WG\_sap1/MF\_sap1} - T_{SEND,\ WG\_sap2/MF\_sap2}).$$

The time interval for the transmission of a "user unconfirmed" SE can be calculated by the formulae:

*uplink*

$$SE_{user\_unconfirmed}\_delay = SE\_delay, uplink$$

$$SE_{user\_unconfirmed}\_delay = (T_{RECV,\ WG\_sap2/MF\_sap2} - T_{SEND,\ WG\_sap1/MF\_sap1})$$

*downlink*

$$SE_{user\_unconfirmed}\_delay = SE\_delay, downlink$$

$$SE_{user\_confirmed}\_delay = (T_{RECV,\ WG\_sap1/MF\_sap1} - T_{SEND,\ WG\_sap2/MF\_sap2})$$

*jitter*

Derived performance parameter, which is a measure of the variation in delay.

*goodput*

Derived performance measure, determined as the size of the data-carrying component (payload) of a particular SE, divided by the observed time at which this SE is being transferred from WG_sap to MF_sap.

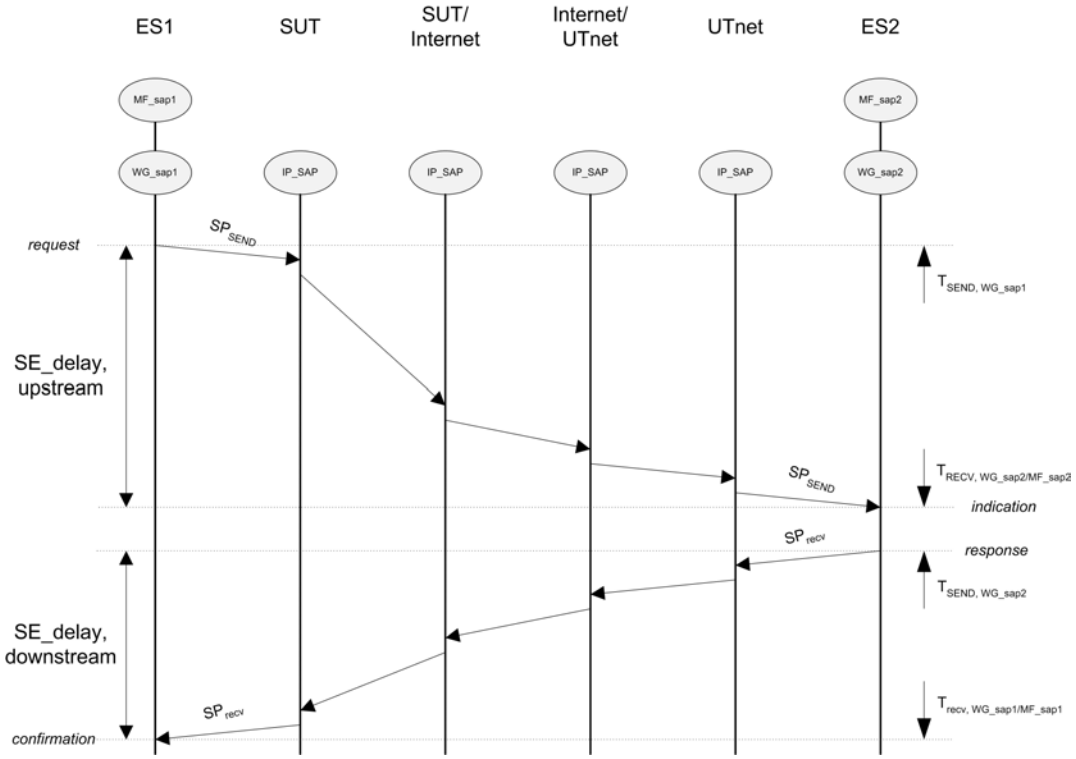



Figure 4.15 Time sequence diagram for the transfer of a SE

## 4.6 System Parameters of Influence

Recall the description of the workload generator at the end of section 4.4. We introduced the notion of "system parameters" being related to instantiations of a SoD and SUT. This section describes a special set of system parameters that may influence the outcome of the SUT performance measurements. This set is identified as "system parameters of influence". The concept of a SoD and SUT instantiation will be described and related to the "system parameters of influence".

System parameters of influence

These set of system parameters influence the outcome of the performance measurements if[29] the values of these parameters are change. For example, if the capacity of a transport link in the SUT is reduced by 50%, the primary performance parameter *delay* will change (i.e. increase) for the same volume of information to send. According to [Hoek1997] this intuitive approach will not result in a proper identification of all candidate system parameters of influence. Furthermore, a precise definition of a system parameter of influence is needed to clearly understand what these parameters are influencing. [Hoek1997] uses the probability theory to answer this question and concludes with a clear definition of system parameters of influence. We believe the following quotations are crucial for the reader to grasp the meaning of the definition:

1) *The performance measure[30] is either a random variable, i.e. a variable that has some probability distribution function (PDF) or a probability. In case of a random variable, e.g. delay, one may think of the performance measure as described by the moments of the PDF of the random variable, such as mean delay and variance of the delay.*

2) *A value of the measure being the outcome of a single probabilistic experiment is taken as an observation for statistical inference.*

3) *A sequence of observations is called a 'sample'. It results from repeating the probabilistic experiment.*

A measure is defined as:

> *A random variable or a probability that quantitatively characterises the behaviour of the SUT [Hoek1997] section 4.6.2.*

*Note: We consider the performance parameters of interest (section 4.5.2) as performance measures and conform to the "measure" definition from this point onwards in our thesis.*

The system parameter of influence is defined as:

> *A parameter whose value is a priori considered to affect the probability distribution function (PDF) of the measure (significantly).*

This definition implies an as good as possible understanding of the SoD and SUT (sections 4.2 and 4.3). It also limits the set of parameters to those that clearly (i.e. significantly) affect the probability distribution function. Identification of candidate system parameters of influence pertaining to SE is based on Table 10 in [Hoek1997] is presented in Figure 4.16:

[29] "if and only if"
[30] We prefer "primary performance parameter measure".

| Parameter class | Parameter sub-class | Candidate system parameters of influence |
|---|---|---|
| *system description related para-meters* | *SUT structure* | computer system<>intra comm. system<> UMTS terminal<>V3GNL network<>APN[31] |
| | *SUT internal behaviour* | IP service + PPP service Computer system<> Intra comm. service<>PPP service UMTS terminal<> V3GNL service<>APN IP service (incl. routing function) |
| | *SUT external behaviour* | IP service:<br>a) SP_send(header, payload),<br>b) SP_recv(header, payload),<br>c) TOS field: not used (Best-Effort QoS)<br>d) TTL field: expect probability of being "0" low.<br>e) No options field: no header extension<br>f) MTU size = unknown |
| | *quantitative aspects of the SUT* | uplink transmission capacity<br>downlink transmission capacity<br>buffer capacity N+1 transport service layer: unknown<br>buffer capacity N-1 host-to-network service layer: unknown<br>Max PDU length (MTU): unknown<br>Max number of concurrent service users per UMTS cell |
| *system usage related para-meters* | *workload* | see section 4.7. |
| | *background load* | SoD and SUT |

Figure 4.16 Identification of candidate system parameters of influence

We also identify SoD candidate system parameters of influence that resides in the transport service and in the non-SUT part of the SoD (e.g. Internet and UTnet in Figure 4.6). According to the [Hoek1997] approach used to identify the system (i.e. SUT) parameters of influence, these SoD parameters should not be part of the selected system parameters of influence. We acknowledge the fact that mixing SoD and SUT parameters of influence is not in line with the approach; however, we do combine them because the identified SoD may influence the performance measurements of the SUT.

The candidate system parameters of influence in Figure 4.16 are completed with SoD system parameters of influence and analysed to derive a set of "selected system parameters of influence". Only parameters for which the effects on the performance measurement are investigated or for which the quantitative aspects are known will be selected. Figure 4.17 presents the overview of SoD and SUT system parameters of influence. The "factors and levels" of these parameters are specified in section 4.7.

[31] More detailed knowledge was not provided by Vodafone (V3GNL + UMTS terminal) or Nokia (UMTS terminal) for commercial reasons.

| Parameter class | Parameter sub-class | Selected system parameters of influence |
|---|---|---|
| *system description related parameters* | *SUT structure* | computer system<>intra comm. system<> UMTS terminal<>V3GNL network<>APN |
| | *SoD structure* | Computer system <>SUT<>Internet<> UTnet<> Computer system |
| | *SUT internal behaviour* | Assume not of influence. |
| | *SoD internal behaviour* | Assume not of influence[32] |
| | *SUT external behaviour* | Assume not of influence |
| | *SoD external behaviour* | TCP service:<br>a) SP_send(header, payload),<br>b) SP_recv(header, payload),<br>c) assume TCP specific options are of no influence |
| | *quantitative aspects of the SUT* | uplink transmission capacity<br>downlink transmission capacity<br>number of concurrent service users per UMTS cell |
| | *quantitative aspects of the SoD* | application send/receive buffer sizes (above transport service)<br>TCP (socket) send/receive buffer sizes |
| *system usage related parameters* | *workload* | see section 4.7. |
| | *background load* | SUT:<br>assumed zero for pre commercial phase,<br>unknown for commercial phase.<br>SoD (Internet + UTnet):<br>existing, but assumed to be of no influence (see *SoD internal behaviour*). |

Figure 4.17 Selected SoD and SUT parameters of influence

SoD and SUT instance

A service delivered by a system discloses the external behaviour of this system. The internal mechanisms of the system contributing to the service provisioning are hidden. From a service user perspective the internal mechanisms or implementation of the system are irrelevant. However, from a performance evaluation perspective the implementation may play a significant role in the quality (e.g. speed) in which the service is delivered.

One implementation of the system may perform better then another. The selected system parameters of influence are considered parameters that modify the implementation of a particular system. Different combinations of system parameters of influence create multiple implementations of a system that delivers the same type of service with (anticipated) different service quality level. Each implementation of a SoD and SUT is considered to represent a particular case or instance[33]. For example, Figure 4.18-A presents an instance of the SoD structure containing the SUT, the Internet and the UTnet. Figure 4.18-B shows an alternative instance containing the SUT and the UTnet.

[32] The transport capacity of the Internet and the UTnet part of the SoD is assumed to be significantly larger then the workload generated during the performance evaluation measurements. We do however validate this assumption in Chapter 6 (delay analysis).
[33] In our performance measurements the SoD and SUT are tightly coupled acting as one *instance*.

Section 5.1.1 presents all the different SoD and SUT instances that used in the overall performance evaluation activity and will be linked to the separate performance measurements.

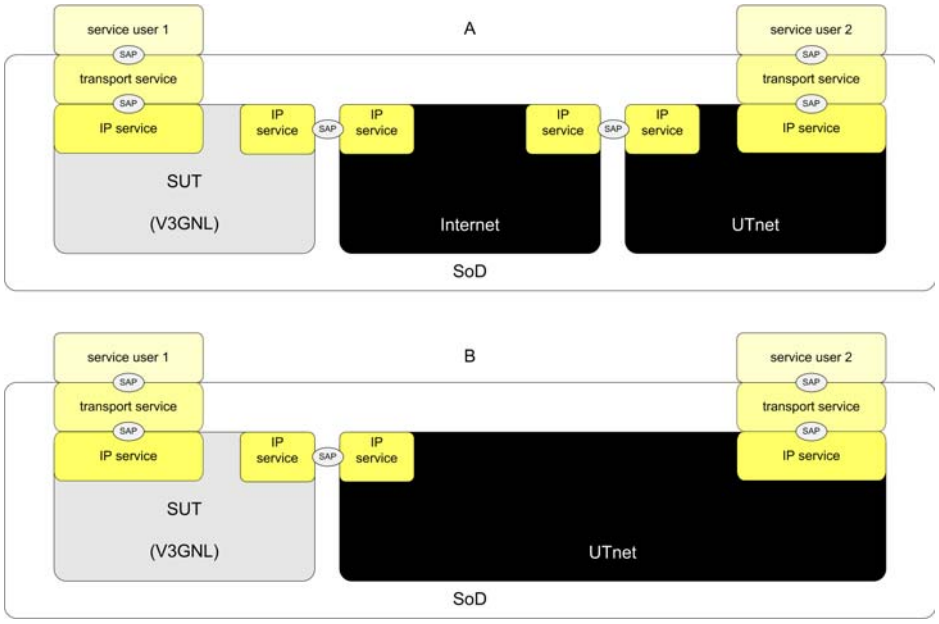



Figure 4.18 SoD instances

## 4.7 Workload and System Parameters

Workload parameters

Figure 4.17 presents *workload* as a "system usage related parameter of influence". Workload parameters are therefore parameters for which the effects on the performance measurement can be investigated. In [Jain1991] section 6.1, workload parameters are defined as:

> *Measured quantities, service requests, or resource demands, which are used to model or characterise the workload.*

If workload parameters characterise the workload, then they specify the parameters and rate of the SE's generated by the workload generator. Recall, the workload generator of our evaluation system will provide multiple executions of SUT related SE's. These parameterised SE's are typically generated, as a sequence of SE's to derive a statistical sound measure of a primary performance parameter (in our case *delay*).

Based on the second performance evaluation objective (section 4.1), the workload parameters selected are *size* and *rate*[34]. These workload parameters will disclose the uplink and downlink behaviour of the SUT for different sizes and rates of SE's; hence, meeting the first objective. A workload parameter that is not easy to identify from the objectives, is the *number* of workload generator instances (i.e. service users). The *number* parameter discloses the behaviour of the SUT for concurrent operating workload generators in one UMTS cell. It instruments a scalability performance measurement of the SUT.

Section 2.3 describes the correlation between the real world domain of the BANip and the conceptual domain of service elements. Recall that the BANip was characterised as an unconfirmed service; the application protocol sends sequences of messages to the transport service without waiting for an acknowledgement from its receiving protocol peer. Therefore, the workload generator has to implement an unconfirmed service to characterise the BANip message sequences correctly. Hence, we consider the *unconfirmed service* as a workload parameter.

> *Note: Because previous implementations of the BANip were based on a confirmed service concept, we also include the "confirmed service" as a workload parameter.*

In short, the specified workload parameters are:

> *size*, *rate, concurrency number, unconfirmed service* and *confirmed service*.

Workload Parameters and Values

Workload parameters need to receive a value to have a meaning in the execution of performance measurements. In [Jain1991] section 2.2, the activity of assigning values to the workload parameters is described as: assigning levels (i.e. values) to factors (i.e. parameters to be varied). We use "factors" and "workload parameters" as the same concept; the same applies to "levels" and "values". For every workload parameter, we select the values used during the performance measurements.

*size*

To characterise the uplink and downlink behaviour of the SUT for different sizes of this workload parameter, we select two different ranges of SE sizes:

uplink : 174 Bytes up to 8192 Bytes
downlink : 174 Bytes up to 48208 Bytes

[34] SE's per second.

The increment of the uplink and downlink SE sizes is chosen such that the quantitative aspects of the SUT are disclosed. The 20 x 20 matrix presented in Figure 4.19 presents an overview of the selected values for the *size* workload parameter. The rows indicate sizes for uplink measurements and the columns indicate sizes for downlink measurements. The number "500" in the top left corner indicates the number of observations per measurement (section 5.2.1). Because of the large number of observations (20x20x500) needed to execute this performance measurement, the 20 x 20 matrix is only executed for one particular instance of the SoD/SUT and one "workload instance". We will refer to this particular performance measurement as the "benchmark". A smaller 8 x8 matrix (Figure 4.20) will be used for different instances of the SoD/SUT. The results of these measurements can be correlated to the "benchmark". The 20 x 20 matrix and the 8 x 8 matrix will only be used in combination with the *confirmed service* workload parameter.

| **500** | 174 | 349 | 524 | 1048 | 2096 | 4192 | 6288 | 7336 | 8384 | 9432 | 10480 | 12576 | 14672 | 15720 | 16768 | 23056 | 29344 | 35632 | 41920 | 48208 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 174 | | | | | | | | | | | | | | | | | | | | |
| 349 | | | | | | | | | | | | | | | | | | | | |
| 524 | | | | | | | | | | | | | | | | | | | | |
| 786 | | | | | | | | | | | | | | | | | | | | |
| 1048 | | | | | | | | | | | | | | | | | | | | |
| 1310 | | | | | | | | | | | | | | | | | | | | |
| 1572 | | | | | | | | | | | | | | | | | | | | |
| 2096 | | | | | | | | | | | | | | | | | | | | |
| 2620 | | | | | | | | | | | | | | | | | | | | |
| 3144 | | | | | | | | | | | | | | | | | | | | |
| 3668 | | | | | | | | | | | | | | | | | | | | |
| 4192 | | | | | | | | | | | | | | | | | | | | |
| 4716 | | | | | | | | | | | | | | | | | | | | |
| 5240 | | | | | | | | | | | | | | | | | | | | |
| 5764 | | | | | | | | | | | | | | | | | | | | |
| 6288 | | | | | | | | | | | | | | | | | | | | |
| 6812 | | | | | | | | | | | | | | | | | | | | |
| 7336 | | | | | | | | | | | | | | | | | | | | |
| 7860 | | | | | | | | | | | | | | | | | | | | |
| 8122 | | | | | | | | | | | | | | | | | | | | |
| **500** | 174 | 349 | 524 | 1048 | 2096 | 4192 | 6288 | 7336 | 8384 | 9432 | 10480 | 12576 | 14672 | 15720 | 16768 | 23056 | 29344 | 35632 | 41920 | 48208 |

Figure 4.19 20 x 20 Matrix of values for the size workload parameter

*rate*

To characterise the uplink behaviour of the SUT for different SE rates, we select only one SE sizes that is assumed the MTU of the SUT: 524 Bytes. The generated *rate* of the SE offered to the SUT's SAP is calculated based on a percentage of the (theoretical) maximum uplink capacity of the SUT. This percentage is known as the *saturation factor* (a.k.a. traffic intensity or load factor) of the uplink link. We use 10 different *saturation factors*: 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.5 and 2.0. The increments of the factors are chosen such that the behaviour of the SUT is disclosed for "normal", "medium to high" and "overloaded" situations.

| **500** | 174 | 1572 | 2096 | 7860 | 8384 | 16244 | 16768 | 48732 |
|---|---|---|---|---|---|---|---|---|
| 174 | | | | | | | | |
| 524 | | | | | | | | |
| 1048 | | | | | | | | |
| 1572 | | | | | | | | |
| 2096 | | | | | | | | |
| 4192 | | | | | | | | |
| 6288 | | | | | | | | |
| 7860 | | | | | | | | |

Figure 4.20 8 x 8 Matrix of values for the size workload parameter

The workload generator will generate a sequence of the specified SE for a time interval of 40 seconds[35]. The number of generated SE's will match the calculated rate and the SE's are evenly distributed over a time span of one second. The workload generator is not anticipating any acknowledgement from a receiving entity. Hence, the workload generator is anticipating an unconfirmed service from the SUT.

*number*

The *number* of workload generator instances (i.e. service users) discloses the behaviour of the SUT for concurrent operating workload generators in one UMTS cell. The numbers chosen are: 1, 5 and 10. The 8 x 8 matrix is used to specify the *size* of the generated SE's and the *confirmed service* workload parameter is used as a service type.

[35] The MobiHealth system supports the use of Evaluation Data Records (EDR). Each record represents the gross number (before data compression) of Bytes forwarded by a MBU to the transmission systems SAP over a time period of 20 seconds. We wanted to be sure that the *rate* measurements could be compared to the EDR data, therefore a double time period of 40 seconds was chosen.

*unconfirmed service* and *confirmed service*

These workload parameters are only used in conjunction with the *size*, *rate* and *number* parameters (see above).

Figure 4.21 presents a resume of the workload parameters and their values.

| Parameter class | Parameter sub-class | Workload parameters values |
|---|---|---|
| *system usage related parameters* | *workload* | size:<br>20 x 20 matrix,<br>8 x 8 matrix and<br>524 Bytes<br>saturation factor (calculated rate):<br>0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.5 and 2.0<br>number:<br>1, 5 and 10<br>service type:<br>"confirmed service" and<br>"unconfirmed service" |

Figure 4.21 Workload parameter values

System parameters

System parameters are "system parameters of influence" which are classified as system description related parameters. For every performance measurement of a SoD/SUT instantiation, these parameters must not change. This creates the possibility to compare the performance measurements of different SoD/SUT instantiations and study the effect of a system parameter of influence. This approach can be used to optimize the performance (e.g. decrease of *delay*) of the SUT. Figure 4.16 specified four sub-classes of system parameters of influence:

*SUT structure,*
*SoD structure,*
*quantitative aspects of the SUT* and
*quantitative aspects of the SoD*.

For each sub-class different system parameters can be specified.

*SUT structure*

The SUT structure has a number of sub-systems that are system parameters of influence of special interest. These parameters are related to the host-to-network service part of the SUT (Figure 4.7). The influence of the "UMTS terminal" system parameter on this service is measured.

A computer system in combination with a UMTS terminal is used to provide the IP service at the SUT's SAP. Different physical media can be used to realise a communication link between the two devices. The system parameter "intra comm. system" is used to measure the effect of using different intra communication systems between a computer system and a UMTS terminal. Closely related to the "intra comm. system" parameter is the "computer system" parameter. The influence of this parameter will be measured for three different types.

The last system parameter of influence that will be studied is the APN of the SUT. The reasoning behind this choice is related to the *SoD structure* (see below).

*SoD structure*

The SoD structure consists of three sub-systems: SUT, Internet and the UTnet. The "Internet" and the "UTnet" are considered to be system parameters of influence. The mechanism to select different SoD structures (e.g. "Internet<>UTnet" and "UTnet") is realized by means of the SUT APN parameter.

*quantitative aspects of the SUT* and *quantitative aspects of the SoD*

As discussed in section 4.3.2, the SUT provides a transport service with a capacity that is correlated to the uplink and downlink volume and rate of the IP datagrams offered at the SUT SAP's. Vodafone controls the system parameters of influence responsible for the change in capacity.

System Parameters and Values

The assignment of values to the system parameters is presented in Figure 4.22. We assume theoretical values for the *quantitative aspects of the SUT* and *quantitative aspects of the SoD* parameters.

| Parameter class | Parameter sub-class | System parameters values |
|---|---|---|
| *system description related parameters* | *SUT structure* | computer system:<br>notebook (nb),<br>pc1,<br>pc2 and<br>iPAQ<br>intra comm. system:<br>USB,<br>Bluetooth and<br>PCMCIA<br>UMTS terminal:<br>Nokia 6650 and<br>PC Card<br>APN:<br>"utwente.nl" and<br>"web.vodafone.nl" |
| | *SoD structure* | computer system:<br>notebook,<br>pc1,<br>pc2 and<br>iPAQ<br>Internet:<br>"present" and<br>"not present"<br>UTnet:<br>"present" and<br>"not present" |
| | *quantitative aspects of the SUT* | uplink transmission capacity:<br>0-16 Kbps (common bearer) and<br>16-64 Kbps (dedicated bearer)<br>downlink transmission capacity:<br>0-16 Kbps (common bearer),<br>16-64 Kbps (dedicated bearer 1),<br>64-128 Kbps (dedicated bearer 2) and<br>128-384 Kbps (dedicated bearer 3)<br>number of concurrent service users per UMTS cell:<br>1,<br>5 and<br>10 |
| | *quantitative aspects of the SoD* | application buffer sizes (above transport service):<br>send - 32 KBytes and 64 KBytes<br>receive - 32 KBytes and 64 KBytes<br>TCP (socket) buffer sizes:<br>send - 32 KBytes and 64 KBytes<br>receive - 32 KBytes and 64 KBytes |

Figure 4.22 System parameter values

# 5 Performance Measurements Design

This chapter focuses on the second phase of the methodology presented in section 3.4. It covers phases 6, 7 and 8 (partially) of the methodology (Figure 5.1).

*Note: We strongly recommend the reader to familiarise with concepts presented in chapter 4 before continue reading this chapter.*

| | |
|---|---|
| *1. State the Goals and System Definition* | |
| *2. List Services and their Outcomes* | |
| *3. Select Performance Criteria (i.e. Metrics)* | |
| *4. List System and Workload Parameters* | |
| *5. Select Factors and their Levels* | |
| ***6. Select System and Workload Parameters*** | |
| ***7. Design and Execute the Experiments*** | |
| ***8. Analyse, Evaluate and Interpret the Data*** | |
| | *a. Select Model Representation* |
| | *b. Parameterise the Model* |
| | *c. Validate and Verify the Model* |
| *9. Present the Results* | |

Figure 5.1 Performance evaluation methodology phase 2

Chapter 4 provides the concepts of SoD and SUT, but we did not apply them to a concrete instance of a SoD. Moreover, we gave the system and workload parameters and their values, but did not select the parameters for particular SoD instances that we are going to evaluate.

Section 5.1 covers phase 6 of the our methodology: selection of system and workload parameters. It presents concrete instances of SoDs, which are feasible and relevant for the execution of reproducible performance measurements and evaluation of real-world instances of these SoDs.

Section 4.4 introduces an abstract description of the evaluation system and its major components. After we have defined the evaluation system's requirements and discovered the unavailability of an off-the-shelf tool (i.e. the evaluation system), we took the decision to develop our own tool. Therefore, section 5.2 describes a functional model of the developed evaluation system.

Section 5.3 provides the most important evaluation system's implementation details. It includes the way the evaluation system interacts with SoDs in general. We consider the design and implementation of the evaluation system as a separate activity, which is outside the scope of our measurement methodology presented in Figure 5.1.

The identification of the SoD instances and provisioning of an evaluation system as a tool to instrument[36] the measurements, provide the basis for section 5.4. This section describes the design and execution of various experiments. These activities cover phase 7 of our methodology.

Section 5.5 provides the preliminary analysis and evaluation of the derived measurements data, and comprises (partially) phase 8 of our methodology. The most important evaluation criterion is a proper time synchronisation of the evaluation system component(s). We evaluate the measurements timing (i.e. the SoD events) based on visualisation of measurements data and on the practical experience gained during the experiments execution. This section also highlights the encountered problems and corresponding solutions.

## 5.1 SoD Instances

The previous chapter presents the concepts of SoD and SUT. This section provides the SoD selection criteria (section 5.1.1) followed by the description of concrete SoDs instances (section 5.1.2). Section 5.1.3 provides the specification of workload combinations and SoDs instances that are subject to the performance measurement.

### 5.1.1 Selection Process

Chapter 4 presents various combinations of system parameters, but not all the combinations are important or relevant for us. Therefore, this subsection presents the parameters' selection criteria, and the selected system parameters.

The SoD (internal) implementation details depend on the system (i.e. the SoD) parameters selection. For example, which Internet subsystem in the SoD is used, depends on the chosen APN (the SoD parameter) in the V3GNL network (the SUT). We can indicate a number of the SoD instances, depending on the system parameters' selection. Due to e.g. time constraints, we selected a limited number of combinations of system parameters. The selection criteria of the SoD instances of interest were:

1) The resource equipment constraint: there was a limited amount of equipment available, e.g. only one notebook (nb) and two PCs (pc1, pc2). Moreover, some of the combinations of the system parameters (hardware, software platforms and technologies) were practically impossible. For example, the iPAQ platform does not offer: a) a USB interface or b) a PCMCIA interface for the Vodafone Mobile Connect UMTS/GPRS Card due to the lack of drivers for the platform's OS.

[36] The measurement instrumentation comprises "*the use or application of instruments to perform measurements, where instruments are measuring devices determining the present value of a quantity under observation*" [Webster].

2) The priority criteria: some of the SoD instances were more important to study than others. This was because these instances could disclose the contribution of Internet and UTnet subsystems' delay in the SoD's delay performance characteristics. Moreover, some of the SoD instances were important for the MobiHealth project (e.g. MBU in BAN uses currently the iPAQ platform).

Figure 5.2 presents the result of the selection of the SoD instances provided the above constraints and prioritisation.

| **computer systems** | nb, pc1 | nb, pc1 | nb, pc1 | nb, pc1 | nb, pc1 | nb, pc1, pc2 | iPAQ, pc1 | nb, pc1 |
|---|---|---|---|---|---|---|---|---|
| **intra comm.** | USB | USB | USB | Bluetooth | PCMCIA | USB | Bluetooth | USB |
| **UMTS terminal** | Nokia 6650 | Nokia 6650 | Nokia 6650 | Nokia 6650 | PC Card | Nokia 6650 | Nokia 6650 | Nokia 6650 |
| **APN** | utwente.nl | utwente.nl | utwente.nl | utwente.nl | utwente.nl | utwente.nl | utwente.nl | web. vodafone. nl |
| **buff. sizes: appl.sock** [KBytes] | 64.64 | 32.64 | 32.32 | 64.64 | 64.64 | 64.64 | 64.64 | 64.64 |
| the SoD instance | **SoD_1** | **SoD_2** | **SoD_3** | **SoD_4** | **SoD_5** | **SoD_6** | **SoD_7** | **SoD_8** |

Figure 5.2 The SoD system parameters selection

The meaning of the particular SoD system parameters selection is as follows. The computer system at the ingress point of the SoD (nb or iPAQ) uses intra-communication (USB, Bluetooth or PCMCIA) to access the UMTS terminal (Nokia 6650 or PC Card). This computer system accesses, through the terminal, the SUT. The APN parameter determines a part of the internal infrastructure of the SoD (i.e. presence of the public Internet component). At the SoD's egress point there is a computer system: pc1 or pc2. All computer systems used in the SoD setup have the same programmable buffer sizes (as indicated in figure).

The internals of the SUT were not accessible for the measurements. Recall that the SUT's ingress and egress SAPs instrumented for measurements are the SoD's ingress and egress SAPs (chapter 4). Furthermore, the SoD's ingress and egress SAPs are the SAPs of the transport service offered by the transport system. These SAPs are then at the transport layer and they are located at the (mentioned above) computers systems. The transport system is distributed over the multiple parties, i.e., the SoD's ingress and egress SAPs are in different geographical locations, consequently the measurement instrumentation is distributed. Section 5.2 will explain this in more detail.

### 5.1.2 Instances' Description

The previous section described the selected SoD instances by means of system parameters' sets, while this section identifies these instances (i.e. system parameters configurations) in detail, including the delineation of the SUT and the SoD. Figure 5.3 depicts all possible instances in a single overview. The text below explains the differences between these instances, starting from SoD_8 followed by SoD_1 to SoD_7. The reason of this order is that, in contradiction to other instances, the instance SoD_8 is commercially available.

**SoD_8**

The commercial implementation of the SUT for instance SoD_8 is available for all SUT users. The APN *web.vodafone.nl* provides access to the Internet. In this configuration, at the ingress point of the SUT there is a notebook (computer system nb). The notebook is connected to the SUT through a Nokia 6650 terminal (a USB cable connects the terminal the notebook). The notebook generates data, which is transported by the SUT and received by a PC at the UTnet (computer system pc1). We indicate the data path in Figure 5.3 in green. Both computer systems are configured with a 64 KBytes application buffer and a 64 KBytes TCP (receive/send) buffer size.

The SUT has the following features:

1) it is a public/shared resource

2) it provides a "best effort" service

3) it has an asymmetrical and multilevel (with capability to switch between the contiguous levels) link capacity:

| | |
|---|---|
| uplink capacity | 16 or 64 Kbps |
| downlink capacity | 16 or 64/128/384 Kbps |

4) it assigns the "private" Class A IP address (e.g. 10.77.86.235) for the system attached at its ingress point; the system attached to the SUT is not accessible from the outside of the SUT.

The Internet sub-system in this SoD provides an IP service at its boundaries. IP router interfaces implement this service. Inside the Internet cloud, we distinguish sub-networks of various providers. Figure 5.4, shows for example how the *tracert* command reveals the UUNet and SURFnet ISPs core networks.

The UTnet has an IP-based interface at its boundaries. The UTnet is connected with a direct connection to the SURFnet ISP core network (1 Gb link capacity) and hosts fast (100 Mbps) departmental networks (LANs). The UTnet uses the IP addresses range 130.89.0.0 - 130.89.255.255.

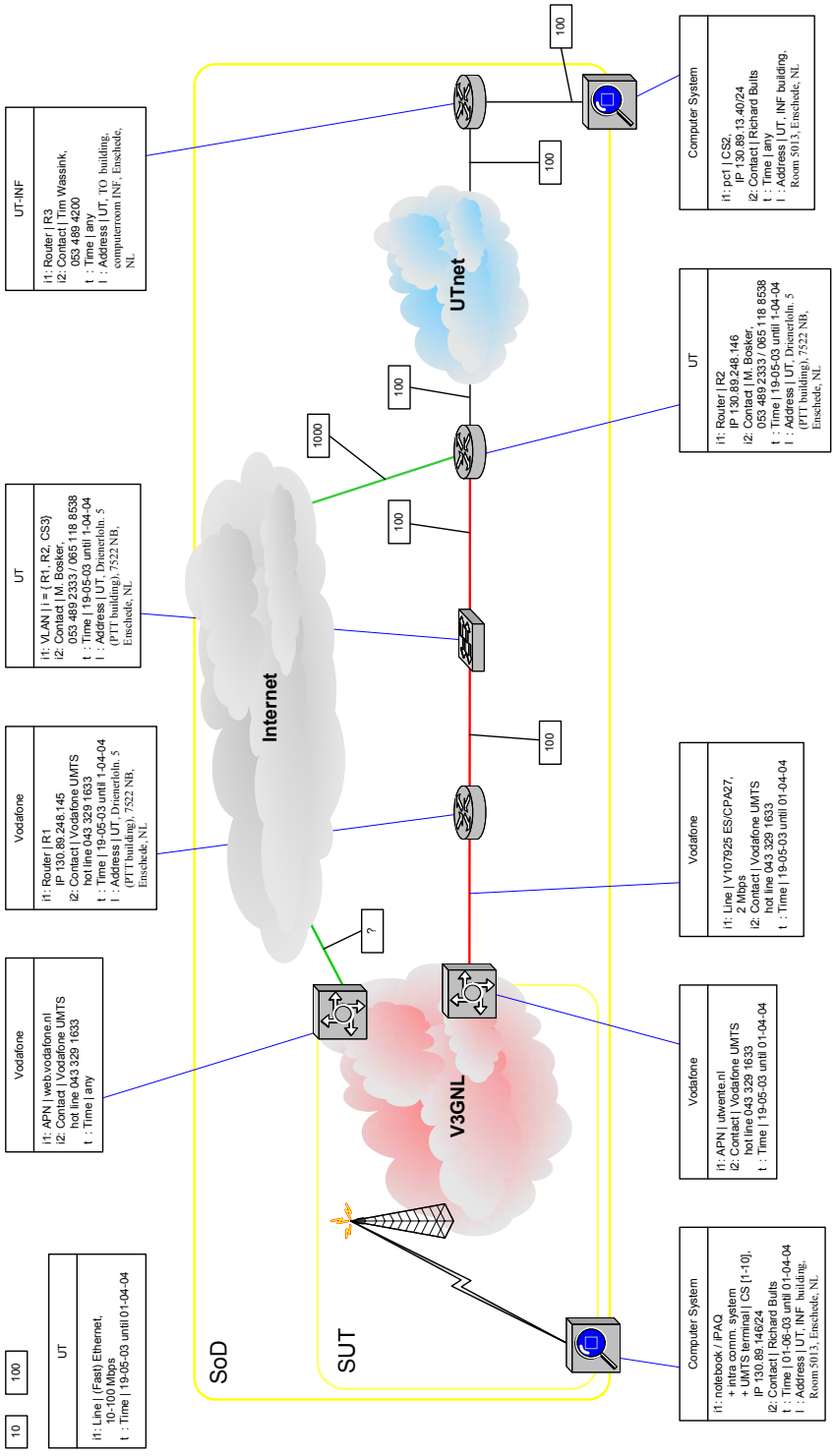



Figure 5.3 SoD instances

| Trace route[37] | | | RTT [ms][38] |
|---|---|---|---|
| 1 | simplon (130.89.10.1) | | 0.559 / 0.305 / 0.389 |
| 2 | if-cs.routing.utwente.nl | (130.89.254.113) | 0.353 / 0.547 / 0.309 |
| **UTnet / SURFnet** | | | |
| 3 | Gi7-0.AR5.Enschede1.surf.net | (145.145.4.1) | 0.313 / 0.330 / 0.265 |
| 4 | PO2-0.CR2.Amsterdam2.surf.net | (145.145.165.13) | 163.091 / 209.073 / 30.375 |
| 5 | PO1-0.CR1.Amsterdam2.surf.net | (145.145.160.13) | 4.056 / 4.004 / 3.963 |
| 6 | PO15-0.CR1.Amsterdam1.surf.net | (145.145.160.17) | 5.289 / 5.132 / 5.070 |
| 7 | P0-0.BR1.Amsterdam1.surf.net | (145.145.166.34) | 31.682 / 198.162 / 6.596 |
| 8 | POS4-1.BR1.AMS3.ALTER.NET | (145.145.166.74) | 4.998 / 4.954 / 4.903 |
| **SURFnet / UUNet** | | | |
| 9 | so-0-2-0.TR2.AMS2.ALTER.NET | (146.188.3.217) | 5.153 / 5.181 / 5.164 |
| 10 | so-6-0-0.XR2.AMS6.ALTER.NET | (146.188.8.89) | 5.160 / 5.206 / 5.157 |
| 11 | so-0-0-0.CR2.AMS7.ALTER.NET | (212.136.176.102) | 5.311 / 5.290 / 5.286 |
| 12 | 312.ATM0-0-0.GW1.AMS7.ALTER.NET | (212.136.176.138) | 5.614 / 5.612 / 5.587 |
| 13 | Vodafone-gw.customer.ALTER.NET | (146.188.37.130) | 7.522 / 6.031 / 6.113 |
| 14 | * * * | | |

Figure 5.4 A path from the UTnet to the SUT through an Internet [39]

### SoD_1

The dedicated implementation of the SUT for configuration SoD_1 is available only for users associated to the UT. Comparing to the SoD_8 instance, this SoD excludes the Internet sub-system. Due to the characteristics of the *utwente.nl* APN SoD (mentioned in following paragraph), we used it in different configurations in majority of measurements. The computer systems accessing the SoD are configured as is indicated in Figure 5.2. At the ingress point of the SUT, there is a notebook (computer system nb). The notebook is connected to the SUT through a Nokia 6650 terminal (a USB cable connects the terminal the notebook). The notebook generates data, which is transported by the SUT and received by a PC at the UTnet (computer system pc1). We indicate the data path in Figure 5.3 in red. Both computer systems are configured with a 64 KBytes application buffer and a 64 KBytes TCP (receive/send) buffer size.

For the period of 8th July 2003 – 1st April 2004 a leased line[40] directly connected the V3GNL (i.e. the SUT) and the UTnet (see Appendix E for a schematic overview). Hence, there was a direct IP-based interface between these two sub-systems. A leased line is a telephone line rented for a private use from the (Dutch) telephone carrier KPN.

Features of the leased line are:

1) it is a dedicated resource

2) it provides the bandwidth guarantees (form of QoS); leased lines tend to only suffer from physical rather than operational failures

3) it has a symmetrical link capacity: uplink/downlink capacity 2048 Mbps (E1).

[37] Used *tracert* program under the Windows XP OS [Tracert]
[38] RTT between the node, initiating the trace and the receiving node for the three separate journeys. RTTs are not cumulative along the trace.
[39] APN *web.vodafone.nl*, Sending node: notebook, receiving: Vodafone's infrastructure node: 62.140.137.62
[40] A private communications channel leased from a common carrier.

Due to the direct connection between the SUT and the UTnet, the IP address assignment to the system attached to the SUT is different from the previous case. Namely, each system attached to the SUT via *utwente.nl* APN gets the Class B IP address in address range of the UTnet, for example: 130.89.146.90. That means that a system attached to the SUT is accessible from the outside of the SUT.

Based on the (one possible) instance of the route presented in Figure 5.5, we identified the UTnet and the SUT's routers at the boundary of the UTnet.

| Trace route | | | RTT [ms] | | |
|---|---|---|---|---|---|
| 1 | wurzen.cs.utwente.nl | 130.89.13.1 | <1 | <1 | <1 |
| 2 | if-cs.routing.utwente.nl | 130.89.254.113 | <1 | <1 | <1 |
| **UTnet / Vodafone** | | | | | |
| 3 | | 130.89.248.145 | 6 | 5 | 5 |
| 4 | | 130.89.248.141 | 6 | 6 | 6 |
| 5 | | 130.89.248.129 | 346 | 335 | 345 |
| 6 | | 130.89.146.90 | | | |

Figure 5.5 A path instance from the UTnet to the SUT through the leased line[41]

As mentioned in the previous section, the UTnet has an IP-based interface at its boundaries. In the SoD_1 case, the UTnet has a direct connection to the SUT core network.

**SoD_2** to **SoD_7 –** alternatives of **SoD_1**

The configurations for SoD_2 to SoD_7 all use the *utwente.nl* APN, but differ in other system parameters, such as the configuration of the computer systems.

The notebook (computer system nb) in SoD_2 is connected to the SUT through a Nokia 6650 terminal (a USB cable connects the terminal the notebook). The notebook generates data, which is transported by the SUT and received by a PC at the UTnet (computer system pc1). Both computer systems are configured with a 32 KBytes application buffer and a 64 KBytes TCP (receive/send) buffer size.

SoD_3 is identical to SoD_2 except that both computer systems we configured with a 32 KBytes application buffer and a 32 KBytes TCP (receive/send) buffer.

SoD_4 is identical to SoD_2 except that the USB cable we replaced with a Bluetooth connection. The application buffer and the TCP (receive/send) buffer are both of 64 KBytes size.

SoD_5 is identical to SoD_4 except that the notebook connects to the VG3NL network with a PCMCIA card.

[41] APN *utwente.nl*, Sending node: notebook, receiving: UMTS node: 130.89.146.90

In SoD_7 a iPAQ computer system is connected to the SUT through a Nokia 6650 terminal (there is a Bluetooth connection between the terminal the iPAQ). The iPAQ generates data, which is transported by the SUT and received by a PC at the UTnet (computer system pc1). Both computer systems are configured with a 64 KBytes application buffer and a 64 KBytes TCP (receive/send) buffer size.

Figure 5.6 (instance SoD_6) shows an implementation of the service view as presented in Figure 4.12-B. The notebook (computer system nb) is connected to the SUT through a Nokia 6650 terminal (a USB cable connects the terminal the notebook). The SUT transports the data generated by the notebook. A PC (computer system pc2), placed in between the SUT and the UTnet, receives the data from the SUT. Consequently, pc2 forwards the data to the PC (computer system pc1) placed in the UTnet. We configured all computer systems with a 64 KBytes application buffer and a 64 KBytes TCP (receive/send) buffer size.

We placed pc2 exactly in between Vodafone's router (i.e. the SUT egress point) and the UTnet router (thus creating the VLAN). It is because we wanted to have a measurement point as close as possible to the egress point of the SUT. This configuration allows us to quantify the contribution of the UTnet subsystem to the delay performance characteristics of the SoD. This configuration also implies the split of SoD_6 into two sub-systems: 1) the SUT and 2) the UTnet.

According to our measurement methodology, the SoD should contain the SUT (SoD_6' in Figure 5.6). We also consider the UTnet sub-system and treat it according to our measurement methodology. We consider the second subsystem as a separate (another) SoD (SoD_6'' in Figure 5.6). SoD_6'' contains the UTnet as the SUT. During our measurements activity, we instrumented both SoDs and took measurements at the ingress and egress points of both SoDs. From these measurements, we derived the performance characteristics of both SoDs and the influence of separate parts of the SoD on its characteristics. Performance measures of interest are the same for both SoDs. Parameters of influence in the SoD_6'' are different from parameters of influence in the SoD_6', but this consideration is beyond of the scope of this project.

### 5.1.3 Workload Selection

In chapter 4, we defined the workload parameters. However, we did not specify on which SoDs particular workloads will be examined, i.e., what are the combinations of the workload and the system parameters that we are going to measure. This section provides combinations selection criteria. The selected combinations are presented in the matrix.

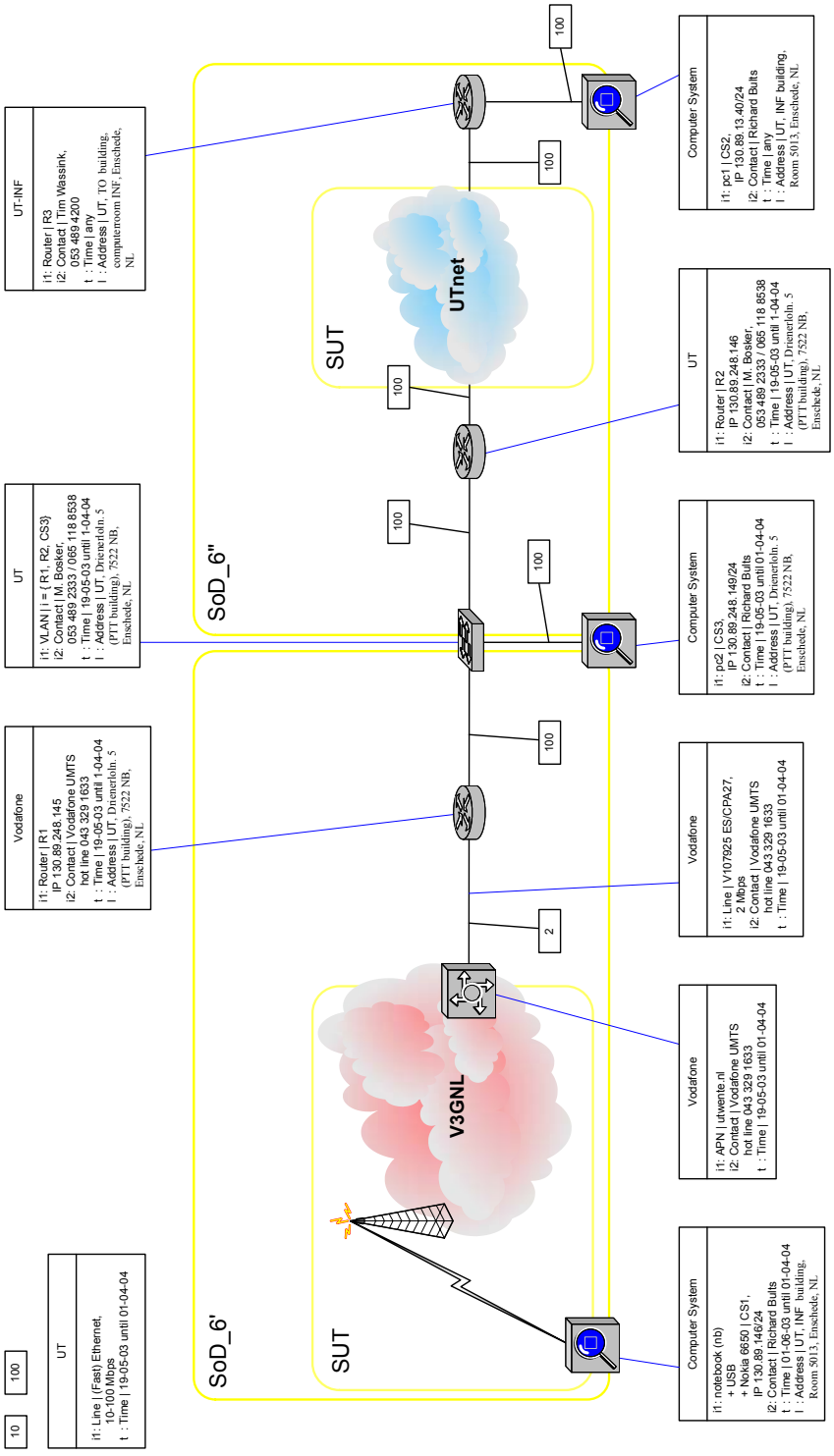



Figure 5.6 Instance SoD_6

Ideally, the measurements should be performed in such way, that all possible combinations of the workload and system parameters (i.e. SoDs) are taken into account. That would give 19.200 different experiments[42]. Moreover, each experiment needs to be repeated a number of times in order to get statistically sound data for the obtained system (i.e. the SUT) performance parameters. Due to resource constraints, we took into account only a limited number of combinations, such that we could obtain the most relevant data with the minimum number of experiments.

Our selection criteria of the combinations of the workload parameters and the SoD instances of interest were:

1) the resource constraints:

   a) the time constraint
   We had 45 days available for measurements and following Jain's note [Jain1991], we anticipated the worst-case scenario: "*Murphy's law strikes measurements more often than other [evaluation] techniques [i.e. analytical modelling or simulation]. If anything can go wrong, it will. As a results, the time required for measurements is the most variable among the three [evaluation] techniques*". The worst-case scenario would mean the repetition of the complete set of measurements, which resulted in estimation of around 23 days available for effective measurements.

   b) the personnel constraint
   There were two people available for 80% of their time.

   c) the equipment constraint
   We had a limited number of equipment available, e.g., only one notebook platform. Consequently, we could not execute the scalability test with this platform.

2) the priority criteria:

   some of the combinations were of higher importance than others. We required the benchmark[43] measurement for the confirmed service case (see section 4.7). For the unconfirmed service case, we selected only the workload of packet size 524 Bytes.

Figure 5.7 presents the result of the selection of the combinations of the SoDs and workload parameters. A cross in the cell means, that the combination of parameters given by the cell are included in the measurement activity. Section 4.7 defines the parameters presented in this figure.

[42] (8 SoDs) x (20 * 20 matrix + 8 * 1 matrix * 10 saturations) x ( 3 No of users) = 19200
[43] Benchmark is selected to be run on a particular SoD instance with the goal of fair comparison of the performance of other SoD instances. Benchmark characteristics are: fair (minimized bias of a specific platform), relevant, easy to measure, easy to explain.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **computer systems** | | nb, pc1 | nb, pc1 | nb, pc1 | nb, pc1, pc2 | iPAQ, pc1 | nb, pc1 | nb, pc1 | nb, pc1 | nb, pc1 |
| **intra comm.** | | USB | Bluetooth | PCMCIA | USB | Bluetooth | USB | USB | USB | USB |
| **UMTS terminal** | | Nokia 6650 | Nokia 6650 | PC Card | Nokia 6650 | Nokia 6650 | Nokia 6650 | Nokia 6650 | Nokia 6650 | Nokia 6650 |
| **APN** | | utwente.nl | utwente.nl | utwente.nl | utwente.nl | utwente.nl | web.vodafone.nl | utwente.nl | utwente.nl | utwente.nl |
| **buff. sizes: appl.sock** | | 64.64 | 64.64 | 64.64 | 64.64 | 64.64 | 64.64 | 64.64 | 32.64 | 32.32 |
| the SoD instance / work-load par | | **SoD_1** | **SoD_4** | **SoD_5** | **SoD_6** | **SoD_7** | **SoD_8** | **SoD_1** | **SoD_2** | **SoD_3** |
| **Service type** | | Confirmed service | | | | | | Unconfirmed service | | |
| **Sample size**[44] | | 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 |
| **Matrix** | | 20x20 | 8x8 | 8x8 | 8x8 | 8x8 | 8x8 | 524 Bytes | 524 Bytes | 524 Bytes |
| **Saturation factor** | 0.5 | | | | | | | x | x | x |
| | 0.6 | | | | | | | x | x | x |
| | 0.7 | | | | | | | x | x | x |
| | 0.8 | | | | | | | x | x | x |
| | 0.9 | | | | | | | x | x | x |
| | 1.0 | | | | | | | x | x | x |
| | 1.1 | | | | | | | x | x | x |
| | 1.2 | | | | | | | x | x | x |
| | 1.5 | | | | | | | x | x | x |
| | 2.0 | | | | | | | x | x | x |
| **No** | 1 | x | x | x | x | x | x | x | x | x |
| | 5 | | | | | x | | | | |
| | 10 | | | | | x | | | | |

Figure 5.7 System and associated workload parameters

*Note: Appendix D provides a detailed specification of the equipment used for the measurement activity.*

## 5.2 Evaluation System Functionality

The functionality of the evaluation system presented in chapter 4 needs further refinement. This section presents the evaluation system's design and implementation requirements (section 5.2.1). Following consecutive sections (5.2.2 to 5.2.5) present the functional building blocks of the implemented evaluation system.

[44] An observation is an execution of a SE (e.g. unconfirmed or confirmed service instance). The number of observations in sample is 500 to get statistically sound data (see section 5.5.1 for arguments).

### 5.2.1 System Requirements

The evaluation system as presented in chapter 4 consists of a workload generator and a measurement function (Figure 4.10). However, we did not specify further the functional details of the evaluation system, and particularly the way it interacts with the SoD. This section further discusses the design and functional requirements of the evaluation system.

The following evaluation system's instrumentation requirements are identified:

1) no cooperation between the workload generator and the measurement function components of the evaluation system[45]

2) full control (carried out by the system) of the interactions between the workload generator and the SoD

3) as little as possible influence of the measurement function on the behaviour of the SoD

4) the evaluation service's end-user can provide/change the workload and system parameters

5) the workload generator has the functionality to generate different workload parameters values automatically

6) the evaluation system has the functionality for (automatic) clock's synchronisation

7) the measurement function has the functionality to (automatically) monitor events and (automatically) correlate measurements to workload parameters

8) the measurement function has the functionality to store the measurement data for asynchronous data retrieval

9) the evaluation system has a simple error handling functionality

The implementation of the evaluation system is started by a survey of existing workload generators or measurement tools, to check if they meet our evaluation system's requirements. It seemed not very likely that we could find an existing workload generator and distributed measurement function that matched our requirements. We provide the following analysis on tools we have considered:

1) *tcpdump* and *ttcp* tools can act as workload generators but the automated generation of different workload parameter values is not trivial to instrument (do not meet requirement no. 5).

[45] Workload generator interacts with the SUT i.e. influences the behaviour of the SUT, but measurement system should not

2) Using *Ethereal* tool as a component for a measurement function creates serious problems with the correlation of measurement data and the workload parameters. Particularly, the correlation of an application-level SP timestamp with the Ethereal TCP segments' timestamps is difficult. Another problem is the automatic administration of the timing data in correlation to the generated workload. The conclusion on the Ethereal analysis is that the tool may be useful as the evaluation system, but it is not easy to configure it such it meets all the system requirements.

The results of the survey on available tools did not provide us with any off-the-shelf tool. Our requirements were very specific. Consequently, we made the decision to design and implement our own evaluation system. The next step is to identify which functionality needs to be implemented.

Recall section 4.4 (Figure 4.10), the evaluation system consists of a workload generator function and a measurement function. An evaluation system interacts with a SoD at the SoD's SAP(s). An evaluation system provides stimuli to a SoD by means of generating workload and it measures the SoD's response to those stimuli by means of the measurement function.

Figure 5.8 presents the high-level functional model of the evaluation system delivering the evaluation service. There are three functional building blocks: *SoD*, *evaluation service* and *end-user functions*. The *SoD* block delivers the transport service (we presented its functionality in detail in chapter 4).

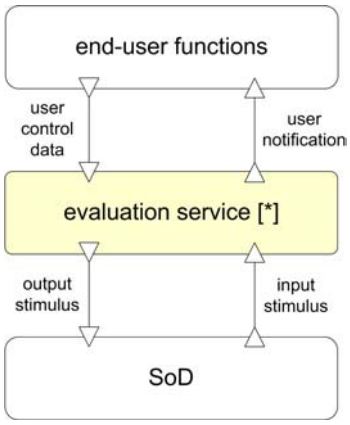



Figure 5.8 Evaluation service functional model

The *evaluation service* block comprises the functionality needed to provide the evaluation service. The evaluation service consists of the evaluation of the behaviour *SoD* block functionality. This is the primary function of this functional block. The *evaluation service* influences the behaviour of the *SoD* by generation of stimuli to it and the evaluation service measures the *SoD*'s reaction to this stimuli. The *evaluation service* is also able to

react to the stimuli from *SoD*. Due to the distributed character of the SoD, there can be a number of evaluation system instances, examining the SoD simultaneously.
The secondary function of the *evaluation service* is provision of the *end-user functions* that facilitate the interaction between the end-user and the service, and therefore deal with:

1) the evaluation service requests' handling; an end-user provides particular workload and system parameters (i.e. user control data)

2) end-user notification on the state of the evaluation service execution (e.g. completed, interrupted due to the error, etc.)

Our analysis of the high-level functional view on the evaluation system resulted in the following evaluation system's design requirements:

1) due to the nature of the service provided by the SUT (IP service) the evaluation system needs to be running on top of an IP entity and it needs to interact with the SUT by the communication means like e.g. IP-based network infrastructure.

2) the evaluation system must be implemented using standard hardware/software platforms

3) the evaluation system must be portable (interoperable) amongst the platforms specified in requirement 2)

Moreover, we took into account the following resource constraints while implementing the evaluation system:

4) equipment constraint: the evaluation system implementation is constrained by available (and known by the system users) hardware/software platforms

5) time constraint: the evaluation system needs to comprise the functionality needed to evaluate the SUT; no additional requirements like e.g. system's robustness are to be implemented

We analysed the evaluation system requirements and considered the availability of equipment, and particularly lack of a specialized hardware platform for the evaluation system. Consequently, we decided to develop of our own evaluation system, which is portable amongst different hardware platforms ("write once, run anywhere"). We implemented the evaluation system in the Java programming language[46] and as a Java application; i.e. Java program that is run standalone[47] at (any) platform [TechWeb]. The application is the means by which the evaluation system provides the evaluation service and runs on a networked machine (IP entity). Reliable TCP sockets[48] facilitate the interaction between the evaluation system and service provided by SoD.

An end-user can set the workload and system parameters. There is no GUI implemented, we assume that the end-user knows the Java programming language, such that he/she can change the parameters provided to the evaluation system (i.e. by changing part of the Java source code).

The evaluation system provides functionality as shown in Figure 5.8. Following sections provide the details on functionality provided by the evaluation system implemented as a Java application. We base the structure of sections on the process of 'step-wise refinement'. This means that we present a high-level functional design model of the evaluation service and introduce the main functional building blocks. Paragraphs describe each individual functional building block in general terms and provide a refinement. We choose the level of refinement such that a high-level design of the individual blocks is provided on the suitable abstract level.

### 5.2.2 Evaluation Service

The *evaluation service* functional block comprises the functionality required for the interaction with the *SoD* and *evaluation service* end-user. This includes the *service logic*, *workload generator* and *measurement function* as shown in Figure 5.9.

The *service logic* function holds all functionality needed to initialize and control the external observable behaviour of the *evaluation service*. Based on the parameters provided by the end-user (i.e. *user control data*), the *service logic* initializes the *workload generator* and *measurement function*s (by means of the *control data* flow).

The *workload generator* comprises the functionality to provide stimuli to the *SoD* (i.e. *output stimulus*) and reacts to stimuli from the *SoD* (i.e. *input stimulus*). It generates stimuli according to the workload parameters (i.e. *workload generator control data*). It is important to notice that from the *workload generator* and *measurement function's* point of view, the stimuli to and from the *SoD* are *messages* (there is *send message* and *receive message* flow).

---

[46] Java is an object-oriented programming language based on the virtual machine (JVM or CVM) that offers the same interface independently of the hardware or operating system [Naug1996]. The concept of Java programs is: "write once, run anywhere".

[47] The Java Virtual Machine at the machine, where the Java application is running, is interpreting the instructions.

[48] A socket is the interface between the application layer and the transport layer within the host. It is also referred to as the API between the application and the network, since the socket is the programming interface which network applications are built in the Internet [Kuro2001]

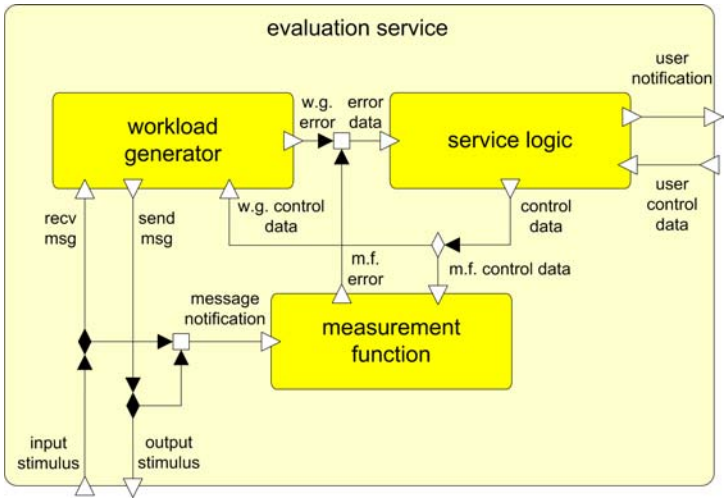



Figure 5.9 Evaluation service refinement

The *measurement function* comprises the functionality to measure the *SoD* response to stimuli based on *message notification* flow. The *measurement function* provides its functionality based on the control data input set (i.e. *measurement function control data*).

Whenever the *workload generator* or the *measurement function* encounters an error while providing their service, they raise an error notification (i.e. *workload generator error* and *measurement function error*) to the *service logic*. The *Service logic* contains the error (i.e. *error data*) handling functionality, which, for example, includes the notification to the end-user on the (erroneous) service execution result (i.e. *user notification*).

### 5.2.3 Service Logic

The *service logic* functional block is responsible for managing the evaluation service execution. It manages the service execution based on the parameters provided in user control data set (i.e. *user control data*). The *service logic* also handles the service execution status. This function contains four functional building blocks (Figure 5.10), which are grouped into three functionality groups: handling the end-user control data, handling the service execution status and ensuring the system clock synchronization. The *system parameter configuration* functional block comprises the system parameters settings provided by the end-user. These parameters are set and seen in the evaluation system as global variables.

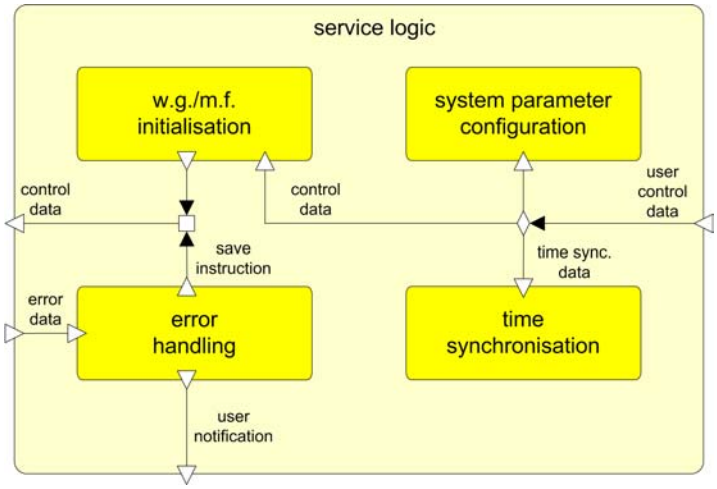



Figure 5.10 Service logic refinement

The *workload generator/measurement function initialisation* block initialises the *workload generator* and *measurement function* with the adequate parameters (i.e. *control data*).

The *time synchronisation function* plays very important role in providing the evaluation service. Based on the *time synchronization data* set it provides the automatic system clock synchronisation on a scheduled basis. The system clock is a critical in the evaluation service provisioning because, due to the distributed nature of the evaluation system, all events need to be generated and timestamped in the accurate and reliable manner.

The *error handling* functional block enables the interception of the erroneous service delivery (i.e. *error data* flow). It generates the error notification to other functional blocks and to the end-user (i.e. *user notification*). In case of an error, the *error handling* function instructs the *workload generator* to terminate and *measurement function* to save the measurement data so far and then terminate (*save instruction* flow).

### 5.2.4 Workload Generator

The *workload generator* (Figure 5.11) provides the output stimulus (i.e. *send message*) to the *SoD* by means of the *send function*. It also comprises the functionality to react to the input stimulus (i.e. *receive message*) from the *SoD* by means of the *receive function*. Both functional building blocks are provided with the respective control data (i.e. *send control data* set and *receive control data* set) by the *service logic* function. The *send function* is provided with the workload parameters while the *receive function* is provided with e.g. timeout on receive.

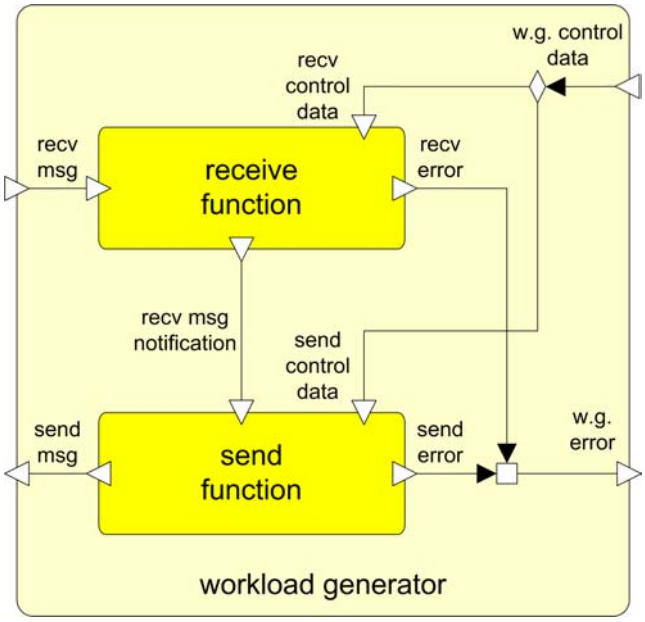



Figure 5.11 Workload generator refinement

There are two different *send* and *receive function* relations. *Receive function* triggered by the received message can stimulate (by means of *receive message notification*) the *send function* if this relation has been enabled in the receive control data set. This relation implements the confirmed service case (recall section 4.7). Otherwise, the *send* and *receive* functions provide services independently, and then the unconfirmed service is implemented.

Both functional building blocks can generate an error notification (i.e. *send error* and *receive error*) to the *service logic* when they encounter an execution problem.

**Send function**

The *send function* provides the message stimulus (i.e. *send message,* output stimulus) to the evaluation system. The *send function* gets all required information from the *workload generator* functional block by means of the *send control data* set. The *send function* consists of three functional building blocks (Figure 5.12): *send control function*, *send message function* and the *timer function*.

The *send control function* cumulates all the information on what (i.e. *workload parameters*) and when (i.e. with which rate, triggered by the *receive function* notification or not) is generated by the *send message function*. In the *send message notification* the *send control function* provides message parameters to the *send message function* and therefore it triggers the *send message*.

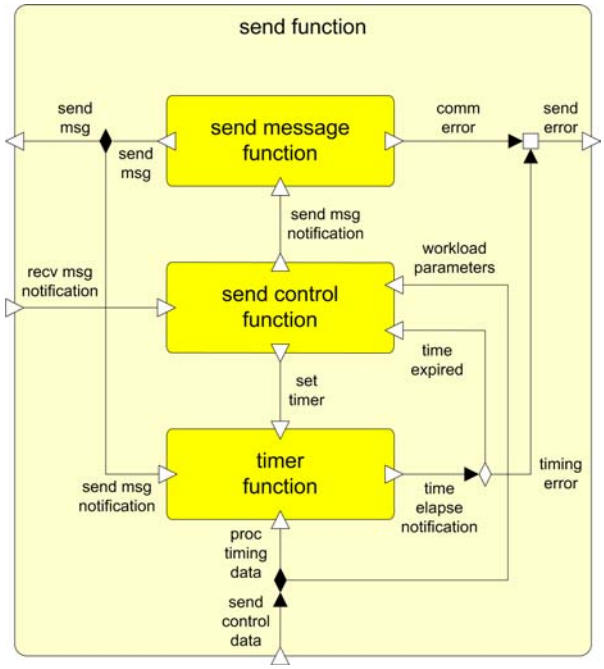



Figure 5.12 Send function refinement

An end-user can setup sets of workload characteristics to be generated to the SoD. In this case the *send control function* needs to have a functionality of automatic selection of particular characteristics from the set (i.e. changing the current characteristics), such that the whole set of characteristics' values is explored. Moreover, end-user can setup the repetition number for the (same) workload characteristics, such that the *send control function* needs to send a message with the same workload characteristics number of times. It this case the counter function for a number of repetitions performs.

There are two possible scenarios for the execution of the *send control function*. The first scenario is that the *send control function* instructs the *send message function* on construction and sending of a message and it instructs the *timer function* on timeout to receive notification from the receiver (i.e. *set timer* data flow). In this scenario, the trigger to generate and send a new message is a *receive message notification*. This is a confirmed service implementation. The timer starts timing at the moment, when the message is send. The *send message notification* is a trigger for the *timer function*. If the *receive message notification* is not received by the *timer function* (from the *send control function*) within this particular time interval since sending a message, the *timer function* raises the timing error. The timing error is raised as the *time elapse notification* and explicit *time expired* information to the *send control function*.

The second scenario is that the *send control function* instructs the *send message function* and it instructs the *timer function* on the time interval for sending of a next message. In this scenario, the *send message function* generates messages on a scheduled basis, i.e., messages are sent with a particular rate, and this is an unconfirmed service implementation.

The functionality of the *send message function* comprises the direct interaction (i.e. *send message*) with the *workload generator* function. The *send message* is also converted to a *send message notification*. The *send message notification* resets the clock in the *timer function* (e.g. like in a stopwatch). When the *send message function* encounters communication problems while sending, it generates error notification (i.e. *communication error*).

The *timer function* times the stimulus input or output occurrence. It times the intervals between the consecutive send messages, but it also times the service time execution (i.e. the service delivery elapsed time). The latter time interval is compared with the value of the maximum service delivery time (provided by the *procedure timing data* in the *send control data* set). If the maximum service delivery time elapses, the *timer function* raises the *timing error*.

The *send function* block can generate *send error* of two kinds: communication (*communication error*) and timing error (*timing error*) to the *workload generator* function.

**Receive function**

The *receive function* fulfils the functionality of the *workload generator* to react to the messages from the *evaluation system* (input stimulus from *SoD)*. Figure 5.13 provides the *receive function*'s two functional building blocks. When the *receive function* receives a message (*receive message*), it notifies the timer function (*receive message notification*) and it passes the message to the *receive data processing* Function. It processes this message to check if a communication error occurred. The *receive data processing* block generates the notification (*receive message notification*) to the *send function* only if no communication error occurred.

The *timer function* times the intervals between the consecutive receiving of messages. If there was no message received within a particular time interval (given as a parameter in the *receive control data* set and particularly in the *procedure timing data* set), a timing error notification is generated. A *received message notification* resets the clock in the *timer function*. Analogically to the timer function in the *send function*, the *timer function* times the service time execution and if the maximum service delivery time elapses, the *timer function* raises a *timing error*.

The *receive function* raises the error notification (*receive error*) to the *workload generator* function if it encounters a communication (*communication error*) or timing error (*timing error*).

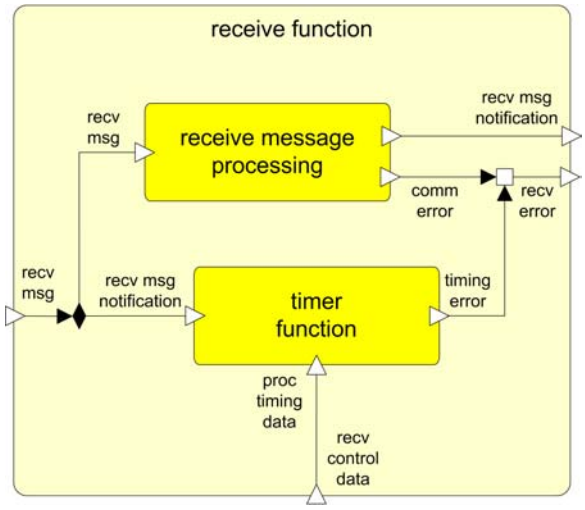



Figure 5.13 Receive function refinement

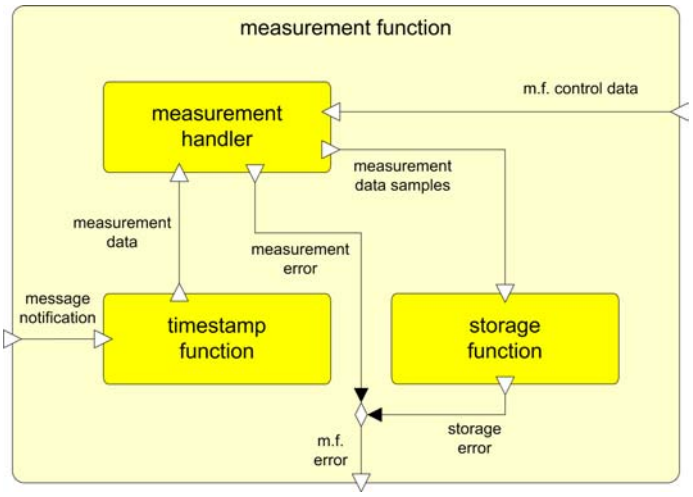



Figure 5.14 Measurement function refinement

### 5.2.5 Measurement Function

The *measurement function* comprises the functionality to measure the *SoD* response to stimuli based on the *message notification* flow. Figure 5.14 presents the functional blocks of the *measurement function*.

When the *message notification* is received, the *timestamp function* adds a timestamp to the message and forwards the "enriched" message (*measurement data*) to the

*measurements handler* function. The *measurements handler* function further processes the *measurement data*. Parameters necessary for the *measurements data* processing are given in the *measurement function control data* set provided by the *service logic* function. The processed and managed (e.g. in particular order) *measurement data* (i.e. data regarding one message) is called an *observation*. A sequence of observations gives the *sample*. The *measurements handler* function stores observations and it has a counter function in order to count them in a sample. Again, the *measurement function control data* set provides the information what is the required number of observations in the sample.

When the *measurement handler* collects the sample, it pushes the sample (*measurement data sample)* to the *storage function* together with the information on the measurement context (i.e. the workload and system parameters). The *storage function* stores the "enriched" sample together with in the data text file. In case when the *measurement handler* or the *storage function* encounters a problem, they raise the error notification (*measurement function error* consisting of the *measurement error* or the *storage error*) to the *evaluation service* function. The *measurement handler* can trigger the *storage function* to save data at any time (e.g. for a partial sample). This is in case when the *measurement handler* gets the error notification in the *measurement function control data* set. This functionality is necessary in case when the error in service delivery occurs.

**Measurement handler**

The *measurement handler* processes the *measurement data* in order to facilitate the *measurement data* storage. It contains two functional blocks: *measurement data validation* and *measurement data administration* (Figure 5.15).

The measurement data (message and its timestamp) is a trigger for the *measurement data validation* function. This function interprets the message (send or receive) and it validates message parameters within the measurement context provided by the *measurement function control data* set. Particularly, if the sequence number of the message does not conform the sequence number expected, the *data validation* function discards the measurement data and it generates the *validation error* (i.e. *measurement error*) to the *measurement function*.

If the measurement data validation process gives a positive outcome, the *measurement data validation* function forwards the data to the *measurement data administration* function. This function is responsible for the proper management of the measurement data (i.e. order of the timestamp and message parameters). This function composes the ordered *observation*. It collects the administered observations within the sample.

The *measurement data administration* function has a counter function to count the collected observations. After the sample is collected (or the function was triggered by the error indication in *measurement function control data* set), the sample together with the measurement context data (*measurement data sample*) are pushed for storage to the *measurement function.* If the *measurement data administration* encounters the problem it generates the *administration error* (i.e. *measurement error*) to the *measurement function*.

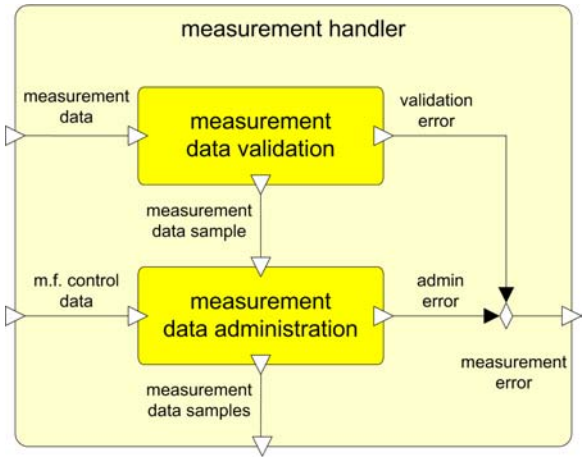



Figure 5.15 Measurement handler refinement

## 5.3 Measurements Instrumentation

The previous section presented the functionality of the evaluation system, but does not present the implementation details. Section 5.3.1 provides the (most important) evaluation system's implementation details. Section 5.3.2 provides the functionality of the system related to its distribution.

### 5.3.1 Evaluation System Implementation

To implement all the functionality needed to provide the evaluation service, we structured the Java application into *classes*. These classes we mainly built ourselves and partially we took from the (existing) class libraries. These classes provide functionalities, class *attributes* to assign parameters and *methods* to perform tasks. Recall Figure 5.9, it distinguishes three high-level functions of the evaluation system: workload generator, measurements function and service logic function. Figure 5.16 presents the class diagram of Java application, and an indication, which classes (and attributes, methods) fulfil particular functionalities of the evaluation system. Appendix F provides the explanation of the UML notation we used in this diagram. Appendix G provides the refined diagrams of classes (each class separately).

*Note: The names of the classes indicate the client-server paradigm used in the evaluation system. This paradigm supports the use of multiple instances of cooperating evaluation systems.*

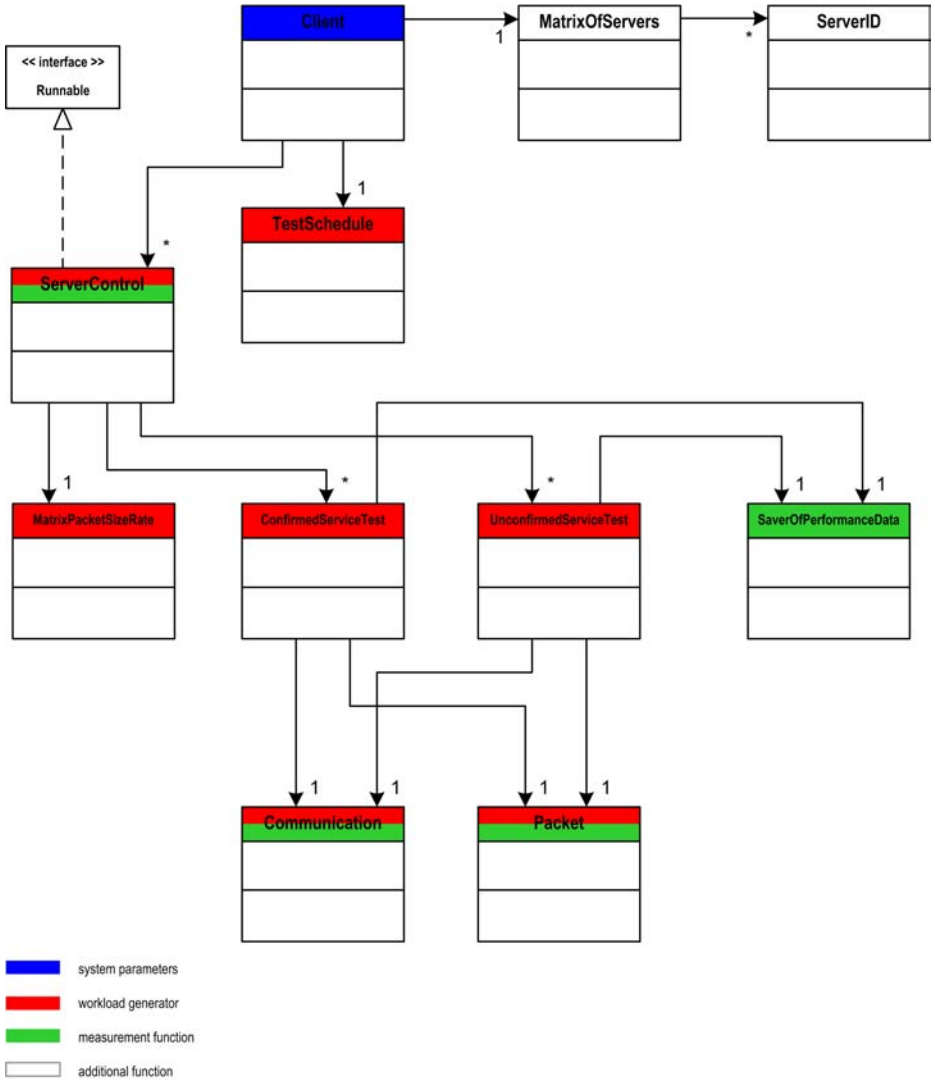



Figure 5.16 Evaluation system class diagram

An end-user provides system parameters by means of: 1) the system setup i.e. SoD instantiation (see section 5.1.1) and 2) setting of the system parameters directly in the application source code. The most important class, which holds all the parameters, is the *Client* class (see class attributes indicated in Appendix G, Figure G.1).

All workload parameters are set by the end-user in the source code. Recall section 4.7, the matrix of packet sizes (e.g. matrix 20x20 in Figure 4.19) make the basis for the workload generation. An end-user can instruct the evaluation system to start the workload generation with characteristics provided by an arbitrary entry in the matrix. An end-user

indicates these characteristics by their coordinates in matrix: row.column (where row and column are calculated from 0...19). The end-user sets this option in the *MATRIX_COORDINATES* attributes. An end-user can also setup administrative (i.e. measurement function related) parameters in the *Client* class, but we do not explain it in detail here.

The *Client* class has 'supervisor' functionality for the evaluation system(s) running at one machine. We denote the single delivery of the evaluation service as a *test run* (or in short: *test*), and the set of consecutive evaluation service deliveries executed by one instance of the evaluation system a *test cycle*. The end-user can instruct the evaluation system to execute a number of test runs (TEST_*OPTION_NUMBER* attribute in the *Client* class). This option is necessary when multiple instances of the evaluation system are interacting. We explain this case in the latter.

An object of the *ServerControlFunction* class comprises the functionality of one evaluation system instance, i.e., workload generator, measurement function and service logic. Every object of this class has a *serverName* attribute, uniquely identifying the instance of the evaluation system.

An object of the *ConfirmedServiceTest* and *UnconfirmedServiceTest* class execute the procedures of automated workload generation (i.e. confirmed, unconfirmed test case). The *Packet* class contains the functionality to create of workload generator unit, i.e., a *message* (denoted also as a *packet*). Each generated message has specific workload characteristics (size, rate, provided by the *MatrixPacketSizeRate* class). The *Communication* class facilitates the socket communication, i.e., sending and receiving of message from the SoD (see Figure 5.17 for implementation details). The sending of a message is followed directly by the *flush* function, which ensures that the message is offered immediately from the application layer (i.e. application buffer) to the transport service. As soon as the message is received by the transport service, it is going to be transported immediately by the SUT[49]. Both send and receive socket communication functions can generate a communication error.

[49] TCP Socket option *dataSocket.setTcpNoDelay* is set *true*, assuming that the TCP header flag PSH ([RFC 793]) is set for segments smaller than a MSS

```
  /**
   * Defines methods for communication over the TCP bearer
   * between two networked hosts (that is client & server).
   *
   */

public class Communication {
  // Attribute(s)

  public final static int COMMUNICATION_ERROR = -273;
  public final static int TIMING_ERROR        = -272;

  // Constructor(s)
  /**
   * Communication constructor.
   */
  public Communication() {
  }

  // Method(s)
  /**
   * Implements the "request" and "response" SP of connection-oriented service,
   * For the given Packet object the timestamp in ms on completion of the send action
   * or COMMUNICATION_ERROR is set if the send action failed
   * (that is the communication problem occurred).
   *
   * @param   outChannel    channel to write to
   * @param   packet        Packet object, which data are send
   */

  public void send_tcp( PrintWriter outChannel, Packet packet ) {

    long timeStamp = 0;

    try {
      // timestamp the event of sending a PDU by a SP to the LLS
      timeStamp  = System.currentTimeMillis();

      outChannel.println( packet.getPDU() );

      outChannel.flush();
    }
    catch ( Exception ex ) {
      System.out.println( "Error [send_tcp]: Can't write to output channel" +
                          ex.toString() );
      packet.setErrorCode( COMMUNICATION_ERROR );
    }
    packet.setTimeStamp( timeStamp );
    }
    catch ( Exception ex ) {
      System.out.println( "Error [send_tcp]: Can't write to output channel" +
                           ex.toString() );
      packet.setErrorCode( COMMUNICATION_ERROR );
    }
    packet.setTimeStamp( timeStamp );
  }

 /**
   * Implements the "indication" and "confirmation" SP of
   * the connection-oriented service.
   *
   * @param   inChannel             channel to read from
   * @param   testRunStopTime       stop time for the test run
   *
   * @return  a <code>Packet</code> object with updated parameters:
   *          timestamp in ms on completion of the recv action or
   *          COMMUNICATION_ERROR if the recv action failed
   *          (that is the communication problem occurred).
   */
  public Packet recv_tcp( BufferedReader inChannel ) {

    String pdu = new String( "" );
    int errorCode  = 0;
    long timeStamp = 0;

    try {
      pdu = inChannel.readLine();
      timeStamp  = System.currentTimeMillis();

      if ( pdu.length() == 0 ) {
        System.out.println( "Error [recv_tcp]: pdu is empty" );
        errorCode = COMMUNICATION_ERROR;
        return new Packet( Packet.constructPCI( 0, 0, 0 ), Packet.constructDataSDU( 0 ),
                           errorCode, timeStamp );
      }
      else {
        return new Packet( pdu.substring( 0,
```

```
                        ( Packet.PCI_COMPONENTS * Packet.PCI_FRAGMENT_SIZE ) ),
                          pdu.substring( Packet.PCI_COMPONENTS * Packet.PCI_FRAGMENT_SIZE ),
                          errorCode, timeStamp );
      }
    }
    catch ( Exception ex) {
     System.out.println( "Error [recv_tcp]: general error " + ex.toString() );
     pdu = "0";
     errorCode = COMMUNICATION_ERROR;
     return new Packet( Packet.constructPCI( 0, 0, 0 ), Packet.constructDataSDU( 0 ),
                        errorCode, timeStamp );
    }
  }
}
```

Figure 5.17 Implementation of the reliable Java socket communication

The *ConfirmedServiceTest* object executes the confirmed SEs; a send SP is always followed by a receive SP. The *UnconfirmedServiceTest* object executes a series of unconfirmed SE; send (or receive) SP followed by each other at particular time intervals. Figures 5.18 and 5.19 provide the implementation details of one SE/SP for both test cases (bold font indicates the most important part of the code).

```
public boolean confirmedServiceTest( String serverName, long testStartTime,
                                     int currentTestRun, int row, int column ) {

< deleted lines>

// start test run...
          // construct packet to be send according to the workload parameters
          sendPacketSize = Server.matrixPacketSizeRate.getPacketSize( r );
          packet.constructPacket(
            Packet.constructPCI( this.packetSeqNo, sendPacketSize, maxPacketRate ),
            Packet.constructDataSDU( sendPacketSize ), 0, 0 );

          // send packet...
          communication.send_tcp( Server.dataOutToClient, packet );
          // retrieve timestamp of the event
          requestTime = packet.getTimeStamp();

< deleted lines>

//request packet successfully send, receive response packet...
            packet = communication.recv_tcp( Server.dataInFromClient );
            // retrieve timestamp of the event
            confirmTime = packet.getTimeStamp();

< deleted lines>
```

Figure 5.18 Confirmed service test case; SE execution

```
  private boolean upLinkTest( String serverName, long testStartTime, int row ) {

< deleted lines>

// construct packet
                  packet.constructPacket(
                    Packet.constructPCI( this.packetSeqNo, sendPacketSize,
                                         currentSendPacketRate ),
                    Packet.constructDataSDU( sendPacketSize ), 0, 0 );

                  // send packet
                  communication.send_tcp( Server.dataOutToClient, packet );
                  // retrieve timestamp of event
                  requestTime = packet.getTimeStamp();

< deleted lines>
```

Figure 5.19 Unconfirmed service test case; SP send execution

The *ConfirmedServiceTest* and *UnconfirmedServiceTest* classes have the functionality to count and select executed SE parameters. This functionality ensures that that the whole set of workload parameters is executed. Moreover, both functions have counter functionality, counting the repetitions of the workload with the same parameters. Figures 5.20 and 5.21 provide the essentials of the procedures. Both classes implement partially the measurement functionality; as indicated in bold font in the Figures, after each execution of SE, the system collects the measurement data for an observation (i.e. interim test result) and then saves complete measurement data for the sample.

```
   // for N packet sizes execute.....
   numberOfPacketSizes = Server.matrixPacketSizeRate.getMatrixLength();
   for ( int r = requestStartPacketSize; r < numberOfPacketSizes; r++ ) {
    // for each packet size S execute...
      for ( int c = confirmStartPacketSize; c < numberOfPacketSizes; c++ ) {
       // repeat procedure x times.....
        for ( int repetition = 0; repetition < Client.SAMPLE_SIZE ; repetition++ ) {

< deleted lines>

      sendPacketSize = ServerControl.matrixPacketSizeRate.getPacketSize( c );
      communication.send_tcp( dataOutToServer, packet );

< deleted lines>

      packet = communication.recv_tcp( dataInFromServer );

< deleted lines>

              // save observation (for SE)
             saveInterimTestResult( this.packetSeqNo, maxPacketRate,
                                    sendPacketSize, recvPacketSize,
                                    requestTime, confirmTime );

< deleted lines>

        } // for loop repetition < Server.SAMPLE_SIZE
        // save sample
        saveMeasurementData( serverName, currentTestRun, timeStampTestRun,
                             System.currentTimeMillis(), sendPacketSize,
                             recvPacketSize, vMatrixOfSendRecvTimes );
      } // for loop s < numberOfPacketSizes
    } // for loop n < numberOfPacketSizes
```

Figure 5.20 Workload generation and measurement function for confirmed service test

```
// for N packet sizes execute test .....
for ( int r = requestStartPacketSize; r < ServerControl.matrixPacketSizeRate.getMatrixLength(); r++ ) {
    sendPacketSize = ServerControl.matrixPacketSizeRate.getPacketSize( r );
    sendMaxPacketRate = ServerControl.matrixPacketSizeRate.getMaxPacketRate( r );
    while ( repetition <= Client.SAMPLE_SIZE ) {
      for ( int currentSendPacketRate = startSendPacketRate;
            currentSendPacketRate <= sendMaxPacketRate;
            currentSendPacketRate++ ) {
          /**
           * for each second in the Server.TRANSMISSION_WINDOW, send packets
           * at the specified rate currentSendPacketRate (packets/second)
           */

          for ( int second = 0; second < transmissionWindow; second++ ) {
            /**
              * generate currentSendPacketRate packets to transmit,
              * calculate the inter packet waiting time timeToWait and wait
              * accordingly between transmission of consecutives packets
              */
            for ( int packetNumber = 1;
                  packetNumber <= currentSendPacketRate; packetNumber++ ) {
                packet.constructPacket(
                  Packet.constructPCI( this.packetSeqNo, sendPacketSize,
                                       currentSendPacketRate ),
                  Packet.constructDataSDU( sendPacketSize ), 0, 0 );
                // send packet
                communication.send_tcp( dataOutToServer, packet );
                requestTime = packet.getTimeStamp();
                // No communication error occurred
                saveInterimTestResult( this.packetSeqNo, currentSendPacketRate,
                                       sendPacketSize, 0, // recvPacketSize = 0
                                       requestTime, 0 ); // indicationTime = 0
                /*
                 * calculate the inter packet waiting time [ms],
                 * requestTime - timeStamp = transmission time of an individual
                 * packet
                 */
                if ( packetNumber < currentSendPacketRate ) {
                  interPacketTime = ( 1000 / currentSendPacketRate ) -
                                    ( requestTime - timeStamp );
                }
                else { // last packet
                  /* padding ms to roundup to 1000 ms and add BACKOFF_TIME to
                   * avoid mutual influence of consecutive test runs
                   */
                  interPacketTime = 1000 - ( packetNumber - 1 ) *
                                   ( 1000 / currentSendPacketRate ) -
                                   ( requestTime - timeStamp );
                }
                waitTime( interPacketTime );
            } // for each packet of the same rate [1..actualRate]
          } // for each second in transmissionWindow
      } // for ( int currentSendPacketRate = startSendPacketRate;
      repetition++;
    } // while ( repetition <= Server.SAMPLE_SIZE )

  } // for ( int r = requestStartPacketSize;
  saveMeasurementData( serverName, currentTestRun, testStartTime,
                       System.currentTimeMillis(),
                       0, 0, vMatrixOfSendRecvTimes );
```

Figure 5.21 Workload generation and measurement function for unconfirmed service test

The *Client* object times the evaluation service delivery in two ways: 1) it times the workload generation function (i.e. there is a limited time to examine the SoD) and 2) it times each SE execution; in a confirmed service case after each send action there is a maximum time for a receive notification to arrive. For the unconfirmed service case, the system calculates the time in between two consecutive send actions (based on the current packet rate). Figure 5.22 provides the implementation details of the timing functionality. If the service time elapses, it is interrupted at any point of its execution and the measurement data save is triggered. If the SE timeout occur, the workload generation function generates a timing error.

```
// for N packet sizes execute.....
// for each packet size S execute...
// repeat procedure x times

      currentTime = System.currentTimeMillis();

      if ( Client.MAX_TEST_RUN_TIME > currentTime -  testStartTime ) {
// Confirmed test
          // send packet
          dataSocket.setSoTimeout( Client.TIMEOUT_DAT_CHNL );
          // wait to receive packet maximum TIMEOUT_DAT_CHNL [ms]
// Unconfirmed Test
          // send packet
          waitTime( interPacketTime );

      }
       else {// neither receiving nor sending possible, service time elapsed!
         saveMeasurementData( serverName, currentTestRun, timeStampTestRun,
                              System.currentTimeMillis(), sendPacketSize,
                              recvPacketSize, vMatrixOfSendRecvTimes );
            return false;
           }
```

Figure 5.22 Implementation of timing peculiarities in evaluation service delivery

The *measurement function* is implemented in the *Communication*, *Packet* and *SaverOfPerformanceData* classes and partially in the confirmed and unconfirmed test methods. The *Communication* and *Packet* classes hold functionality to timestamp the single events (i.e. sending and receiving the packets at a socket) and adding this timestamp to the packet. The functions take timestamps in milliseconds; the value of the timestamp is the difference between the current system time and midnight, January 1, 1970 UTC (*System.currentTimeMillis* method). The functions collect observations within sample while executing the confirmed/unconfirmed tests. The *SaverOfPerformanceData* object takes care of the actual storage of the sample. Recall that Figure 5.20 and Figure 5.21 provided details of measurement data storage.

The *SaverOfPerformanceData* object comprises the functionality to store measurement samples and their context data in the dedicated text file. For retrieval purposes, each stored file has a meaningful name, the name includes partially the system and workload parameters. Figure 5.23 provides an example file name and file content.

Hence, the measurement function fulfils the requirement of automated monitoring of events and correlation of measurements to workload parameters.

There is clearly no cooperation between the workload generator and the measurement system of the evaluation system. We implemented both functionalities in a co-existing manner in the *Packet*, *Communication, ConfirmedServiceTest* and *UnconfirmedServiceTest* classes. The workload generator function uses a socket interface to the *SoD*, while the measurement function is always in passive mode, it just timestamps events observed on the SoD socket interface.

The implemented evaluation system has extended functionality to handle errors. The system always notifies the end-user on a successful or erroneous service execution. In case of an error, the system triggers the measurement data save function. Figure 5.24 provides an example of the error handling functionality.

```
Confirmed test example
File name: 01_0100_174.174_1079013261736.txt

File content:
*********************************************************************
* owners           : Kate Wac & Richard Bults, University of Twente
* program version  : 2.3.2
* date             : Thu Mar 11 14:53:51 GMT+01:00 2004
* test run no.     : 1
*
* hostname         : 01
* service type     : TCP_USER_CONFIRMED
* sample size      : 500
* trans. window    : 40 (unconfirmed service only)
* buffer sizes [B] : socket        65536
*                    application   65536
* timing [ms]      : start         1079013261736
*                    stop          1079013534398
*                    backoff       30000
* link specs.      : link capacity 8192
*                    sat. factor   1.0
*********************************************************************
* Column legend:
* Sno = Packet sequence number, Pr = Packet rate,
* SPs = send packet size, RPs = received packet size,
* Rt = send packet request time, Ct = received packet confirm time
*
* Sno   Pr      SPs     RPs     Rt      Ct
*
1       1       174     174     1079013261736   1079013262447
2       1       174     174     1079013262447   1079013263268
* * *
500     1       174     174     1079013534178   1079013534398
```

Figure 5.23 Example of the measurement data file

```
// Client class
      error = executeTestCycle( serverName, getRowMatrixCoordinates(),
                                getColumnMatrixCoordinates() );
----//------
// ConfimedServiceTest class
< deleted lines>

       System.out.println( "Info [confirmedServiceTest]: test is running..." );
          if ( packet.getErrorCode() == Communication.COMMUNICATION_ERROR ) {
            System.out.println( "Error [confirmedServiceTest]: send communication error occured " );
            saveInterimTestResult( this.packetSeqNo, maxPacketRate,
                                   sendPacketSize, 0, // recvPacketSize = 0
                                   packet.getErrorCode(), // error = requestTime
                                   0 ); // confirmTime = 0
            saveMeasurementData( serverName, currentTestRun, timeStampTestRun,
                                 System.currentTimeMillis(), sendPacketSize,
                                 recvPacketSize, vMatrixOfSendRecvTimes );
            return true; // error, test run interrupted
          }

< deleted lines>
        saveMeasurementData( serverName, currentTestRun, timeStampTestRun,
                                 System.currentTimeMillis(), sendPacketSize,
                                 recvPacketSize, vMatrixOfSendRecvTimes );
        System.out.println( "Info [confirmedServiceTest]: SE " + sendPacketSize +
                           ","+ recvPacketSize + " test run successfully finished in " +
                           ( System.currentTimeMillis() - timeStampTestRun ) / 1000 + " s" );
return false; // no error
---//---

// Client class
    testCycleExitStatus = error ? "forced" : "normal";
    System.out.println( "Info [run]: " + testCycleExitStatus +
                        " exit test cycle" );
```

Figure 5.24 End-user notification on the service delivery status

### 5.3.2 Evaluation System Distribution

Section 5.3.1 presents the implementation details of the evaluation service. This section provides the implementation details concerning the distribution of the evaluation service.

Due to the distribution of the transport system and the nature of delivered transport service, the evaluation system is also distributed. There is always a pair of evaluation system instances that interact to deliver the evaluation service. Figure 5.25 presents the service view of the system containing two evaluation system instances that interact over the SoD.

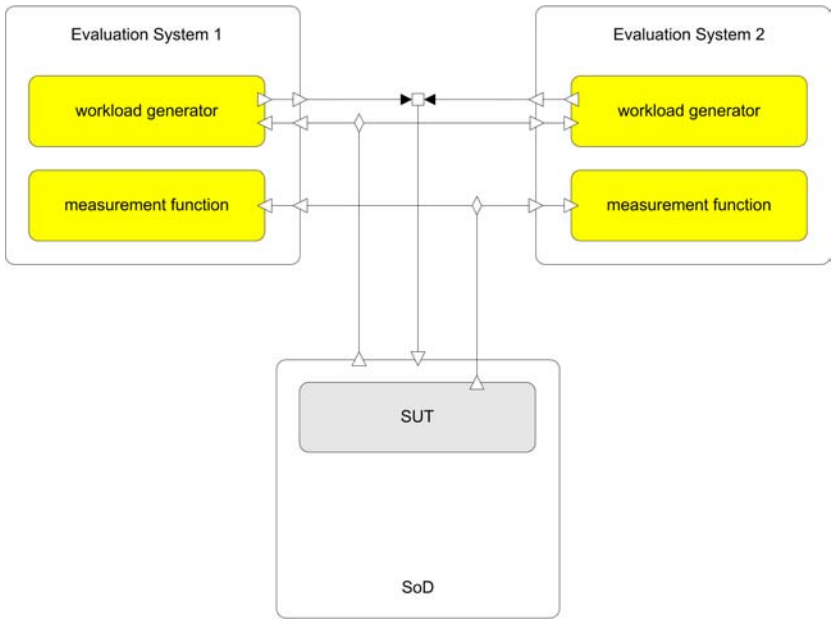



Figure 5.25 Functional view of SoD and two evaluation systems

Particularly the workload generator function executed by the interacting evaluation systems varies in instances of interacting evaluation systems. The measurements system functionality in both systems is identical.

Recall the scenario of the systems interaction provided in section 4.4. The workload generator of one the evaluation system instances always takes an initiative to stimulate the SoD. The second evaluation system instance responds for this stimulus, and as a return, it generates a new stimulus to the SoD. Due to this specific roles of evaluation system instances, we called them *client* and *server*. A server is an instance of the workload generator, which always initializes the evaluation service. A client is the system instance, which can respond to the behaviour of the SoD. Moreover, the server is the workload generator responsible for SE's uplink generation, while the client is the workload generator responsible for of SE's downlink generation.

We assume that the end-user provides the workload and the SoD parameters once to the system (regardless of the number of evaluation system instances that are examining a SoD concurrently). Therefore, for every instance of the evaluation system, these parameters need to be provided, such that the actions of all individual instances are synchronised. The important issue is pointed:

> *How to synchronise the actions of the evaluation system instances to create the coordinated actions?*

We dealt with the coordination of the client and server actions by introducing of a evaluation systems *initialisation phase*. In this phase, the client provides set of workload and the SoD parameters (i.e. management data) to its server peer. The successful execution of the initialisation phase is required before the evaluation service execution can start (i.e. before the instances start data generation). The client and server exchange management data over a dedicated (reliable) data channel. Particularly, the client sends the workload and system parameters to the peer server indicated in Figure 5.26.

Exchanging management data is as follows: the *service type* determines the way a server and client generate workload to the SoD (i.e. unconfirmed service test or confirmed service test). Regarding the workload parameters, the client feeds the server with the information on where to start the data generation in the workload matrix (coordinates) and how many times each SE needs to be repeated (i.e. what is the sample size). Moreover, the client sends the link capacity's system parameter and link saturation value (workload parameter) to server.

For administration purposes, the client provides a server also with the test date. The client feeds the server with the maximum test run time and with the value of the socket read timeout.

As we said in the previous section, we have automated the evaluation service delivery and implemented it in such a way that it is possible to execute a number of successive independent evaluation service deliveries (*test runs*). Therefore, it is possible to execute the *test cycle*. This option is utilised when multiple instances of paired client-server systems examine the SoD simultaneously. Figure 5.27 visualizes the TEST_OPTION_NUMBER value with relation to the number of client-server systems

```
/**
 *  Date of testcycle execution
 */
public static String TEST_DATE = date.toString();

/**
 *  Executed service type
 */
public final static String SERVICE_TYPE = CONFIRMED_SERVICE_TYPE;

/**
 *  Maximum time interval [ms] for evaluation service provision
 */
public final static int MAX_TEST_RUN_TIME = 9 * 60 * 60 * 1000;

/**
 *  Matrix of test run start times [ms]
 */
 matrixTestRunStartTimes = Client.testScheduleObject.getMatrixTestRunStartTimes(
           registrationNumberOfServer, Client.MAX_NUMBER_OF_SERVERS, Client.TEST_OPTION_NUMBER);

/**
 *  Maximum time [ms] interval to execute one SE.
 */
public static int TIMEOUT_DAT_CHNL = 60 * 1000;

/**
 *  Number of repetitions of SE execution (i.e. sample size).
 */
public static long SAMPLE_SIZE = 500;

/**
 * uplink / downlink [bps] > from Client to Server saturation factor.
 */
public static final double UP_LINK_SATURATION_FACTOR = LINK_SATURATION_FACTOR;

/**
 * From the Server perspective: maximum transmission capacity of slowest link
 * in communication system.
 */
public final static int UP_LINK_CAP = UMTS_UPLINK_CAPACITY;

/**
 * backoff time [ms] between two consecutive test runs (empirical value).
 */
public static final int BACKOFF_TIME = 30 * 1000;

/**
 * starting point of test run: i.e. starting coordinates in the
 * MatrixPacketSizeRate (0.0).....(x.y)
 */
public static String MATRIX_COORDINATES = "0.0";
```

Figure 5.26 Management data provided by client to server

```
/**
   * TEST_OPTION_NUMBER controls the number of test runs and the test run per
   * test cycle. This attribute can have a value 1..MAX_NUMBER_OF_SERVERS
   * (i.e. maximum number of servers that can be registered to this client).
   *
   * Example use: MAX_NUMBER_OF_SERVERS = 5 and TEST_OPTION_NUMBER = 3.
   *              The test cycle will be constructed based on the figure below:
   *
   *       S1          S1,2          S1,2,3        S1,2,3,4      S1,2,3,4,5
   * |-------------|-------------|-------------|-------------|-------------|
   *   option 1      option 2      option 3      option 4      option 5
   *
   * The test cycle starts with a run that uses three servers in parallel and
   * end with a run that uses five servers in parallel.
   *
   */
  public final static int MAX_NUMBER_OF_SERVERS = 1;
  public final static int TEST_OPTION_NUMBER = 1;  // [1...MAX_NUMBER_OF_SERVERS]
```

Figure 5.27 Test cycle management implementation

We assume that in a test cycle, all instances of the client-server peers examine the SoD simultaneously and by means of the same procedures (i.e. using the same system and workload parameters). The evaluation system's end-user inputs the workload and system parameters only once to the system and there is only one client system instance (*Client* object), which has all the information on the system and workload parameters. This instance distributes this information to all other instances during the initialisation phase. Therefore, each new test cycle starts with the initialisation phase during which the parameters of the evaluation service are set. This scenario is implemented by means of the *Java threads*.

Due to the organisation of the evaluation service delivery in test cycles, each server gets the *matrixTestRunStartTimes* parameter from its peer client evaluation system. This parameter contains evaluation service start-times. Therefore, to successfully deliver the evaluation service, both (i.e. peer client-server) system clocks need to be synchronised.

## 5.4 Measurements Setup

In section 5.1, we identified the SoD instances and combinations of the workload and system parameters to be measured. In section 5.2 and 5.3, we refined the evaluation system. However, we did not provide the experiments details. Hence, the following section (5.4.1) provides the experiments and their execution in detail ("where, when and who").

### 5.4.1 Experiments

We defined 11 experiments based on eight SoD instances. In each experiment, we deployed the evaluation system instance (Java program) at each computer system involved. We deployed the server evaluation system's instance at the ingress point of the SoD, such that it generates the uplink traffic for the SoD. We deployed the client instance at the SoD egress point, such that it always generates the SoD downlink traffic. In case of the SoD_6, we deployed an additional instance of evaluation system at pc2 computer system, i.e. in between the SUT and the UTnet. We call this instance of the evaluation system the *proxy*. The server sees the functionality of the proxy as it would be client. Analogously, the client sees the proxy as the server.

Each experiment starts with the initialisation phase, during which we synchronise clocks of the involved computer systems. Moreover, before the service starts, its end-user provides the workload and system parameters to the evaluation system. The evaluation system instances exchange this information in the management phase. We defined the management channel between the peered client-server evaluation system instances as the TCP connection on a port number[50] 4444. We defined the data channel, over which they exchange the workload data as the TCP connection on a port number 6790.

[50] In a TCP/IP-based network such as the Internet, it is a number assigned to an application running in the computer. The number is included in the transmitted packets to link the incoming data to the correct service. [TechWeb]

During the measurements, links between computer systems are kept as silent as possible. This means that no users/processes other than the ones strictly needed for the measurement were running on the systems and did not permit any other traffic on the links.

We have executed all tests in the timeframe of 1st February 2004 – 31st March 2004. Because the SUT has been free from other users (pre-commercial network), the tests were executed without taking into account of the SUT load distribution, and we treated every day and night the same. Entities (equipment) and parties that have been involved in test activities as indicated in Figure 5.3. Server(s) and Client(s) were always placed at the University of Twente, Zilvering building, room 5013. For the scalability test, we placed all the involved servers in this room, which means in the area of around $10m^2$. The coverage of the Vodafone's UMTS network was always full. We did not test mobility or system / intersystem (GPRS/UMTS) handover mechanisms. All of the equipment have been fully charged and constantly powered during the experiment execution.

| | | | |
|---|---|---|---|
| **TestID** | Experiment number:<br>e.g. *01* | | |
| **APN** | APN system parameter used:<br>*utwente.nl* or *web.vodafone.nl* | | |
| **HOPS** | Evaluation systems ported:<br>*Server <> Client* for the case when comp. systems used are: nb/iPAQ and pc1<br>or<br>*Server <> Proxy <> Client* for the case when comp. systems used are: nb, pc2, pc1 | | |
| **Test description** | Syntax:<br>ServiceType, Matrix, ExComm.–APN/IntComm., EvalSys1/EvalSys2 [/EvalSys3], NoOfEvalSys<br>e.g. *Confirmed Service, 20x20, UMTS-UT/USB, Server/Client, 1 Terminal* | | |
| **Extra comms** | For the computer system at the ingress point, a UMTS wireless communication is used to access the SUT | | |
| **Intra comms** | For the computer system at the ingress point of the SUT, a wireless communication is used to access the UMTS terminal e.g. *USB*, *Bluetooth* or *PCMCIA* | | |
| **Clock sync** | Software used to synchronise server system clock<br>e.g. *Tardis 2000 V1.5* | Value of clock drift (for server)<br>e.g. *1s / test run* | NTP server used<br>e.g. *ASUS nb* |
| **Equipment -** computer systems on which we ported the instances of evaluation system, the UMTS terminal used | | | |
| **Identifier** | name | HW-platform | SW-platform |
| **ClientId 01** | pc1<br>e.g. *Utip194* | e.g.<br>*P4, 2.4 GHz, 512 MBmem* | Operating system<br>e.g. *WindowsXP, JDK 1.3.0* |
| **ServerId 01** | nb, iPAQ<br>e.g.<br>*Freelander* | mP3, 1 GHz, 640MBmem | Operating system<br>e.g. *WindowsXP, JDK 1.3.0* |
| **Terminal** | e.g.<br>*Nokia 1* | e.g.<br>*Nokia 6650* | e.g. *PR4* |

Figure 5.28 Example of an experiment specification table

The following sections provide the experiments details. Experiments naming convention (i.e. TestID) is "T" plus the (unique) test number (test number is not necessary the same as an experiment number). Figure 5.28 provides an experiment specification table containing the workload and SoD parameters. Example values of particular parameters are provided in italic.

| **no. of observations** **500** | | **columns:** **downlink packet sizes** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 174 | 349 | 524 | … | 35632 | 41920 | 48208 |
| **rows:** **uplink packet sizes** | 174 | 0,0 | 0,1 | 0,2 | … | 0,17 | 0,18 | 0,19 |
| | 349 | 1,0 | 1 | 2 | … | 17 | 18 | 19 |
| | 524 | 2,0 | 1 | 2 | … | 17 | 18 | 19 |
| | … | … | … | … | … | … | … | … |
| | 7336 | 17,0 | 1 | 2 | … | 17 | 18 | 19 |
| | 7860 | 18,0 | 1 | 2 | … | 17 | 18 | 19 |
| | 8122 | 19,0 | 1 | 2 | … | 17 | 18 | 19 |
| **500** | | 174 | 349 | 524 | … | 35632 | 41920 | 48208 |

| | |
|---|---|
| NOP | not planned, not executed |
| 1 | planned, not executed |
| 2 | out-of-band time synchronisation |
| 3 | in-band time synchronisation |
| 4 | execution failed |

Figure 5.29 test run matrix legend

Figure 5.30 provides the specification of Experiment 1 (a.k.a. the "benchmark"). Appendix H provides the specification of Experiments 2 to 11. Below each experiment's specification there is a table indicating which workload parameters were executed successfully. Figure 5.31 shows the successfully executed SEs for Experiment 1. In most cases, due to the time constraint, we did not succeed to execute all planned SEs. Figure 5.29 provides the legend of SEs execution.

**Experiment 1**

| | | | | |
|---|---|---|---|---|
| TestID | 01 | | | |
| APN | utwente.nl | | | |
| HOPS | Server <> Client | | | |
| Test description | Confirmed Service, 20x20, UMTS-UT/USB, Server/Client, 1 Terminal | | | |
| Extra comms | UMTS | | | |
| Intra comms | USB | | | |
| Clock sync | Tardis 2000 V1.5 | | 0.712 seconds/day | ASUS notebook |
| Equipment: | | | | |
| Identifier | Name | HW-platform | SW-platform | |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 | |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0 | |
| Terminal | Nokia 1 | Nokia 6650 | PR4 | |

Figure 5.30 Specification of Experiment 1

## 5.5 Measurements Design Evaluation

This section provides the preliminary validation of the (raw data) measurements (section 5.5.1). The data evaluation phase derives raw data statistics and visualises this data in conjunction with statistical results. This section emphasizes also the problems encountered during the measurements execution and their possible solutions (sections 5.5.2 to 5.5.5).

### 5.5.1 Raw Data Evaluation

Raw data set consists of the outcomes of the executed performance measurements. Outcomes are random variables, and they can be different each time the measurements are executed. Due to this, the statistical analysis (e.g. variability) of gathered data is very important.

| 500 | 174 | 349 | 524 | 1048 | 2096 | 4192 | 6288 | 7336 | 8384 | 9432 | 10480 | 12576 | 14672 | 15720 | 16768 | 23056 | 29344 | 35632 | 41920 | 48208 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 174 | 0,0 | 0,1 | 0,2 | 0,3 | 0,4 | 0,5 | 0,6 | 0,7 | 0,8 | 0,9 | 0,10 | 0,11 | 0,12 | 0,13 | 0,14 | 0,15 | 0,16 | 0,17 | 0,18 | 0,19 |
| 349 | 1,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 524 | 2,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 786 | 3,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 1048 | 4,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 1310 | 5,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 1572 | 6,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 2096 | 7,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 2620 | 8,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 3144 | 9,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 3668 | 10,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 4192 | 11,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 4716 | 12,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 5240 | 13,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 5764 | 14,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 6288 | 15,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 6812 | 16,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 7336 | 17,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 7860 | 18,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 8122 | 19,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 500 | 174 | 349 | 524 | 1048 | 2096 | 4192 | 6288 | 7336 | 8384 | 9432 | 10480 | 12576 | 14672 | 15720 | 16768 | 23056 | 29344 | 35632 | 41920 | 48208 |

Figure 5.31 Experiment 1 test run matrix

The raw data evaluation process is structured as follows. We collected and (roughly) validated the (raw) measurement results. We generated of the statistical data based on the data obtained from each individual experiment.

> *Note: Specialized analysis tools and programs exist for the purpose of data analysis (e.g. Matlab), but due to the specific nature of the measurement data that we needed to analyse, we developed our own module for the automatic data processing as a Java application.*

The Java application delivers functionality for statistical analysis of the raw data and the raw data visualization. An overview of the application functionality:

1) it retrieves the measurement data files from the client / server / proxy part of the evaluation system

2) it matches and combines the corresponding client / server / proxy measurement data files

3) it calculates (based on retrieved and matched data files):
   the uplink, downlink and round-trip delay(s) performance measure,
   the mean and standard deviation of the measured delay(s), and
   the 95% confidence interval of the estimated delay(s).

4) it visualises the raw measurement data

5) it stores (for later retrieval) the visualized raw data and the corresponding statistical results

Volume 2 (only electronically available) of this report presents the visualized raw data (plus its statistical data) collected for all executed measurements.

Based on the visualized raw data and its corresponding statistics the measurements are evaluated. Based on this preliminary evaluation we decide to keep the data or to discard it and re-execute the measurements. It occurred, that we encountered some problems during measurements and that we realized the worst-case scenario mentioned in section 5.2.1 (i.e. *"If anything can go wrong, it will…"*). We had to repeat the whole set of measurements. Due to the time constraint, only the majority, (not all) of the measurements were re-executed. Particularly problems with clock synchronisation / clock drift, and the unconfirmed test case procedure execution, have caused the measurements re-execution. In the next sections, we highlight our measurements experiences, problems and propose some solutions.

**Number of observations**

After the first round of measurements, we verified the empirical number of 500 observations in the measurement sample. The number of observations in sample is important, such that the sample measurement data is statistically sound. We took dedicated measurement in Experiment 1, for SE packet size: 524 Bytes, 524 Bytes for

500 and 2500 observations in sample, and wanted to observe the influence of the number of observations on the accuracy of the results. Theoretically, the accuracy of the results increases with the number of observations. However, Figure 5.32 shows that there are no improvements in statistical data accuracy for 2500 observations in comparison to 500. For a larger number of observations, problems with the system's in-band clock synchronisation and/or the network resource problems[51] are more likely to occur.

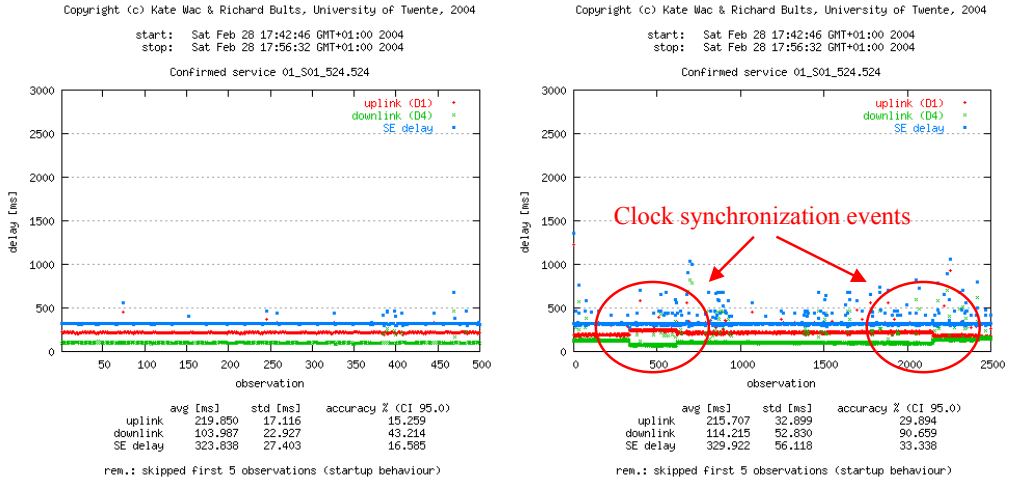



Figure 5.32 500 and 2500 observations and their statistical characteristics

We concluded that there is a trade-off between the number of observations and the accuracy of the obtained results. However, the majority of measurements with 500 observations resulted in accuracy up to 20% that may be far from the desired accuracy of 5%-10%[52], we accepted it. Therefore, we remained the number of 500 observations, as statistically sound for our measurement activity.

### 5.5.2 Clock Synchronisation

The synchronisation of evaluation system instances' actions in time is of vital importance for a successful delivery of the evaluation service. The real-time clock of the machines, at which an instance of evaluation system is running, has to be synchronised accurately with some external time source by means of Network Time Protocol (NTP) protocol or Simple Network Time Protocol (SNTP).

[51] The SUT was unloaded but there is a possibility of 'unforeseen' background traffic
[52] The delay accuracy of e.g. 5% for 95% confidence interval means that the value of delay is within the interval (mean delay – 5% mean delay, mean delay + 5% mean delay) with a probability 0.95

Methods for clock synchronisation depend on:

1) the location of the time source (i.e. NTP server) in relation to the SoD; time source can be placed inside or outside the SoD

2) the interface used to synchronise with the time source; the same interface is used to also deliver the evaluation service or a dedicated interface is used for the purpose of time synchronisation

In-band time synchronisation involves a time source inside the SoD and using the same interface for synchronisation and evaluation service. In contrary, the out-of-band time synchronisation involves a time source outside the SoD and using dedicated interfaces for the synchronisation and evaluation service. Evaluation service links are not providing constant delays, while links dedicated for synchronization purposes provide the constant delays.

These methods were distinguished only after we identified the clock synchronisation (based on the raw data) as a vital problem to the measurements (11th March 2004). Before that date, we based our measurements on the in-band clock synchronisation. The preliminary evaluation of these measurement data disclosed the problems of this method and then we designed and implemented the out-of-band time synchronisation method. Most of the measurements, which gave bogus results while executed with the in-band method, we repeated with the out-of-band synchronisation method. Therefore, later, when we attempted measurements and had a choice between synchronisation methods, we used the out-of-band time synchronisation (whenever the system configuration allowed it). An out-of-band time synchronisation method is preferred due to the separation of the synchronisation and evaluation service.

**In-band time synchronisation**

*Definition:*

> *In the public switched telephone network, (PSTN), in-band signalling is the exchange of signalling (call control) information on the same channel that the telephone call itself is using.*

In case of the evaluation system's in-band time synchronisation, a time source is placed within the SoD and the evaluation system synchronises its clock over the same link, as the evaluation service is executed. Figure 5.33 provides the details. The black straight arrows indicate the evaluation system data flow between the evaluation system and the SoD (i.e., stimuli), the blue curved arrows indicate the time synchronisation data flow (i.e. NTP/SNTP protocol data exchange) between the evaluation system and the time source.

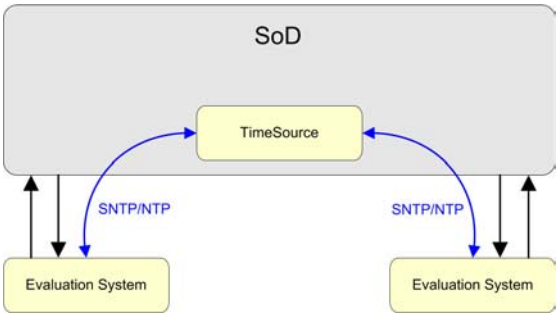



Figure 5.33 In-band time synchronisation

We use in-band time synchronisation, for example, when the system configuration did not allow for the out-of-band synchronisation (e.g. in case of the evaluation system attached to the SoD_6).

The software tool we used for time synchronisation was the Tardis Service application [TardisWeb]. This tool is running as a Windows XP's background process and automatically synchronises the system clock with (provided) NTP server(s) on a scheduled basis. This program promises synchronisation precision of 1ms. The NTP server used for synchronisation purposes is *ntp.utwente.nl* (IP address 130.89.1.19, alias *prodnet.civ.utwente.nl*) located within the UTnet. We choose to update clocks as frequently as possible (every 10 minutes) to keep them synchronised (i.e. avoid clock drift). Appendix I provides detailed information on the application and its configuration. For the iPAQ platform, we used the *ntpdate* command for synchronising of the clock with the external time source. Appendix J provides the *ntpdate* manual.

We did not encounter problems with clock synchronisation when executed between measurements activities (i.e. when the system was idle). However, problems were encountered when clock synchronisation was performed during measurements activities. This was due to the unpredictable behaviour of the wireless communication link over which Tardis had to synchronise the clocks. In particular if, a link is asymmetrical[53] (like UMTS) and heavily loaded. In addition, bearer change events (typical for UMTS networks) influence the clock synchronisation activity.

**Out-of-band time synchronisation**

*Definition:*

> *Out-of-band signalling is telecommunication signalling (exchange of information to control a telephone call) that is done on a channel that is dedicated for that purpose and separate from the channels used for the telephone call*

[53] The asymmetry of the communication link, over which the synchronization is executed, introduces the synchronization error. This is because the NTP protocol takes into account the (one-way) delay between the NTP server and NTP client when generating the reference timestamp for a NTP client.

The out-of-band time synchronisation method consists of a time source outside the SoD and uses dedicated interfaces for synchronisation and evaluation services (Figure 5.34). The black straight arrows indicate the evaluation service's data flow between the evaluation system and the SoD. The blue curved arrows indicate the time synchronisation's data flow (i.e. NTP/SNTP protocol data exchange) between the evaluation system and the time source.

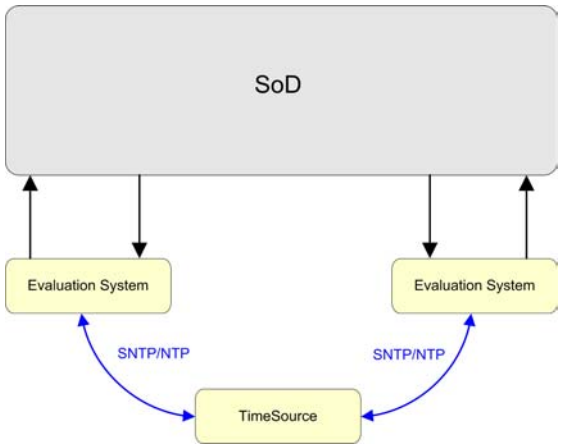



Figure 5.34 Out-of-band time synchronisation

We use the out-of-band time synchronisation method whenever a dedicated wired connection is available. The out-of-band connection was setup to a third dedicated computer system, which acted as the time source (i.e. NTP server).

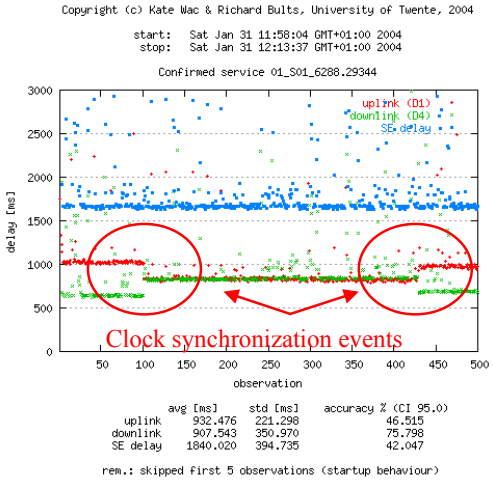
The software tool we used for time synchronisation is Tardis. Tardis client instances are running on two machines that communicate to the NTP server [WinWeb]. The system used as the NTP server is a machine with an IP address 192.168.0.48 placed within the dedicated LAN (network address/mask 192.168.0.0/24). We synchronise the clocks of the NTP clients automatically at time intervals of 10 minutes. Frequent synchronisation ensures the clock correctness and the clock drift avoidance during the tests execution. There were no problems encountered with the out-of-band synchronisation method.

**Method comparison**

The main reason of repeating the majority of measurements was that we encountered a serious problem with the computer systems' clock synchronisation. The out-of-band time synchronisation proofed to be more accurate than the in-band synchronisation. This is because the out-of-band synchronisation method uses a more predictable (i.e. wired, symmetrical) communication link than the in-band method (i.e. wireless, asymmetrical link). Figure 5.35 visualizes the raw measurements data for two particular SEs. The measurements on the left side (a) were obtained using the in-band synchronization method and the measurements on the right side (b) using the out-of-band synchronisation method. We observed an improvement in computer systems clock synchronisation

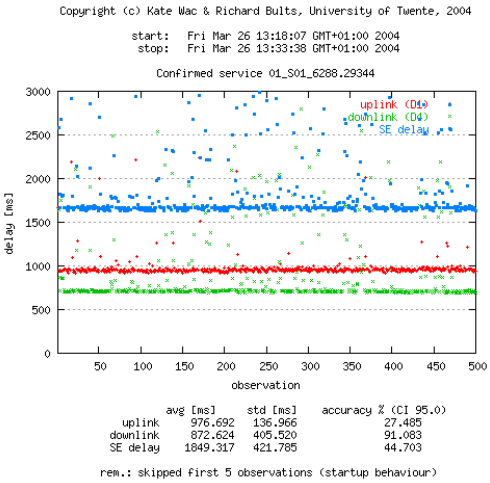
accuracy while using the out-of-band synchronisation method. In addition, we observed that the synchronisation error in the uplink delay is 'compensated' by the same synchronisation error in the downlink delay.

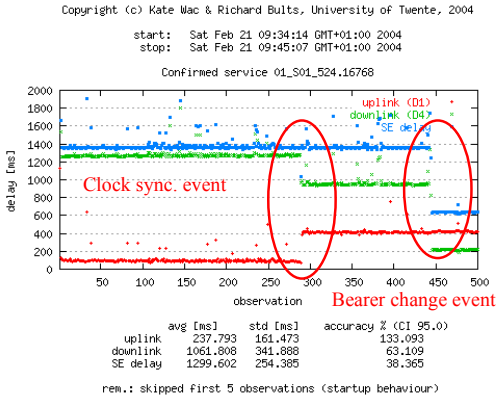
a) in-band clock synchronisation




clock synchronisation events

b) out-of-band clock synchronisation




clock synchronisation and changing bearer event

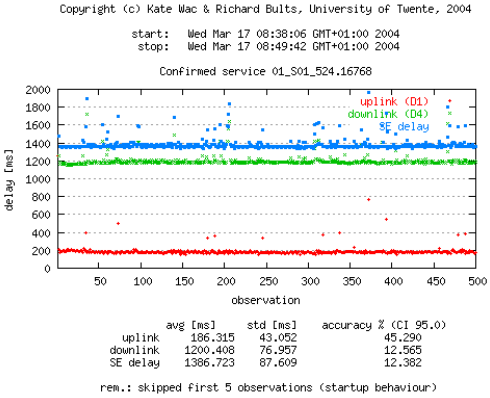



Figure 5.35 Measurements data for in-band (a) and out-of-band (b) clock synchronisation

Figure 5.35(a) shows that the clock synchronisation and bearer change events are distinguishable events. The clock change at one system (i.e. clock synchronization) results in the change of the measured uplink and downlink delays, such that the overall delay (SE delay) stays the same. This is different from the bearer change event, which affects the delay only in one direction. Hence, bearer changes affect the overall SE delay. Another important observation concerns the SUT bearer assignment mechanism. We

observed that at the beginning of the data transfer, the SUT assigns the common bearer. However, after number of observations (around 5), whenever the SUT assigns the dedicated bearer in one direction, the other direction is also assigned with the dedicated bearer. There cannot be the case that there is a common bearer assigned in one direction and dedicated in the other direction.

### 5.5.3 Clock Drift

We observed clock drift in both synchronisation methods. The clock drift has a negative influence on the statistics of the obtained measurements results (i.e. it decreases the statistical accuracy). The system clock drift occurs due to the influence of the external factors (e.g. temperature) on the clock behaviour. We discovered the clock drift in visualized measurement data, for example in Figure 5.36.

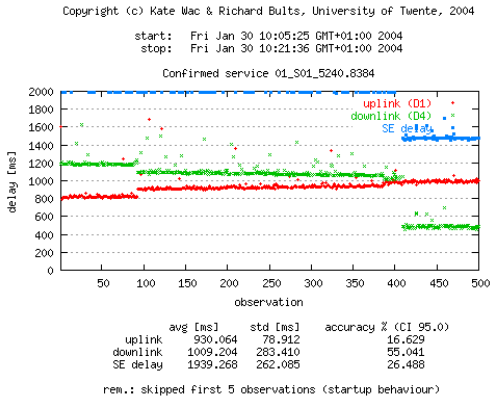



Figure 5.36 Observed clock drift

One of the possible explanations of the observed drift can be the high sensitivity of the clock to the fluctuations of temperature – inside the system and the external temperature. We observed it especially in case of the notebook computer system, which we used in the building where the heating system is operational from around 8.00-18.00. The notebook was placed on the desk next to the heater. We noticed the drift as the example Tardis log in Figure 5.37 shows.

```
2004/02/25 06:58:35.15 Debug:  Drift is -0.444514 (seconds/day)
2004/02/25 07:08:35.31 Debug:  Drift is 0.007622
2004/02/25 07:18:35.81 Debug:  Drift is 0.348974
2004/02/25 07:28:36.10 Debug:  Drift is 0.643457
2004/02/25 07:38:36.43 Debug:  Drift is 0.850201
2004/02/25 07:48:36.72 Debug:  Drift is 1.040077
2004/02/25 07:58:37.04 Debug:  Drift is 1.180771
2004/02/25 08:08:37.33 Debug:  Drift is 1.298303
2004/02/25 08:18:37.64 Debug:  Drift is 1.391597
2004/02/25 08:28:37.93 Debug:  Drift is 1.473020
2004/02/25 08:38:38.26 Debug:  Drift is 1.529977
2004/02/25 09:08:39.24 Debug:  Drift is 1.828834
```

Figure 5.37 Clock drift due the external temperature change

### 5.5.4 Clock Resolution

There are different system clock resolutions depending on the operating system. Figure 5.38 provides the clock resolution values for different operating systems.

The resolution of the clock under the Windows XP OS and Linux OS is different. Therefore, we consider the time differences in the range of 20 ms (in case when we used one machine with Windows XP OS in the measurement activity) to 40 ms (two machines) as the timing accuracy tolerances.

| Operating System | System clock resolution [ms] |
|---|---|
| Linux (2.2, x86) | 1 |
| Mac OS X | 1 |
| Windows 2000, XP | 10 |
| Windows 98 | 60 |
| Solaris (2.8, i386) | 1 |
| Solaris (2.7, sun4u) | 1 |

Figure 5.38 System clock resolution for different operating systems [YIPChi]

### 5.5.5 Terminal's Output Buffer Overflow

Output buffer overflow occurs when generation rate of packets send by the evaluation system to the UMTS terminal is too high comparing to the rate at which the UMTS network operates. During the evaluation of raw data, particularly data from execution of Experiments: 9, 10 and 11 (unconfirmed service test case), we encountered this problem. This problem, in most cases results in the service delivery failure. Buffer overflow occurs when we generated packets at the maximum rate (as calculated in Figure 5.39), while the

SUT initially assigns the common (slow) bearer for data transportation and could not handle the traffic offered.

To avoid the output buffer overflow described above, we introduced a slow start mechanism at the Java application level. During the slowstart, the SUT assigns the dedicated bearer and the traffic offered does not overflow the UMTS terminal's output buffer. Hence, we consider the slowstart phase as an initialisation phase for the actual performance measurements phase. We developed a formula for the number of packets generated during the slowstart in Figure 5.39.

Without the slowstart phase the network behaviour is unpredictable for a large number of packets send; high delays and delay variability is experienced, as the Figure 5.40 shows. In the figure, we show the experiment that we performed with the slowstart (blue line) and the experiment without the slowstart (red line). The difference in the system delay behaviour is significant in both experiments.

```
/*
 * protocol layer overhead (per packet) for PPP based communication
 */
  private static final int TCP_OVERHEAD = 32;
  private static final int IP_OVERHEAD = 20;
  private static final int PPP_OVERHEAD = 8;
```

$$\max PacketRate = \left\lfloor \frac{up\_link\_capacity * saturation\_factor}{packetSize + protocolOverhead} \right\rfloor$$

$$NoOfSlowstartPackets = \sum_{i=1}^{\max PacketRate} i = \frac{\max PacketRate * (\max PacketRate + 1)}{2}$$

Figure 5.39 Slowstart parameters

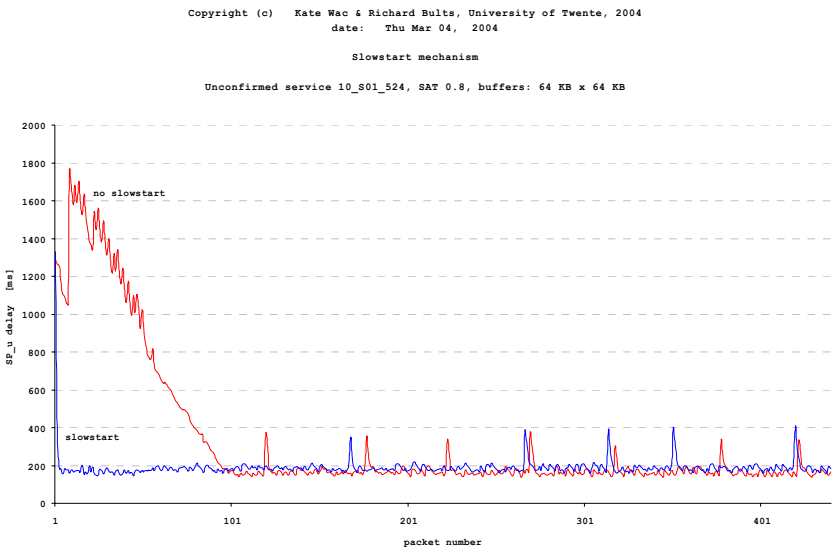



Figure 5.40 Slow start mechanism's influence on experienced delays

## 5.6 Conclusion

We presented the execution of the second phase of the performance evaluation methodology in this chapter. This comprises the identification of distinct measurement experiments, their design (i.e. measurement instrumentation) and execution.

Based on the system and workload parameters identified in chapter 4, we identified eight different real-world SoD instances to be evaluated. Based on these SoDs, we designed eleven distinct measurement experiments. To instrument them, we designed and successfully implemented the evaluation system. This activity was outside the scope of our performance evaluation methodology. During the measurement activity, an evaluation system generated a priori defined sequences of packets (i.e. workload) to the transport system under evaluation, and measured the performance characteristics this system.

After the execution of experiments, we processed the measurement data to evaluate it. The objectives of the data processing were: 1) to visualize the (raw) data obtained from each individual measurement activity in an extensive set of graphs, and 2) to generate the statistical data based on the (raw) data obtained from each individual measurement activity.

A first (quick) evaluation of the measurement data indicated significant problems with the time synchronisation of the different components of our evaluation system. We concluded that in-band time synchronisation of the components over UMTS wireless links with changing characteristics (e.g. roundtrip delays) is not usable. We repeated the measurement execution with an out-of-band synchronization method. Out-of-band time synchronisation method delivered better time synchronisation of the evaluation system components, but due to the instrumentation of the measurements, only a minimum tolerance of 40ms was possible.

The time synchronization problem was the main problem of our performance measurements design. After resolving it, we had collected a trustable measurements data. We approach the third (final) phase of the methodology, which is the evaluation of transport system performance.

# 6 Performance Modelling and Evaluation

This chapter focuses on the third part of our performance evaluation methodology as presented in section 3.4, and covers phases 8 and 9 (Figure 6.1).

| | |
|---|---|
| *1. State the Goals and System Definition* | |
| *2. List Services and their Outcomes* | |
| *3. Select Performance Criteria (i.e. Metrics)* | |
| *4. List System and Workload Parameters* | |
| *5. Select Factors and their Levels* | |
| *6. Select System and Workload Parameters* | |
| *7. Design and Execute the Experiments* | |
| ***8. Analyse, Evaluate and Interpret the Data*** | |
| | ***a. Select Model Representation*** |
| | ***b. Parameterise the Model*** |
| | ***c. Validate and Verify the Model*** |
| ***9. Present the Results*** | |

Figure 6.1 Performance evaluation methodology phase 3

Phase 8 of the methodology includes activities (8a-8c) to derive a high-level analytical model of the SUT based on the measurements data obtained in phase 7. This modelling activity answers the third research question: "What model can be used to represent the transport system and what are the conditions under which the model is valid?" Consequently, section 6.1 presents a simple queuing model for the transport system that we use as the SUT performance evaluation tool. We use the measurements data for parameterisation, validation and verification of the model.

It is important to notice that we used the SoD to perform the measurements, but in our analysis, we focus on the SUT performance evaluation. Therefore, section 6.2 provides conclusive graphs that visualize the combined measurement data and form the basis for the overall performance evaluation of the SUT. The consecutive sections describe the performance evaluation of the SUT based on the measurements data and the developed model. Section 6.3 presents the SUT's delay characteristics as experienced by a single service user, the bottleneck analysis follows in section 6.4. Section 6.5 presents the influence of changing of system parameters on the SUT performance. Section 6.6 describes the scalability of the system with respect to the number of concurrent users. Consequently, sections 6.3-6.6 cover the activities of phase 9 of our performance evaluation methodology.

The fourth research question: "Consider the MobiHealth BANip as a user confirmed application protocol: What is the optimal PDU size of a UMTS based transport system?", and the fifth question: "Consider MobiHealth BANip as a user unconfirmed application protocol: What is the optimal PDU size and PDU transmission rate of a UMTS based transport system?", are answered based on the overall performance evaluation of the SUT.

## 6.1 Transport System Model

The main objective of this section is to propose a simple high-level model that captures the performance characteristics of the MobiHealth transport system. Since it is not possible to place measurement probes along the data path in the network, measurements were limited to end-to-end performance metrics (e.g. delay and goodput) as seen at the application layer. The low-level details of the transport system were ignored (due to the lack of access to system details), although future versions of the model may include such refinements. For a more detailed performance model, more probes are needed inside the transport system.

Modelling question

> Can we model speed-related performance characteristics of the transport system for different workload parameters?

A "black-box" view of the transport system (i.e. the SoD) is shown in Figure 6.2. Transport service is assumed available and reliable. Transport requests (packets) arrive to the system at its ingress point, are served with a particular speed and depart from the system at its egress point at the same order.

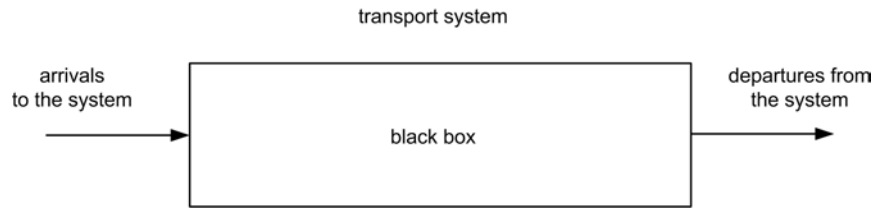



Figure 6.2 "Black box" view of a transport system

Performance criteria and parameters

The primary performance parameters of interest are speed-related such as delay, jitter and goodput. In this section we consider only the delay performance parameter, which is defined as the packet transportation time over the system (the SoD). Recall section 4.5.2 for detailed definition.

Queuing Model

A queuing system is an adequate representation for modelling speed-related characteristics of the transport system. Therefore, the transport system is modelled as a queuing system consisting of one node (service facility) as shown in Figure 6.3. Packets of fixed size arrive at the SoD's ingress point at a fixed rate, and depart at the SoD's

egress point (see section 5.1). Although packets have a fixed size, their service time in the queuing model is a random variable, which corresponds to the transport delay through the network (the SoD).

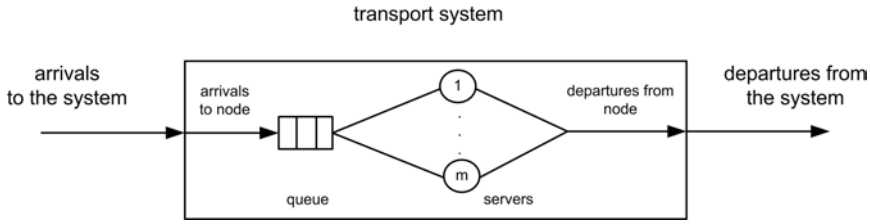



Figure 6.3 High-level queuing model of a transport system

Model assumptions

In this section the assumptions and the relevant analytical results relating to the above queuing model are presented. The queuing model of the transport system is fully characterized by the following:

1) Arrival process: packets (of given (fixed) size) arrive to the transport periodically at the fixed rate, which means that the arrival process is deterministic.

2) Service time distribution: service times are IID random variables (generally distributed) corresponding to the random transport delay through the network (the SoD).

3) Number of servers: there are *m* (m>=1) servers (since more than one packet may be going through the transport system simultaneously).

4) Buffer size: assumed to be infinite (or large enough) to accommodate the offered traffic (i.e. the arriving packets waiting for transport). This is the default value of the buffer size.

5) Population size: is infinite (since the arrival rate is fixed); that is, an open queuing system is considered. This is the default value of the population size.

6) Service discipline: the order at which packets are served at the serving entity is First Come First Served (FCFS). This is the default value of the service discipline parameter.

According to the Kendall's notation the queue representing the transport system is of type: *D/G/m/∞/∞/FCFS*. This implies, a deterministic packet inter arrival time, generally distributed service time, *m* servers, infinite buffer size, infinite population size, and FCFS

serving discipline. Note that if a default characteristic is used, then we may omit this characteristic from the above notation, hence the model is a *D/G/m* queuing system.

The key variables used in the analysis of the proposed queuing model are as follows (see Figure 6.4):

$\tau$ : interarrival time, which is the time between two successive arrivals; a random variable with a mean ($\bar{\tau}$) and variance ($\sigma_\tau^2$)

$\lambda$ : packet arrival rate (= $1/\bar{\tau}$)

s : service time, which is the delay of a single packet through the transport system; a random variable characterized by its mean ($\bar{s}$) and variance ($\sigma_s^2$)

$\mu$ : service rate (= $1 / \bar{s}$)

w : mean waiting time, that is, the mean time interval between a packet arrival and the instant its transport begins

r : mean response time (i.e. packet waiting time and transport delay; $r = w + \bar{s}$)

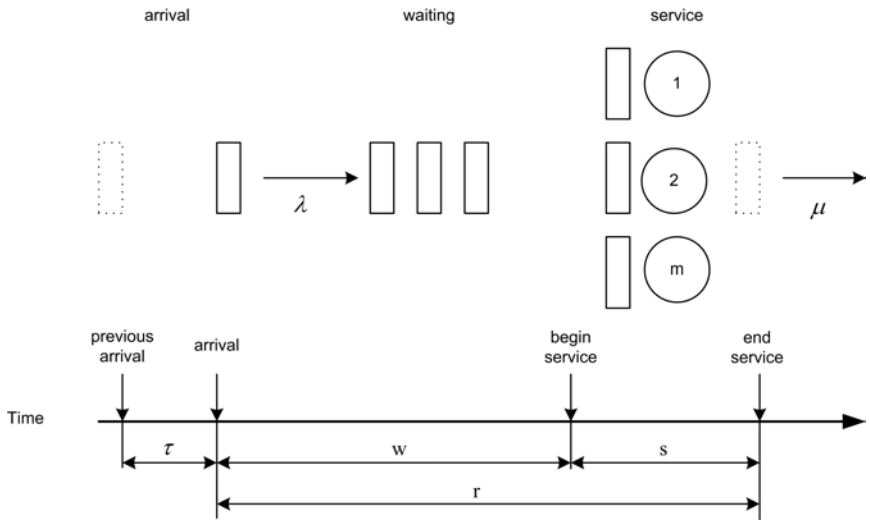



Figure 6.4 Random variables in a queuing system

Delay bounds in the G/G/m queue

In the following, we present known results on the delay bounds for the *G/G/m* queue. These general results will be later specialized to fit our specific model assumptions.

For this queuing model, simple closed-form expressions for the upper and lower bounds on the mean waiting time are given below [Klei1976]:

$$\frac{\lambda * \frac{\sigma_s^2}{m^2} - \frac{\bar{s}}{m} * (2 - \rho)}{2 * (1 - \rho)} - \frac{(m-1) * \sigma_s^2}{2 * m * \bar{s}} \leq W \leq \frac{\lambda * (\sigma_\tau^2 + \frac{\sigma_s^2}{m^2} + \frac{(m-1) * (\bar{s})^2}{m^2})}{2 * (1 - \rho)}$$

Where $\rho = \frac{\bar{s}}{\bar{\tau} * m} = \frac{\lambda * \bar{s}}{m}$ is the server utilization. For a stable system: $\rho < 1$.

To determine the queuing model parameters (namely $\lambda$, m, $\bar{s}$ $\sigma_s^2$) we make use of the measurements results in the Experiments 1 and 9 (see section 5.2.1). Results from Experiment 1 (confirmed service type) indicate the quantitative behaviour of the system with no contention, while the results from Experiment 9 (unconfirmed service type) provide the system behaviour under contention. For these two experiments, the following holds:

1) focus on the SUT's uplink behaviour

2) one fixed packet size (equal to 524 Bytes or 4192 bits) for parameterisation, validation and verification

3) only light-load scenarios are considered to avoid buffer overflow and to guarantee a stable operation for the unconfirmed service scenario.

Figures 6.5 and 6.6 present the measurement results.

We conclude that for $\lambda \leq 12$ packets per second, the system is stable ($\rho < 1$) and there are no queuing delays. From the above table (Figure 6.6) we see that for a stable system, the number of servers is $m \geq 3$. For $\lambda = 13$ there are no measurement results available. For $\lambda \geq 14$, the system is stable, but queuing is observed. The system response time is increasing (i.e. waiting time is increasing) until the point where the application stops sending data to the transport system by the TCP transport protocol. In this case, packets experience the maximum response (and waiting) time. The proposed model does not take into account this TCP behaviour, and therefore such congestion must be avoided when attempting to validate the model.

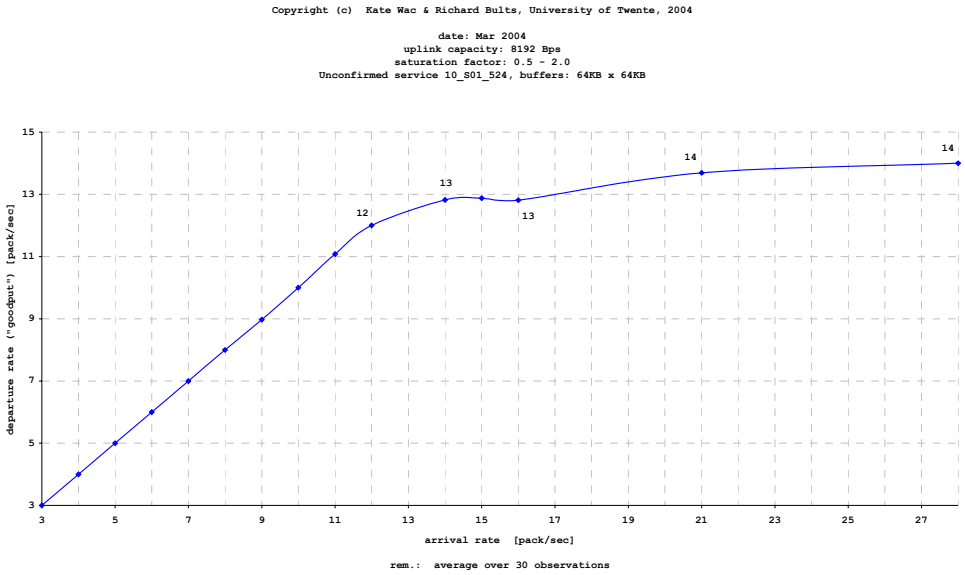



Figure 6.5 System goodput versus packet arrival rate

| Mean packet transport time $\overline{s}$ [s] | Mean transport rate [54] [Kbps] | Packet arrival rate $\lambda$ [pack/s] | Packet goodput rate [pack/s] | Packet goodput rate [Kbps] | Server utilization $\rho = \frac{\lambda * \overline{s}}{m}$ |
|---|---|---|---|---|---|
| 0,184 | 22,25 | 7 | 7 | 28,65 | 1,288 / m |
| | | 8 | 8 | 32,75 | 1,472 / m |
| | | 9 | 9 | 36,84 | 1,656 / m |
| | | 11 | 11 | 45,03 | 2,024 / m |
| | | 12 | 12 | 49,125 | 2,208 / m |

Figure 6.6 Measurement results used for model parameterization

To verify our assumption regarding the number of servers *m* in the proposed D/G/m queuing model, the upper bound on the mean waiting time is calculated from the above equation for $\lambda \leq 12$ packets per second (Figure 6.7) and m = 3.

[54] mean transport rate = ( 524 * 8 ) / ( mean transport time *1024 ) [Kbps]

| **Packet arrival rate $\lambda$ [pack/s]** | **Packet waiting time (upper bound) [s]** |
|---|---|
| 3 | 0,0001 |
| 7 | 0,0004 |
| 8 | 0,0005 |
| 9 | 0,0007 |
| 11 | 0,0011 |
| 12 | 0,0014 |

Figure 6.7 Upper bounds packet waiting time

The lower bound of the waiting time gave for all cases negative result, which is a trivial conclusion and not presented in the table. The upper bound on the mean waiting time is less than 0,5% of the measured response (waiting + transport) time. This confirms that indeed there is hardly any queuing experienced by packets through the transport system. Therefore, we conclude that a proposed D/G/3 queuing system may be used to model the transport system. Comprehensive validation of the queuing model has not been possible because of the unavailability of data to test the transport system under contention.

Conclusion

In this section we have shown that speed-related performance characteristics of the transport system may be modelled by D/G/3 queuing system (G denotes service time characterized by its mean $\bar{s}$ = 0,184 ms and variance $\sigma_s^2$ = 0.00618). However, this model is validated only for the uplink when there is no contention. It is also of interest to analyse the behaviour of the transport system on the downlink and to include contention. The behaviour of the system for packets sizes other than 524 Bytes could be observed by means of measurements to further validate the model.

## 6.2 Performance Evaluation Base

In chapter 4 and 5, we explained that workload generator SE's are used to simulate the transport of BANip packet with different sizes. We took all the measurements, but not presented nor analysed them so far. Therefore, in this section we provide the basic information for the SUT performance evaluation for a single service user scenario.

Information consists of a set of combined conclusive graphs, i.e., graphs holding the combined results of all measurements. We present graphs for the SUT uplink and downlink in the following order: firstly, graphs presenting the SUT delay behaviour and then graphs presenting the SUT goodput behaviour. The graphs' presentation order, i.e., the delay followed by goodput, is dictated by the fact that the goodput performance measure is derived from the delay performance measure. There are two subsections dedicated for the graphs presentation activity:

1) Presentation of the delay and goodput behaviour conclusive graphs for the SUT uplink and downlink (section 6.2.1)
2) Description on how these graphs are derived (section 6.2.2)

The order of these subsections seems to be inverted [55] logically, but it has been done with a purpose. Following the saying: "One picture tells you more than 1000 words", we choose to have the conclusive graphs in hand and then explain the reader how these graphs were derived.

From a MobiHealth system perspective, the SUT uplink behaviour is more important than the downlink behaviour[56]. However, there are experiments performed for both: the SUT uplink and downlink. Understanding of the SUT behaviour in both directions is of importance for at least two reasons. Firstly, it may be that the downlink behaviour influences the uplink behaviour. Second reason is that future versions of the MobiHealth system may need to use the downlink for the remote BAN control, and therefore the performance characteristics of the SUT downlink need to be disclosed. Therefore, the following graphs provide the SUT delay and goodput behaviour for both directions and the following sections of this chapter provide the behaviour analysis for both directions.

### 6.2.1 Conclusive graphs

In this section, the delay and goodput behaviour of the SUT's uplink and downlink is presented in conclusive graphs, i.e. graphs holding the combined results of measurements. For the analysis of the SUT behaviour we use the results obtained from Experiment 1[57]; the benchmark for all other performance evaluation activities. The workload generated for the SUT consists of user confirmed SE's with different SP sizes for uplink (SP_u) and downlink (SP_d). The details on the generation of the different SE's are explained in section 5.3.1.

> *We use the notation of SP_$u_{subscript}$ (and SP_$d_{subscript}$) throughout the report. This notation indicates particular characteristic of the SP as given in the subscript. For example the notation SP_$u_{size}$ and SP_$d_{size}$ is used, which means the SP in uplink / downlink with a packet size as indicated in the subscript. There is a "fixed" packet size (i.e. of a particular value) e.g. SP_$u_{174}$ (174 Bytes), or a "variable" packet size*

[55] reversed in position, order, or relationship
[56] MobiHealth BAN acts as a producer of data, while the BEsys acts as a consumer of (this) data; there is a reverse producer-consumer paradigm, which implies that the uplink behaviour is more important than downlink
[57] Experiment 1 uses "user confirmed" SE's as workload; recall the BANip application protocol is based on a "user unconfirmed" service. We consider a "user confirmed" service as a combination of two "unconfirmed services", one for uplink and one for downlink communication services. Hence, experiment 1 can be used to simulate BANip packet sizes.

*($SP\_u_{variable}$). There are also Service Primitive characteristics indicated in the subscript e.g. SP's delay $SP\_u_{delay}$, SP's rate $SP\_u_{rate}$ or goodput $SP\_u_{goodput}$.*

*Note In the following figures the measurement points are connected, however the presented behaviour should not be interpreted as a continuous function.*

**SUT uplink behaviour**

In this section, we present the average SUT's uplink delay behaviour as a function of SP_d size i.e. the packet size transported in the uplink direction. To study this behaviour we utilized uplink statistical data of all executed SEs. However, to facilitate the data presentation we decided to group SEs of the same packet size characteristics. Our selection criterion for a group of SEs was that the group comprises SEs characterized by: fixed SP_u size and arbitrary SP_d size. Because in our measurement activity we have 20 different SP_u sizes, we had 20 groups of SEs for further analysis. Moreover, because in our measurement activity we have 20 different SP_d sizes, each SEs' group numbers exactly 20 SEs. For each group of SE we can present its statistical data separately and provide a separate uplink delay function in relation to SP_d size. Therefore, having 20 different groups of SEs, we distinguished 20 different delay functions with arguments of 20 different SP_d sizes.

Figure 6.8 presents the average SUT's uplink delay behaviour as a function of SP_d size. There are 20 functions distinguished; each of them has a fixed SP_u size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding the statistical average of the primary performance parameter *delay*[58] (section 4.5.2) in milliseconds.

In the following paragraph, we present the SUT's uplink average delay and jitter behaviour as a function of SP_u size i.e. packet size transported in the uplink direction. To derive this function, we utilized uplink statistical data of selected SEs. Our selection criterion for SEs was that it consists of: an arbitrary SP_u size and SP_d size fixed at 524 Bytes. Because in our measurement activity we have 20 different SP_u sizes, this way we selected 20 SEs for further analysis. We motivate our choice by the fact that for the support of m-health services we need to understand mainly the uplink behaviour of the SUT. Hence, for the selected group of SE we provide an uplink delay (and jitter) function in relation to SP_u size. Delay function has as an argument 20 SP_u sizes.

Figure 6.9 presents the SUT's uplink average *delay* and *jitter* (i.e. measured variance of the delay) behaviour as a function of SP_u size. The horizontal axis provides the values of the SP_u size (in Bytes). The SP_d size is fixed at 524 Bytes. The vertical axis presents the corresponding average uplink delay in milliseconds. The vertical (red) line indicates the *jitter* for each delay measurement point and should be interpreted as a range of the possible delay values for this point.

[58] Statistical random variable.

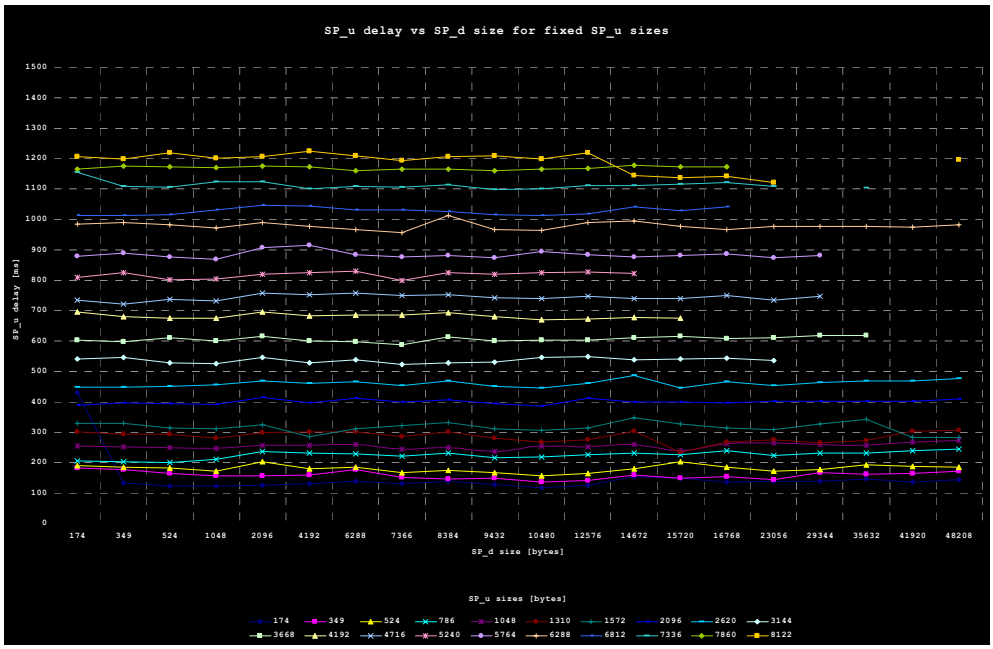



Figure 6.8 SUT uplink delay behaviour versus SP_d size

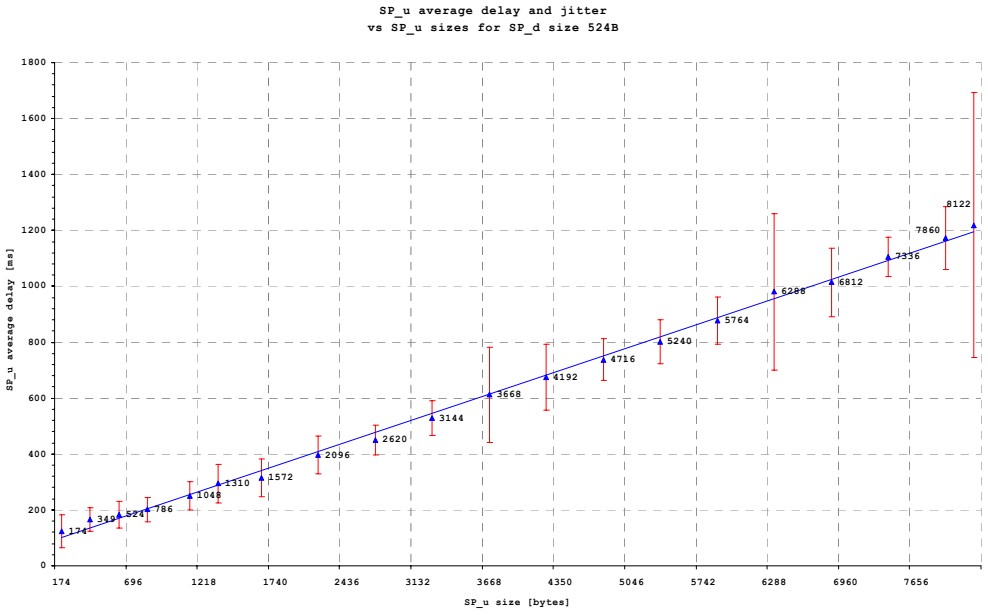



Figure 6.9 SUT uplink delay and jitter behaviour versus SP_u size

*Note: From the application developer point of view, the Figure 6.9 should be read as follows. Choose the application packet size and obtain from the graph the corresponding average uplink delay and minimum and maximum delay values. Reflect if the application can handle the provided delay range.*

In the following graphs the SUT uplink goodput (defined in section 4.5.2, a net SUT throughput measured at the application layer) and efficiency (a fraction of the nominal link capacity used for the transmission of an application data) graphs are presented. From the primary performance parameter *delay* we derived the goodput and efficiency performance parameters. Therefore, Figure 6.10 presents the SUT's uplink average goodput behaviour as a function of SP_d size. There are 20 functions distinguished; each of them has a fixed SP_u size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding statistical average of the derived performance parameter *goodput* in Kbps. Two red horizontal lines indicate the SUT capacity boundaries for the common (16 Kbps) and dedicated bearer (64 Kbps) in the uplink direction.

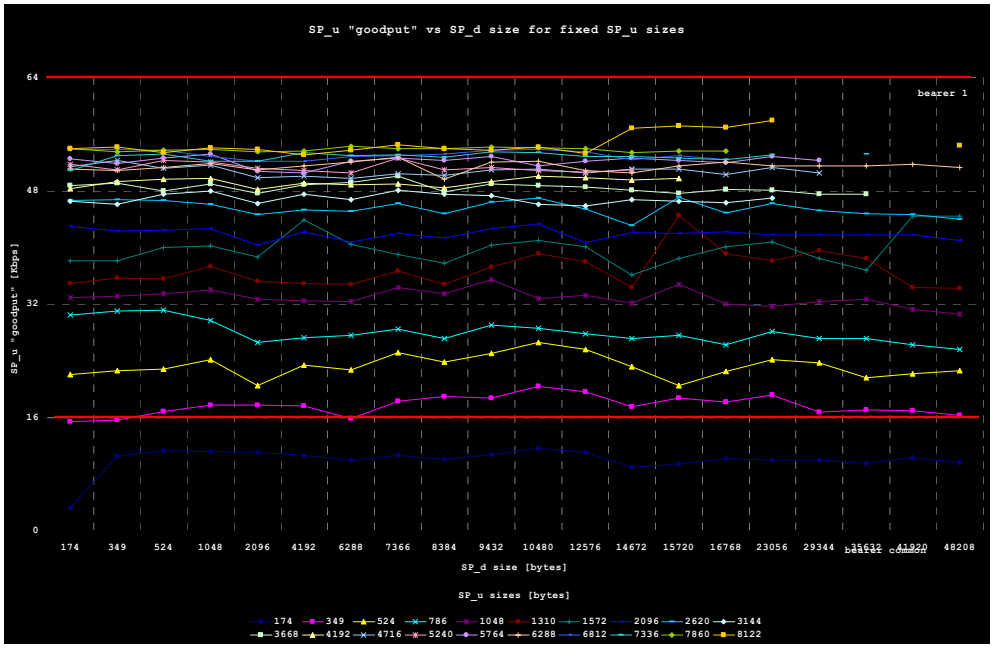



Figure 6.10 SUT uplink goodput behaviour versus SP_d size

Following the SUT jitter behaviour graph, we focus on the *efficiency* (derived performance parameter from *goodput*) analysis on the SUT uplink. Figure 6.11 presents three corresponding functions. The first function (blue) presents the SUT's uplink average *goodput* and *efficiency* behaviour as a function of SP_u size. The second function (black) presents the corresponding SUT's uplink average *throughput* behaviour as a

function of SP_u size. The horizontal axis provides the values of the SP_u size (in Bytes). The SP_d size is fixed at 524 Bytes. The vertical axis presents the corresponding average uplink goodput and throughput in Kbps. The *efficiency* is indicated on the right vertical axis as percentage of the nominal uplink dedicated bearer capacity (which is 64 Kbps).

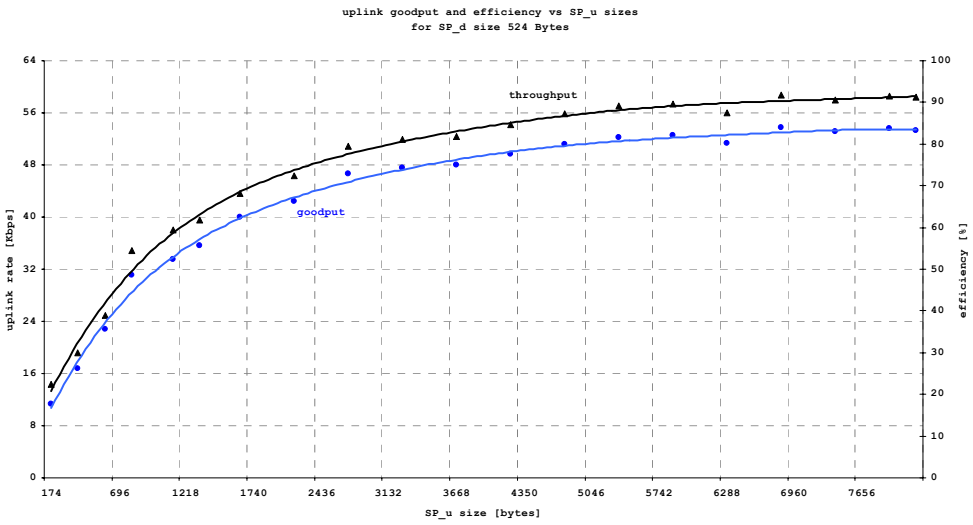



Figure 6.11 SUT uplink goodput, throughput and efficiency behaviour versus SP_u size

**SUT downlink behaviour**

In the following paragraph, we present the average SUT's downlink delay behaviour as a function of SP_d size; i.e. the packet size transported in the downlink direction. To derive this function we utilized uplink statistical data of all executed SEs. However, to facilitate the data presentation we decided to group SEs of the same packet size characteristics. Our selection criterion for a group of SEs was that the group comprises SEs characterized by: fixed SP_u size and arbitrary SP_d size. Because in our measurement activity we have 20 different SP_u sizes, we had 20 groups of SEs for further analysis. Moreover, because in our measurement activity we have 20 different SP_d sizes, each SEs' group numbers exactly 20 SEs. For each group of SE we can present its statistical data separately and provide a separate downlink delay function in relation to SP_d size. Therefore, having 20 different groups of SEs, we distinguished 20 different delay functions with arguments of 20 different SP_d sizes.

Figure 6.12 presents the average SUT's downlink delay behaviour as a function of SP_d size. There are 20 functions distinguished; each of them has a fixed SP_u size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding the statistical average of the primary performance parameter delay in milliseconds.

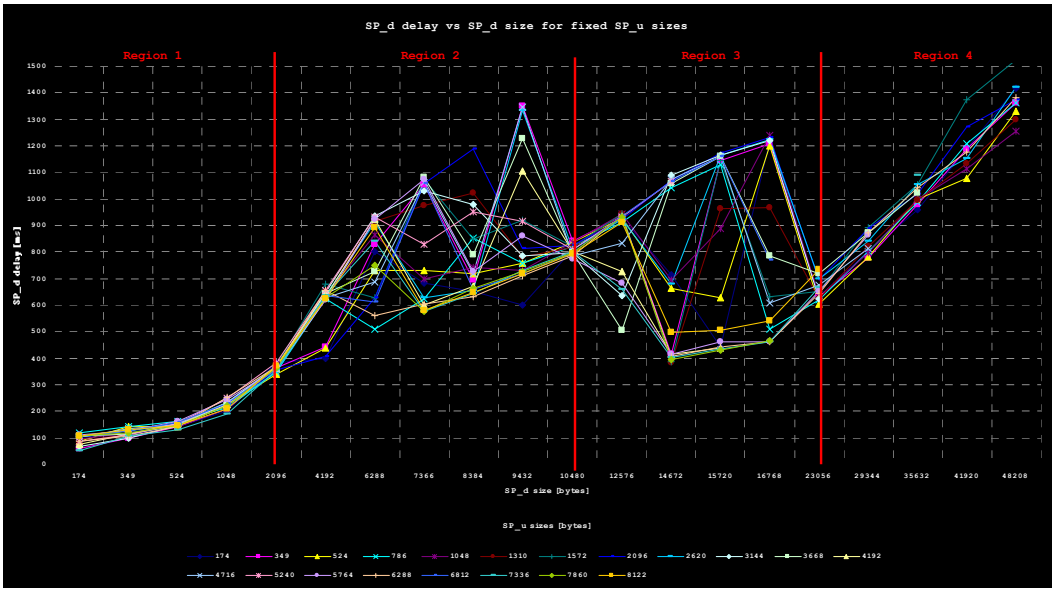



Figure 6.12 SUT downlink delay behaviour versus SP_d size

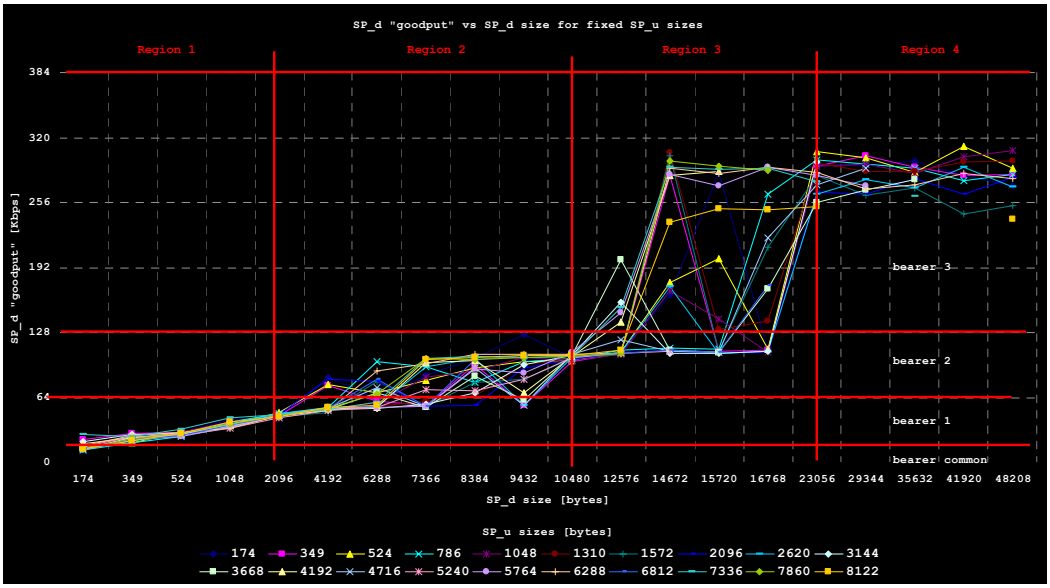



Figure 6.13 SUT downlink goodput behaviour versus SP_d size

Figure 6.13 presents the SUT's downlink average goodput behaviour as a function of SP_d size. There are 20 functions distinguished; each of them has a fixed SP_u size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding statistical average of the derived performance parameter *goodput* in Kbps. Four red horizontal lines indicate the SUT capacity boundaries for the common (16 Kbps) and dedicated bearer (64 Kbps, 128 Kbps, 384 Kbps) in the downlink direction.

### 6.2.2 Derivation of Conclusive Graphs

The presented conclusive graphs holding the combined results of SEs' measurements, have been derived from the raw data obtained in the Experiment 1. The execution of Experiment 1 is described in section 5.4.2. The successfully executed entries in the 20x20 matrix are presented in green colour in Figure 5.31 and represent 500 observations (representing one sample) of a given SE execution; in other words, each sample comprises 500 observations for a given matrix entry. The first 5 observations are considered outliers and therefore removed from the sample to eliminate the initial changing behaviour of the links[59]. The remaining observations are used in the statistical calculations for the average *delay*. Later change in bearers has an impact on the variance of the delay (i.e. *jitter*) for both uplink and downlink.

> *Note: Due to the (human) resources limitations we were not able to develop statistical analysis software that automatically recognizes a bearer change in a sample and only consider the average delay for a period where the bearer is fixed.*

For all individual SEs (i.e. entries in cell) "raw data" graphs were generated (Volume 2, T01 graphs directory). Statistical calculations were performed to obtain the average (i.e. mean) delay, its standard deviation (i.e. jitter) and the accuracy of the average delay for a 95% confidence interval. For each SE we derived the statistical values for the uplink, downlink and SE delay behaviour.

Figure 6.14-17 present four SE raw data graph examples that are used to explain the *delay* behaviour of the SUT and the high variance of the average *delay*. In the Figures, on the horizontal axis there is an observation number (1 to 500) within the sample, while on the vertical axis, the corresponding value of delay in milliseconds can be read. For each observation, the delay's measurement points are given separately: for uplink (red colour), downlink (green colour) and total SE delay (i.e. blue colour). The delay measurements points represent 500 consecutive observations, in the order as they have been measured. As a result, there are three lines distinguishable in the delay plot area, meaning *trends* of the equivalent delay.

In each figure, informational data is given above the delays' plot area. *Start* and *stop* labels indicate the plotted sample's measurements timeframe. Below the plot area, there is delays' statistical data for this sample, i.e. average uplink, downlink and SE delays, their jitter and the accuracy of the average delays for a 95% confidence interval.

[59] Uplink and downlink change from the common bearer to the first dedicated bearer.

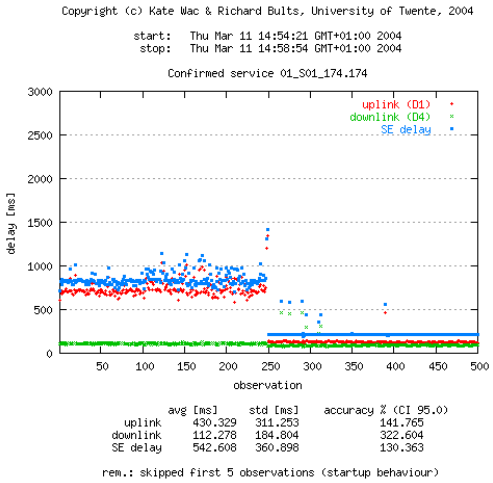



Figure 6.14 SUT uplink converging behaviour

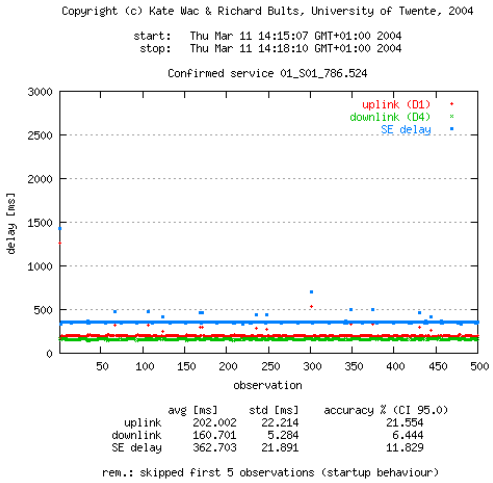



Figure 6.15 SUT up/downlink predictable behaviour

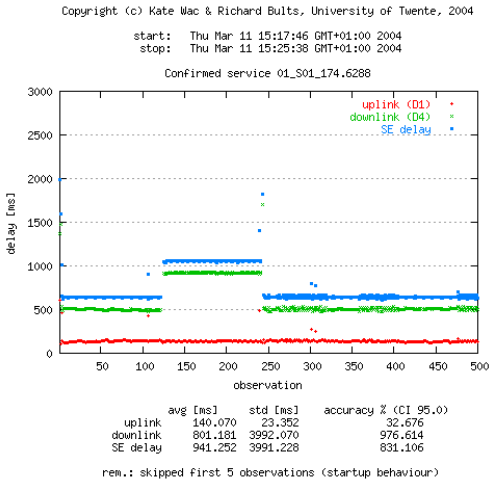



Figure 6.16 SUT downlink alternating behaviour

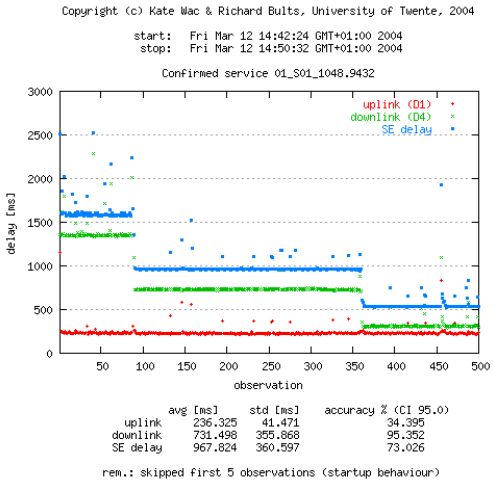



Figure 6.17 SUT downlink converging behaviour

Figure 6.14 (SE: $SP\_u_{174}$ and $SP\_d_{174}$) presents the converging behaviour of the SUT. For the first few observations, the SUT assigns the common bearer for uplink and downlink. After these few observations, the downlink has been assigned with the dedicated bearer 1, which remains for the rest of the data under transport. For the uplink, for observations 1 to about 250, the SUT has assigned the common bearer. Only after (about) 250 observations, the SUT assigns the dedicated bearer 1 for the uplink. Another finding

based on Figure 6.14 is, that the usage of a common bearer has a negative impact on the delay variation.

From Figure 6.15 (SE: SP_$u_{786}$ and SP_$d_{524}$) we deducted that both links have been assigned with the dedicated bearer 1 after the few first observations. Both links display predictable delay behaviour and as a result have a low variance. This is desired converging behaviour of the SUT.

From Figure 6.16 (SE: SP_$u_{174}$ and SP_$d_{6288}$) we observe the downlink's alternating behaviour. Initially both links have been assigned with the dedicated bearers (dedicated bearer 1 in uplink, bearer 2 in downlink) after the few first observations. The downlink has been assigned with the dedicated bearer 2 (observations 10 to about 120), then reassigned with the dedicated bearer 1 (observations about 120 to about 240) and then reassigned back to the dedicated bearer 2 (observations about 240 to 500). While the downlink has been assigned with the dedicated bearer for all observations, it behaves accurately.

Figure 6.17 (SE: SP_$u_{1048}$ and SP_$d_{9432}$) presents the SUT behaviour with the downlink converging behaviour. Both links have been assigned with the dedicated bearer 1 after the few first observations. The bearer's assignment starts from the dedicated bearer 1 (observations 10 to about 80), and through dedicated bearer 2 (observations about 80 to about 360) and it converges to the dedicated bearer 3 (observations about 360 to 500).

> *Note: The SUT bearer changes are confirmed by calculation of the link goodput in terms of net obtained bits per second and correlation of the goodput to the nominal capacity of the bearer (recall section 4.3.2, Figure 4.8).*

After the experiment has been executed, the statistical data for all SEs (i.e. entries in the 20x20 matrix) are collected in a *StatDataFile*. A section of this file is presented in Figure 6.18. The meaning of the columns is as follows: D1 = uplink delay, D2 = downlink delay, SE_D = SE delay, Ac = accuracy for the 95% Confidence Interval, Ul = uplink, Dl = downlink goodput. Row for SE of size "174.174" (meaning: SE: SP_$u_{174}$ and SP_$d_{174}$) is the data used to provide the raw data graph in Figure 6.14 with the statistical data. The conclusive graph in Figure 6.8 is generated from the set of rows; set of the SEs' results, namely from the SEs where SP_u size is fixed (column 1) and SP_d is variable (column 2). In Figure 6.8 the data given in column 3: an average uplink delay (avg D1) is plotted. Similarly Figure 6.12 uses the same set of rows to plot the average downlink delay (avg D2, column 4).

Figure 6.9 is generated from rows containing SP_d sizes 524 Bytes[60] and the delay's accuracy figure presented in column 9 (Ac D1). The *jitter* (i.e. variation of delay) is however presented as a vertical (red) line with a length indicating possible delay values between a minimum and maximum value. The minimum and maximum values are calculated with the accuracy number presented in column 9. For example: The average uplink delay for SP_$u_{174}$ and SP_$d_{524}$ is 124 ms and the accuracy of this measurement is 48%. The *jitter* ranges from 124*(1-0.48) = 65ms to 124*(1+0.48) = 184ms.

[60] The size of the SP_d is not influencing the SUT uplink behavior (section 6.3.1 – downlink delay analysis).

| SP_u | SP_d | avg D1 | avg D2 | avg SE_D | std D1 | std D2 | std SE_D | Ac D1 | Ac D2 | Ac SE_D | Ul_ goodput | Dl_ goodput |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 174 | 174 | 430 | 112 | 543 | 311 | 185 | 361 | 142 | 323 | 130 | 3 | 12 |
| 174 | 349 | 133 | 110 | 243 | 21 | 28 | 33 | 30 | 49 | 27 | 10 | 25 |
| 174 | 524 | 124 | 143 | 267 | 30 | 59 | 66 | 48 | 81 | 48 | 11 | 29 |
| 174 | 1048 | 124 | 201 | 325 | 21 | 45 | 50 | 32 | 44 | 30 | 11 | 42 |
| 174 | 2096 | 125 | 385 | 510 | 21 | 244 | 246 | 34 | 124 | 94 | 11 | 44 |

Figure 6.18 Example of statistical data

The statistical data in the *StatDataFile* was also used to generate graphs presenting the SUT uplink, downlink and cumulative (SE's) delay behaviour for measurement set where SE's have a fixed SP_u size and variable SP_d sizes (Volume 2, T01 conclusive graphs directory). Figure 6.19 presents an example of this graph where SEs have fixed $SP\_u_{174}$ and a variable SP_d size.

red line (bottom) : SP_u delay analysis for $SP\_u_{174}$ and $SP\_d_{variable}$.
green line (mid) : SP_d delay analysis for $SP\_u_{174}$ and $SP\_d_{variable}$.
blue line (top) : SE delay analysis for $SP\_u_{174}$ and $SP\_d_{variable}$.

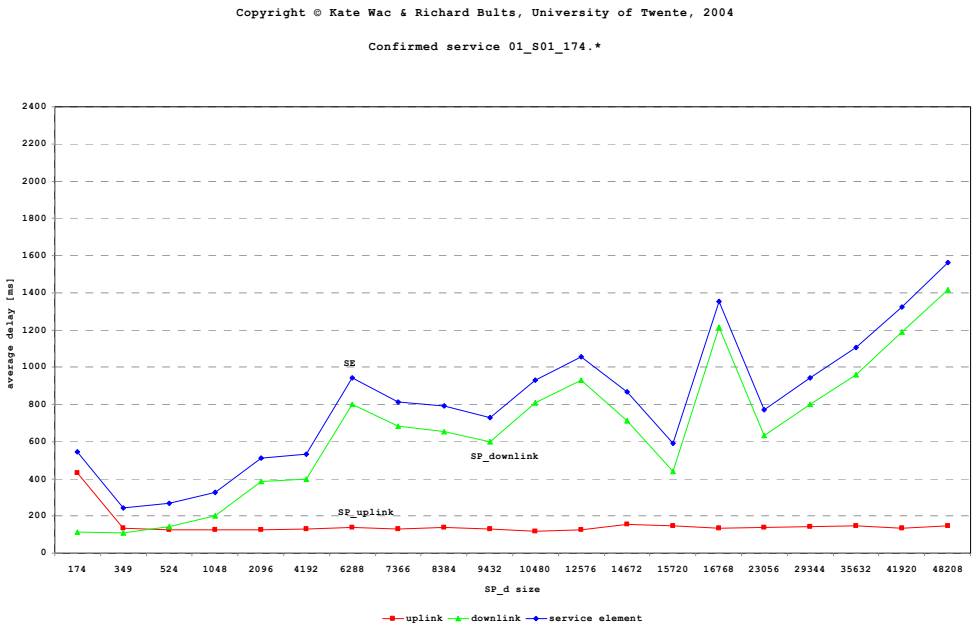



Figure 6.19 SUT delay behaviour versus SP_d size

The statistical data used in the delay analysis, was used to obtain the derived SUT performance parameters goodput, throughput and efficiency.

*Note: In our calculation the nominal link capacity is the maximum capacity of the SUT for a particular bearer as given by Vodafone. Section 4.3.2 provides the nominal link capacity (denoted as $SUT_{capacity}$) for the uplink (16 Kbps, or 64 Kbps) and for the downlink (16 Kbps, 64 Kbps, 128 Kbps or 384 Kbps).*

The goodput is defined as:

$SUT_{goodput}$ = net SUT throughput measured at the application layer

$SUT\_u_{goodput} = SP\_u_{size} / SP\_u_{delay}$

$SUT\_d_{goodput} = SP\_d_{size} / SP\_d_{delay}$

units of measure: [Kbps] = [b] / [ms]

The throughput is defined as:

$SUT_{throughput}$ = gross SUT throughput measured at the application layer

$SUT\_u_{throughput} = (SP\_u_{size} + protocolOverhead) / SP\_u_{delay}$

$SUT\_d_{throughput} = (SP\_d_{size} + protocolOverhead) / SP\_d_{delay}$

units of measure: [Kbps] = [b] / [ms]

*Note: A goodput is the link capacity calculated without taking into account the lower protocol layers overhead. A throughput is the link capacity calculated with taking into account the lower protocol layers overhead. Therefore, goodput is less than the throughput. The throughput is less than the nominal link capacity.*

We define the *efficiency* as:

*A fraction of the nominal link capacity used for the transmission of an application data.*

$SUT\_u_{efficiency} = ( SUT\_u_{goodput} / SUT\_u_{capacity} ) * 100$

$SUT\_d_{efficiency} = ( SUT\_d_{goodput} / SUT\_d_{capacity} ) * 100$

units of measure: [%] = [Kbps] / [Kbps]

During our measurements, we have successfully collected raw data for SEs in the format as in Figures 6.14 - 6.17. For each SE we took the statistical values for the uplink, downlink and SE delay and used it to compose the conclusive graphs. It is important to note that the conclusive graphs are provided with the average delay figures regardless the

variance of this delay. Moreover, to facilitate the analysis of the SUT's delay behaviour, we present the SUT's uplink and downlink behaviour separately. The uplink and downlink delay figures for particular SE are presented in separate conclusive graphs; Figures 6.8-6.10 for uplink and Figures 6.12 and 6.13 for downlink.

For the uplink delay analysis Figure 6.8 has been generated, from which the uplink goodput characteristic presented in Figure 6.10 follows. Therefore, Figure 6.10 presents the relation between SP_u goodput and a variable SP_d size. The SP_u is fixed. Two red lines placed in Figure 6.10 indicate the (maximum) nominal uplink capacity of the SUT for the common (16 Kbps) and dedicated bearer (64 Kbps).

Figure 6.11 is generated based on the SUT uplink delay figure (Figure 6.9) for the variable size SP_u and a fixed SP_d size of 524 Bytes. Figure 6.11 presents three functions. The first one (blue) presents the SUT uplink goodput and efficiency behaviour as a function of a SP_u size. The SP_d size is fixed ate 524 Bytes. The second function (black) presents the corresponding SUT's uplink average *throughput* behaviour as a function of SP_u size. The goodput and throughput are indicated on the left vertical axis (in Kbps), while the efficiency is indicated on the right vertical axis (in %). Efficiency has been calculated with respect to the uplink $SUT_{goodput}$ for the dedicated bearer, which is 64 Kbps (from the delay analysis it's known that while executing SEs with SP_d size 524 Bytes the dedicated bearer has been used).

Analogously to the uplink analysis, for the downlink delay analysis Figure 6.12 has been generated, from which the downlink goodput characteristic presented in Figure 6.13 follows. Therefore, Figure 6.13 presents SP_d goodput as a function of SP_d size. The SP_u is fixed. Four red lines placed in Figure indicate the SUT's nominal uplink capacity boundaries for the common (16 Kbps) and dedicated bearers (64 Kbps, 128 Kbps and 384 Kbps).

## 6.3 Delay Analysis

The delay analysis of the SUT aims to fulfil the first and second objectives described in section 4.1. Recall these objectives:

1) To characterize the quantitative behaviour of the SUT as a MobiHealth transport service.
2) To optimise the maximum BANip packet size and packet rate for a specified (maximum) delivery time.

We combine these objectives into one question that focuses on the specific goal of the delay analysis:

> What is the influence of the "user confirmed" SE's size on the *delay* behaviour of the SUT?

To support the analyses the SUT's uplink and downlink delay behaviour we use the time-sequence diagram presented in Figure 6.20 (based on the time-sequence diagram presented in Figure 4.15). It provides insight to the transportation mechanisms of SE's and all delay contributions (related to SoD and SUT) which constitute the (accumulated) SE transport delay.

*Note: The forthcoming explanations assume that the SUT's uplink is switched to the first dedicated bearer unless explicitly mentioned. The SUT's downlink can be switched to any of the four supported bearers.*

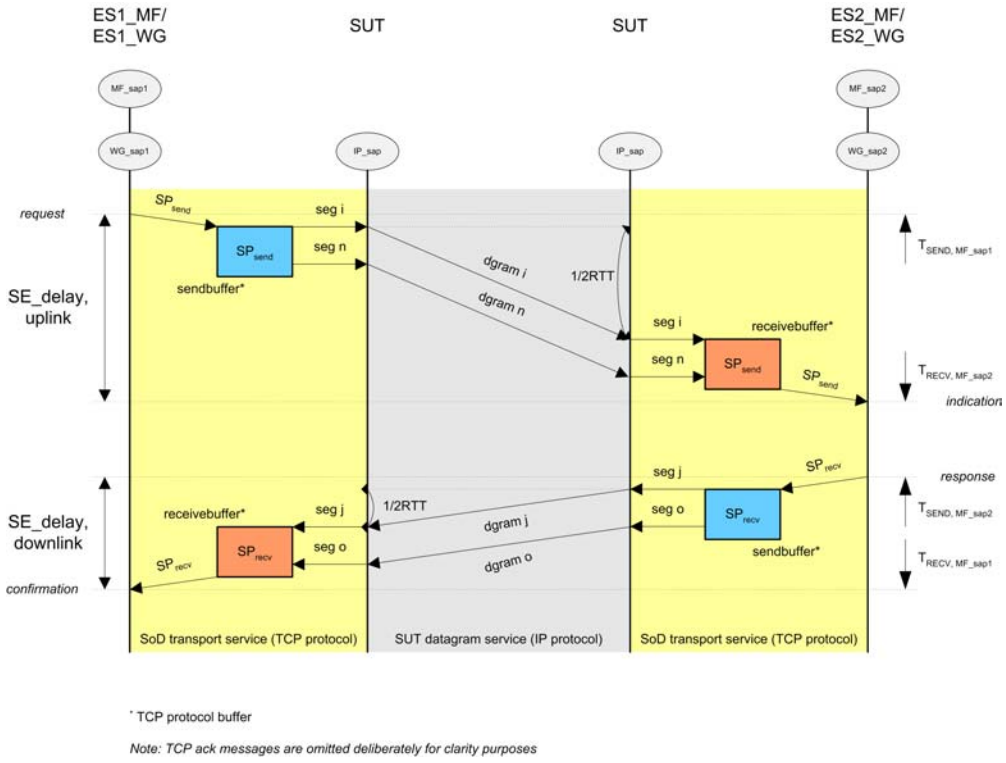



Figure 6.20 Time-sequence diagram of SE execution

Explanation of transport mechanisms:

A SE contains two time-related SPs: SP_send and SP_recv. Firstly, the SP_send is executed. It represents the send action of a packet (i.e. application layer message) in the SoD/SUT uplink direction. Secondly, the SP_recv is executed, representing the send action of a packet in the SoD/SUT downlink direction.

Workload Generators control and execute the actual exchange of packets at their WG_saps. Each packet is timestamped by a Measurement Function at the MF_sap

when the packet is forwarded to the TCP protocol; this protocol implements the (i.e. lower level) transport service provided by the SoD.

The TCP transport service has buffering capacity at the sending and receiving side (i.e. send and receive buffer, denoted in blue and orange coloured boxes) for packets traversing the SoD. The SoD's TCP protocol controls and executes the transportation of each packet. The TCP protocol entity at the sender side decomposes each packet into a stream of TCP segments, which are forwarded to the IP datagram service provided by the SUT. The SUT's IP datagram service (grey area in Figure 6.20) transports IP datagrams between its IP_saps, implementing a (lower level) transport service for the TCP protocol entities of the SoD sender and receiver sides.

The yellow areas in Figure 6.20 denote the execution of the TCP transport service and indicate transportation of segments within the SoD. The TCP protocol entity at the receiving side collects and reassembles the transported segments into the (original) packet, and passes it to the Workload Generator SAP (e.g. WG_sap2).

The Workload Generator performs several checks on the received packet and forwards it to the Measurement Function's SAP (e.g. MF_sap2). Finally, the Measurement Function time stamps the received packet and a one-way transportation of a packet is completed. The mechanisms described above also hold for the transportation of a packet from the receiver to the sender. However, in this case the SUTs downlink is used.

Explanation of SE transport delay:

Figure 6.20 shows that the uplink contribution to the SE transport delay (SE_delay, uplink) is calculated by subtracting $T_{RECV,\ MF\_sap2}$ from $T_{SEND,\ MF\_sap1}$. A closer look at Figure 6.20 reveals the individual delay contributions of (under lying) transport services implemented by the SoD and SUT.

We are interested in the delay analysis of the SUT and therefore assume the SoD delay contributions (yellow areas in Figure 6.20) caused by the TCP transport service to be zero. This is valid if the computer systems that implement the TCP protocol entities do not cause any measurable delays (less than 10 ms, section 5.5.4) due to local resource problems. Therefore, we dedicate all computer systems used for the measurement execution to this single task and consider the processing time for the TCP transport service (including buffering algorithms) on these systems within the 10 ms bracket. We provide a highly detailed argumentation in the rectangular area below.

Each packet generated by the sending Workload generator is time stamped by a Measurement Function at the MF_sap when the packet is forwarded to the WG_sap1 (i.e. TCP socket). The time it takes to put the packet into the TCP send buffer is zero. The TCP send buffer is having enough buffer capacity to store the entire packet for transportation. The TCP receive buffer is having enough buffer capacity to store the entire packet. The time it takes to read packet from the TCP receive buffer is zero. The receiving Workload Generator polls the WG_sap2 (i.e. TCP socket) and subtracts the message. The subtraction time equals zero. The receive timestamp is taken by the Measurement Function when the packet is successfully subtracted.

Furthermore, we assume that the SUT is not congested; hence, there is no need for TCP to activate its congestion control algorithm. This assumption is based on a high probability that we are the only persons generating (work) load on the UMTS part of the SUT. In addition, we have to assume that the rest of the SUT (i.e. the core part of the V3GNL network) is not congested.

Note, that the explanation for the SE uplink delay described above also hold for the SE downlink delay.

Closer observation of the grey area in Figure 6.20 indicates two delay components that constitute the SUT transport delay: "½ RTT" and a number of "dgram". These delay components are known as *propagation delay* and *transmission delay* respectively [Kuro2001, section 1.6]. Our evaluation system does not have the functionality to measure the propagation delay. It can only measure the SE_delay for the SUT's uplink or downlink.

The quantification of both delay components is important for the (forthcoming) delay and bottleneck analysis. Therefore, we executed a separate measurement action to estimate the propagation delay; the average propagation delay of the SUT's uplink (first dedicated bearer) is 86 ms. Providing an estimation of the SUT's downlink propagation delay is omitted due to: 1) the bearer switching behaviour of the SUT's downlink under applied workload, and 2) the relative importance of the downlink to the MobiHealth system. We provide a detailed description of the activity to estimate the propagation delay for the SUT's uplink in the rectangular area below.

Computer system "nb" uses a PC card UMTS terminal (section 4.7, Figure 4.22) and the "web.vodafone.nl" APN to access host 130.89.10.47 on the UTnet via the Internet. The ICMP protocol based *ping* command is used to generate workload to the SUT. The first dedicated bearer of the SUT's uplink is selected by sending 10 echo requests of 2048 Bytes from "nb" to 130.89.10.47. Immediately thereafter, 500 echo requests of 0 Bytes are send from "nb" to 130.89.10.47. The ping command provides an indication of the average roundtrip time upon completion (e.g. Minimum = 151 ms, Maximum = 411 ms, Average = 183 ms). The obtained average roundtrip time forms the bases for the removal of outliers (> 350 ms) with a non-repetitive character. As a result, the (new) statistical average round trip time for was calculated over 493 observations. The round trip time obtained for 0 byte ping packets is: 181 ms (std 9,08, accuracy for 95% confidence interval 9,84). The transmission time for 36 Bytes (headers of ICMP + IP + PPP) over a 64000 bps uplink (dedicated bearer) is 4.5 ms. The ping packets traversing over the SUT's uplink and downlink caused the SUT to assigns the first dedicated bearer for both links (based on analysis of our performance measurements data). Therefore, the SUT is considered as a symmetrical transport system for this particular case. Hence, the propagation delay of the uplink is equal to the round trip time for 0 byte ping packets minus two times the (previously) calculated transmission time, divided by two: (181ms – 2 * 4.5ms) / 2 = 86 ms

Conclusion

The SE transport delay is the sum of the SE_delay for the uplink and downlink and constitutes of the SUT uplink and downlink transport delay. The accumulated uplink SE transport delay consists of an 86 ms propagation delay (½ RTT) and a variable transmission delay, the accumulated downlink SE transport delay consists of a variable (unspecified) propagation delay and a variable transmission delay.

Figure 6.20 and the related explanations of the transport mechanisms and SE transport delay contributions, provide the stage for the forthcoming delay analysis of the SUT. The rest of this section is divided into two sections:

1) uplink delay behaviour analysis (section 6.3.1) based on the conclusive combined graphs presented in section 6.2

2) downlink delay behaviour analysis (section 6.3.1) based on the conclusive combined graphs presented in section 6.2

### 6.3.1 Uplink Behaviour

The SUT uplink delay behaviour's analysis is based on Figures 6.8 and 6.9 and lead to the following conclusions. For a fixed SP_u sizes, SP_d size does not influence the SUT uplink delay behaviour. This rule has an exception, because for SP_u size equal 174 Bytes and SP_d size 174 Bytes, after about 250 observations the SUT changed assignment of the bearer from the common bearer to the first bearer (Figure 6.14). For SP_d sizes bigger than 174 Bytes, the change of bearers occurs after a few observations (Figure 6.15).

Based on a quantitative analysis of SEs' raw data results, we conclude that the size and rate of data transported by the SUT's uplink and downlink cause the changing behaviour of the UMTS transport subsystem i.e. *bearer assignment*, and therefore the change in the observed overall SUT capacity. For any standalone request for data transport, the SUT always first assigns the common bearer in both directions, and then it may assign dedicated bearer(s) adequate to the data traffic under transport (i.e. the SUT has a "learning" behaviour). The bearer assignment behaviour is always gradual, i.e., the SUT always assigns only the contiguous bearer (to the current one) in the uplink and/or downlink.

The bearer assignment behaviour is considered to be normal if it converges to an adequate dedicated bearer and this bearer is going to be used for the remaining part of a data transfer (Figure 6.14, 6.15 and 6.17). In 99.8% of the executed performance measurements this behaviour is validated. Only the behaviour exhibited in Figure 6.16 showed an alternating behaviour of the downlink between two contiguous bearers.

We deduct from the Figure 6.9 that the uplink average delay ($SP\_u_{delay}$) increases linearly with the SP_u size. The general form of the SUT uplink delay function of SP_u packet size is given as follows:

$$SP\_u_{delay} = \alpha * SP\_u_{size} + C$$

The coefficient $\alpha$ is estimated from Figure 6.9 with use of the simple mathematical formulas, as given below.

$$\alpha = \frac{SP\_u_{delay8122} - SP\_u_{delay174}}{SP\_u_{size8122} - SP\_u_{size174}} \left[ \frac{ms}{Bytes} \right]$$

The constant $C$ may be obtained from the figure as an average distance of (20) measurement points from the function $SP\_u_{delay} = \alpha * SP\_u_{size}$

$$C = (\frac{1}{20} \sum_{i=1}^{20} SP\_u_{delay_i} - \alpha * SP\_u_{size_i}) [ms]$$

This formula provided the value of C = 98.10 ms. However, from Figure 6.20 we conclude that C = 0.5*RTT and we estimated, by means of dedicated measurements, the value of C = 86.00 [ms]. Therefore, we considered this value for further calculations. Consequently, the refined formula for the delay $SP\_u_{delay}$ as a function of the SP_u packet size is given as follows:

$$SP\_u_{delay} = 0{,}138 * SP\_u_{size} + 86 \text{ [ms]}$$

Units:

$$[SP\_u_{delay}] = \left[\frac{ms}{Bytes}\right] * [\text{Bytes}] + [\text{ms}] = [\text{ms}]$$

As we said, from Figure 6.9 we can see that the uplink average delay ($SP\_u_{delay}$) increases linearly with the SP_u size. This is an expected behaviour. For all SP_u sizes, the SUT delay behaviour consists of a transmission delay and a propagation delay (Figure 6.20).

$$SP\_u_{delay} = SP\_u_{transmission} + SP\_u_{propagation}$$

The following relations hold:

$$SP\_u_{transmission} = 0{,}138 * SP\_u_{size} \text{ [ms]}$$
$$SP\_u_{propagation} = 0.5 * \text{RTT} = 86 \text{ [ms]}$$

For all SP_u sizes the propagation delay is constant and the transmission delay (throughout the SUT) is proportional to the SP_u size.

Another important observation is that for a small packet sizes the $SP\_u_{delay}$ propagation delay component is significantly bigger that the transmission delay component.

From the Figure 6.9 we can see the SUT uplink jitter behaviour, i.e. variation of delay of (identical) packets under transport. The jitter is due to the random delay, experienced by transported packets in the SUT components [Kuro2001]. Based on the Figure we conclude that the SUT uplink jitter is significant for all sizes of SP_u; it varies from 6% (for $SP\_u_{7663}$) up to 38% percent (for $SP\_u_{8122}$) of a delay value. This is an unwanted behaviour. We attribute the observed delay variations to the inherent mechanisms of the different communication subsystems within the SUT. Recall from chapter 4 that the SUT is a “black-box”, therefore there is no possibility to use measurement probes within the SUT. Hence, we can only speculate on possible sources for jitter and identify three of them: 1) UMTS bearer changes, 2) packet loss in one of the SUT subsystems (causing TCP retransmissions), and 3) unpredictable SUT delay behaviour due to resource problems in the SUT subsystems.

### 6.3.2 Downlink Behaviour

The downlink delay behaviour's analysis based on Figure 6.12 lead to the following conclusions. The downlink delay is a complex function of the SP_d size. Particularly, SP_d size does influence the SUT's downlink delay behaviour but the delay does not increase linearly with the SP_d size, as it was observed in case of the uplink behaviour.
There are 20 functions presented in the figure, and based on them we put forward the hypothesis on the SUT behaviour. According to us, the SUT's downlink delay behaviour may be characterized by fours different delay behaviour regions depending on SP_d sizes.

The first region (*Region 1*) we identified relates to small packets up to and including SP_d size of 2096 Bytes. In this region, the delay increases monotonically with the packet size and the delay value is very much predictable for all functions.

The second region (*Region 2*) relates to SP_d sizes bigger than 2096 Bytes and smaller than 10480 Bytes. In this region, the delay is highly variable. For a SP_d equal 10480 Bytes, the delay value is very much predictable for all functions.

The third region (*Region 3*) relates to SP_d size bigger than 10480 Bytes and smaller than 23056 Bytes. In this region, the delay is highly variable. For a SP_d equal 23056 Bytes, the delay value is predictable for all functions.

The fourth region (*Region 4*) relates to SP_d sizes bigger than 23056 Bytes. In this region, the delay increases monotonically with a packet size and the delay value is predictable for all functions.

To analyse the findings on the variability of the SUT's downlink delay behaviour, we involve the SUT's bearer (re)assignment behaviour presented in chapter 4. Generally, the experienced delay behaviour of the SUT always depends on the SUT's capacity. In our case, the SUT capacity directly depends on the capacity of the assigned bearer. Therefore, we put forward the hypothesis of an association of the identified delay behaviour regions with the bearer(s) assigned within the SUT. We believe that the amount of data under transport has a direct influence on the bearer assignment mechanism, and therefore our hypothesis is as follows.

In Region 1, transported packets are relatively small, and the SUT has assigned common or dedicated bearer 1 for their transport. In Region 2, transported packets are bigger and the SUT has assigned dedicated bearer 1 or dedicated bearer 2 for their transport. In Region 3, transported packets are relatively large and the SUT has assigned the dedicated bearer 2 or dedicated bearer 3 for their transport. Finally, In Region 4 transported packets are large and the SUT has assigned dedicated bearer 3 for their transport.

The hypothesis is in line with our previous conclusion that size of data transported cause the bearer (re)assignment. Because there are four bearers, we believe that for each of them there is a *threshold* of size of data transported, which causes the assignment of a next (contiguous) bearer. The threshold is associated with the (maximum) boundary capacity of the current bearer and is the (minimum) boundary capacity of the next (contiguous) bearer. We observed that for the common bearer this threshold is very low.

Whenever we started the data transmission, the dedicated bearer 1 has been assigned after a few observations.

Regarding the volume threshold, we think that when the data volume transported is lower than this threshold, then the SUT is not stimulated to switch bearers. Therefore, the probability of assigning another (contiguous) bearer is low. We believe that this situation takes place for three distinctive packet sizes: 2096 Bytes, 10480 Bytes, and 23056 Bytes, where the SUT is stimulated to use the corresponding dedicated bearers 1, 2 or 3. This situation also takes place for packets in region 1, where small data volume is transported under the assignment of dedicated bearer 1, and for packets in region 4, where large data volume is transported under the assignment of dedicated bearer 3 with the highest capacity.

When the volume of data transported by the SUT is "close" to the bearer's threshold, then the SUT may assign a contiguous bearer, we observe high randomness of bearer assignment for data transport. This situation takes place for packets from Region 2 and Region 3.

## 6.4 Bottleneck Analysis

In section 6.1, the analytical analysis of the SUT lead to the derivation of the high-level queuing model of the system as a tool for a system performance analysis. Moreover, in section 6.3, we presented the results of the SUT delay analysis. However, these analyses did not say anything about the SUT bottleneck with respect to the goodput performance parameter (defined in chapter 4). Therefore, we raise the question on the transport sub-system, which is a bottleneck, i.e. a limiting factor in achieving higher goodput. We base the bottleneck analysis, similarly to the delay analysis, on the measurement results and on the derived queuing model of the system.

*Definition:*

> *Bottleneck is lessening of throughput. It often refers to networks that are overloaded, which is caused by the inability of the hardware and transmission lines to support the traffic* [TechWeb]

The analysis of the SUT goodput behaviour is divided into two sections:

1) uplink goodput behaviour analysis (section 6.4.1) based on the conclusive combined graphs presented in section 6.2

2) downlink goodput behaviour analysis (section 6.4.2) based on the conclusive combined graphs presented in section 6.2

### 6.4.1 Uplink Behaviour

The uplink goodput behaviour's analysis is based on Figures: 6.10, 6.11 and on the knowledge obtained from the system model, and lead to the following conclusions:

1) SP_d size does not influence the SUT uplink goodput behaviour. There is an exception from this rule for SE: $SP\_u_{174}$ and $SP\_d_{174}$. For this SE, the common bearer is used, while for the SP_d sizes higher than 174 Bytes the SUT assigns the dedicated bearer 1, such that the goodput seen by the $SP\_u_{174}$ is higher (see the corresponding discussion on delay in section 6.3.1).

2) For transportation of SP_u of sizes bigger than 174 Bytes, the SUT assigns dedicated bearer 1 (see Figure 6.10).

3) The SUT uplink goodput behaviour depends on the SP_u size. The relation is initially linear and then levels off as shown in Figure 6.11. There is a limit on the maximum achievable goodput. The SUT efficiency is always lower than 100%. This indicates the system bottleneck, as we prove in coming paragraphs.

   The general form of the SUT uplink goodput ($SUT\_u_{goodput}$) as a function of SP_u packet size is given as follows:

$$SUT\_u_{goodput} = \frac{SP\_u_{size} * 8}{SP\_u_{delay}}$$

$$\text{units of measure: } [SUT\_u_{goodput}] = \left[\frac{bits}{ms}\right] = [\text{Kbps}]$$

   Where the $SP\_u_{delay}$ as a function of $SP\_u_{size}$ is given as follows (section 6.3.1):
$SP\_u_{delay} = 0{,}138 * SP\_u_{size} + 86$ [ms]

   Therefore, the $SUT\_u_{goodput}$ can be presented as follows:

$$SUT\_u_{goodput} = \frac{SP\_u_{size} * 8}{0{,}138 * SP\_u_{size} + 86} \text{ [Kbps]}$$

   The general form of the SUT uplink throughput ($SUT\_u_{throughput}$) as a function of SP_u packet size is given as follows:

$$SUT\_u_{throughput} = \frac{(SP\_u_{size} + protocolOverhead) * 8}{SP\_u_{delay}}$$

$$SUT\_u_{throughput} = \frac{(SP\_u_{size} + protocolOverhead)*8}{0{,}138*SP\_u_{size} + 86}$$

$$\text{units of measure: } [SUT\_u_{throughput}] = \left[\frac{bits}{ms}\right] = [\text{Kbps}]$$

The $SUT\_u_{efficiency}$ formula is as follows:

$$SUT\_u_{efficiency} = \frac{SUT\_u_{goodput}}{SUT\_u_{capacity}} * 100[\%]$$

$SUT\_u_{capacity}$ equals 64 Kbps (for $SP\_u_{size}$ > 174 Bytes). Therefore, the $SUT\_u_{efficiency}$ is given by the following formula (valid for $SP\_u_{size}$ > 174 Bytes):

$$SUT\_u_{efficiency} \frac{\frac{SP\_u_{size}*8}{0{,}138*SP\_u_{size}+86}}{64} * 100[\%]$$

As observed from Figure 6.11, throughput is higher than the goodput. Because throughput takes into calculation the protocol overhead, it is an expected behaviour. For small SP_u packets, the SUT goodput is very low and the hypothesis to explain it may be that for the small packets the influence of the protocol overhead is bigger than for the bigger packets. We consider that this hypothesis is wrong, because the protocol overhead is significant for all packet sizes and is even more significant for bigger packet sizes due to the packet's segmentation mechanisms. To explain why the SUT goodput is very low for small SP_u packets, we formulate the following hypothesis. For small packet sizes, the SP_u delay's propagation delay component is significantly bigger that the transmission delay component and we believe that this fact results in low SUT goodput observed for small packet sizes.

4) If we consider the SUT as a resource, then transportation of SP's results in the resource utilization, i.e. the SUT is used for some time to transport SPs. The utilization of the resource can be efficient or not. Based on the Figure 6.11, and on the knowledge obtained from the system model (Experiment 9), we can derive the relation between the SUT utilization and obtained efficiency. The definition of both performance measures is as follows.

We define the SUT utilization as:

*The percentage of time the SUT is busy with transportation of packets*

Recall the SUT efficiency definition:

> *A fraction of the SUT nominal link capacity used for the transmission of application data*

In Experiment 9 (user-unconfirmed service) packet of 524 Bytes size, are generated by the Workload Generator with different rates and then, with use of the transport service, transported through the SUT. Theoretically, for the SUT nominal capacity (64 Kbps), the SUT can support maximally $\lambda_T$ = 14 packets per second (assuming 48 Bytes of the protocol overhead per packet[61]). In this case, the SUT utilization is 100%, as the SUT is busy with transporting packets all the time. The goodput equals 57,3 Kbps and efficiency is 89,5%. In this case, because the SUT utilization is 100%, the SUT is a goodput bottleneck.

In Experiment 9 we have made measurements with arrival rate ($\lambda_{in}$), i.e., rate at which packets are offered the SUT by the Workload Generator, varying from $\lambda_{in}$ = 7 up to $\lambda_{in}$ = 28 packets per second. Transported packets depart from the SUT with a rate $\lambda_{out}$. In Figure 6.21 we present the average SUT behaviour for the case when packet arrival rate equals $\lambda_{in}$ = 28 packets per second. There are two functions in this Figure. The first function (blue) is the SUT packet arrival rate ($\lambda_{in}$) versus the packet number. The second function (red) is the corresponding function of an average SUT packet departure rate ($\lambda_{out}$) versus the packet number. Therefore, on the horizontal axis the packet number is given. The vertical axis provides the corresponding value of average SUT packet arrival (and departure) rate as a number of packets per second.

[61] 48 Bytes = 20 Bytes TCP header + 20 Bytes IP header + 8 Bytes PPP header

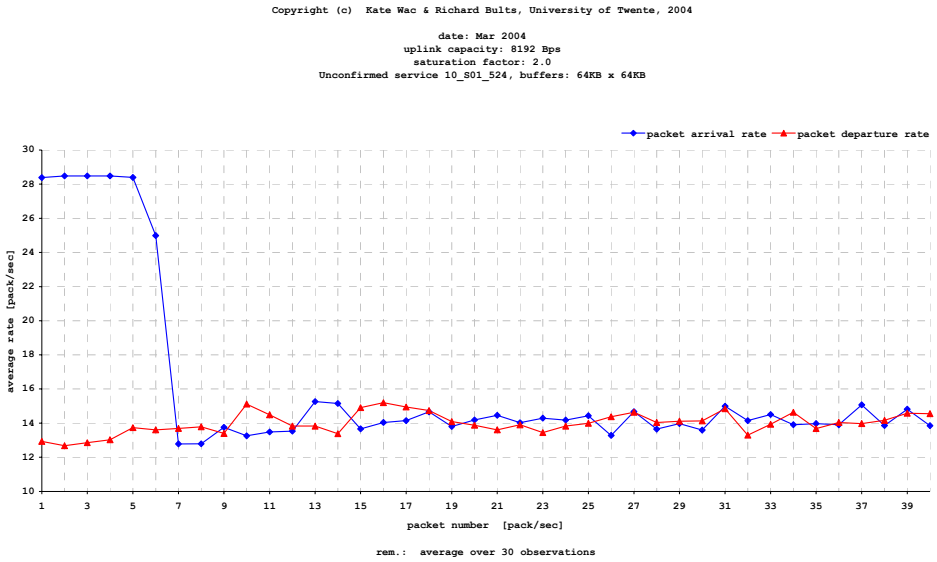



Figure 6.21 SUT's packet arrival and departure rate versus a packet number

As it can be concluded from the figure, the rate $\lambda_{in}$ = 28 packets per second can be maintained by the Workload Generator only for a few first generated packets. After around 7 packets offered by the Workload Generator, the transport service (i.e. TCP protocol) takes over and stops the Workload Generator from generating packets at such a high rate. We believe that this behaviour is caused by the fact that the transport service buffer is filled and cannot handle more packets from the Workload Generator. Packet arrival rate $\lambda_{in}$ is then lowered to the value of $\lambda_{in}$ = 14 packets per second. Regardless of how large $\lambda_{in}$ is, the corresponding rate $\lambda_{out}$ equals around $\lambda_{out}$ = 14 packets per second[62]. Summarizing the analysis of Figure 6.21, we conclude that for the case when TCP protocol takes over the control of packets generation, the packet arrival and departure rates are equal and $\lambda_{in}$ = $\lambda_{out}$ = 14 packets per second. Comparison of this case with the theoretical case described in the previous paragraph leads to the conclusion, that the SUT utilization as seen by the Workload Generator is 100%. The SUT goodput seen by the Workload Generator equals 57,3 Kbps, which means that the SUT efficiency is only 89,5% ( for the nominal capacity of the SUT is 64 Kbps). Therefore, the above indicates that the SUT is a goodput bottleneck.

[62] $\overline{\lambda}_{in}$ = 14,046 [pack/sec] and $\overline{\lambda}_{out}$ = 14,066 [pack/sec] with an accuracy of 12% and 11% for a 95% Conf. Interval

### 6.4.2 Downlink Behaviour

The downlink goodput behaviour's analysis based on Figure 6.13 lead to the following conclusions. The downlink goodput behaviour is a complex function of the SP_d size. Particularly, the SP_d size does influence the SUT's downlink goodput behaviour. There are 20 functions presented in the figure, and based on them we put forward the hypothesis on the SUT goodput behaviour. We involve the hypothesis on the SUT behavioural regions characterization, as presented in the SUT downlink delay analysis (section 6.3.2). According to us, the SUT's downlink goodput behaviour may be characterized by four different goodput behaviour regions depending on SP_d sizes.

Region 1 relates to small packets up to and including SP_d size of 2096 Bytes. In this region, the SUT has assigned common or dedicated bearer 1 for their transport. Maximum goodput obtained is around 44 Kbps for SP_d 2096 Bytes.

Region 2 relates to SP_d sizes bigger than 2096 Bytes and smaller than 10480 Bytes. In this region, the goodput behaviour is highly variable. The SUT has assigned dedicated bearer 1 or dedicated bearer 2 for packets transport. For the SP_d equal 10480 Bytes, the goodput obtained is a maximum and equals around 105 Kbps.

Region 3 relates to SP_d size bigger than 10480 Bytes and smaller than 23056 Bytes. In this region, the goodput behaviour is highly variable. Transported packets are relatively large and the SUT has assigned the dedicated bearer 2 or dedicated bearer 3 for their transport. For a SP_d equal 23056 Bytes, the goodput value equals 273 Kbps.

Region 4 relates to SP_d sizes bigger than 23056 Bytes. In this region, the goodput is constant and the delay value is predictable for all functions. Finally, In Region 4 transported packets are large and the SUT has assigned dedicated bearer 3 for their transport. The goodput value is around 300 Kbps.

## 6.5 Influence of System Parameters

In this section we investigate the influence of the *SoD* system parameters (as defined in section 4.7 and Figure 4.22) on the primary performance parameter *delay* and derived performance parameter *goodput*. For every SoD instance one system parameter of influence is investigated. The benchmark (Experiment 1) is used as a reference for all other measurements. Due to the scale of this measurement activity, and the large volume of raw data (available in Volume 2), we focus only on the SUT's uplink and downlink behaviour for the following group of SEs: SP_u size is fixed at 524 Bytes and SP_d size is varied.

We pose the following questions:

1) What is the influence of changing the intra communication system (i.e. communication system between the computer system and the UMTS terminal) from USB to Bluetooth?
   Measurement results from Experiments 1 (bm) and 5 (E5) are used to answer this question.

2) What is the influence of changing the UMTS terminal (interface) from USB-Nokia 6650 to PCMCIA-PC Card?

   *Note: We abstract from the USB and PCMCIA intra communication systems behaviour and consider these systems as a part of the UMTS terminals.*

   Measurement results from Experiments 1 (bm) and 8 (E8) are used to answer this question.

3) What is the influence of changing the computer system from notebook to iPAQ (for the Bluetooth intra communication)?
   Measurement results from Experiments 2 (E2) and 5 (E5) are used to answer this question.

4) What is the influence of including the Internet component (sub-system) in the SoD structure (as in Figure 4.22)?
   Measurement results from Experiments 1 (bm) and 7 (E7) are used to answer this question.

5) What is the influence of changing the buffer sizes of the SoD's transport service's and the application protocol?
   Measurement results from Experiments 9 (E9), 10 (E10) and 11 (E11) are used to answer this question.

The answer to the posted questions constitutes the SUT behaviour analysis. This analysis is divided into the following steps:

1) Presentation of conclusive delay and goodput graphs (section 6.5.1)

2) Analysis of the delay and goodput conclusive graphs leading to answering questions (section 6.5.2)

### 6.5.1 Conclusive Graphs

The answers to the five questions are based on the analysis of relevant measurements data. Raw data graphs from Volume 2 are used to explain the SUT's uplink and downlink behaviour. Moreover, the conclusive graphs are generated based on relevant statistical data from the measurements[63]. Figures 6.22 and 6.23, 6.24 and 6.25 present combined graphs of delay and goodput from five experiments (E1, E2, E5, E7, and E8) and will be used to answer questions 1 to 4, while Figure 6.34 (presented later) will be used to answer question 5.

*Note: The red solid line in Figures 6.22, 6.23, 6.24 and 6.25 should be interpreted as lines connecting the measurement points for a benchmark experiment. For the other experiments, only the measurement points are presented.*

Figure 6.22 presents the SUT's uplink average delay behaviour as a function of SP_d size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding average uplink delay in milliseconds.

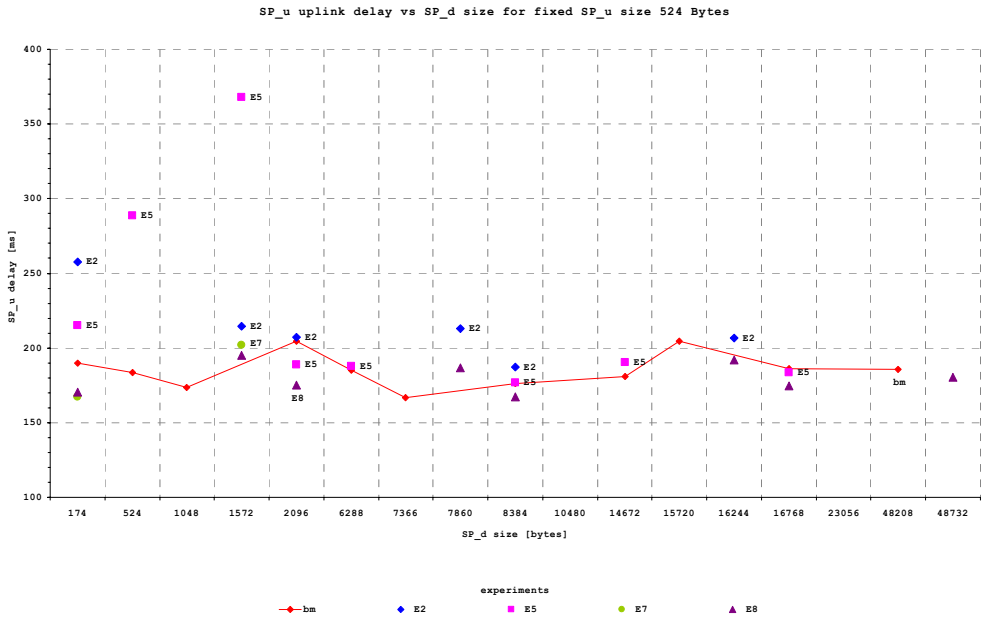



Figure 6.22 SUT uplink delay behaviour versus SP_d size

[63] Approach is similar to section 6.2.2, therefore we omitted a description of how do we generate the conclusive graphs.

Figure 6.23 presents the SUT's uplink average goodput behaviour as a function of SP_d size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding average uplink goodput in Kbps. Two red horizontal lines indicate the SUT capacity boundaries for the common (16 Kbps) and dedicated bearer (64 Kbps) in the uplink direction.

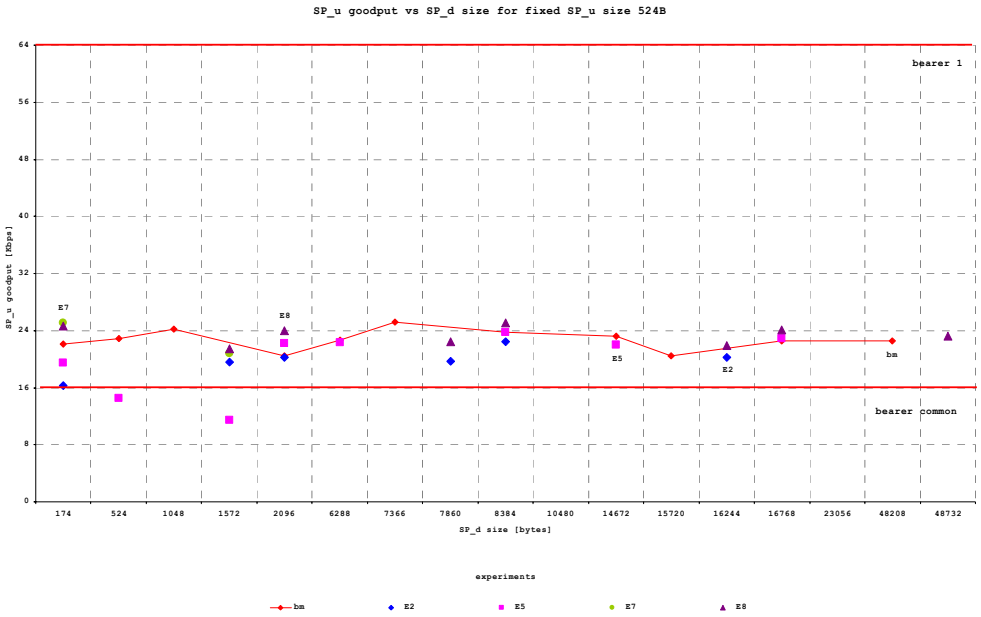



Figure 6.23 SUT uplink goodput behaviour versus SP_d size

Analogously to Figure 6.22, Figure 6.24 presents the SUT's downlink average delay behaviour as a function of SP_d size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding average downlink delay in milliseconds. Three red dotted vertical lines indicate the SUT behaviour regions' boundaries in the downlink direction.

Analogously to Figure 6.23, Figure 6.25 presents the SUT's downlink average goodput behaviour as a function of SP_d size. The horizontal axis provides the values of the SP_d size (in Bytes). The vertical axis presents the corresponding average downlink goodput in Kbps. Four red horizontal lines indicate the SUT capacity boundaries for the common (16 Kbps) and dedicated bearer (64 Kbps, 128 Kbps, 384 Kbps) in the downlink direction. Three red dotted vertical lines indicate the SUT behaviour regions' boundaries in the downlink direction.

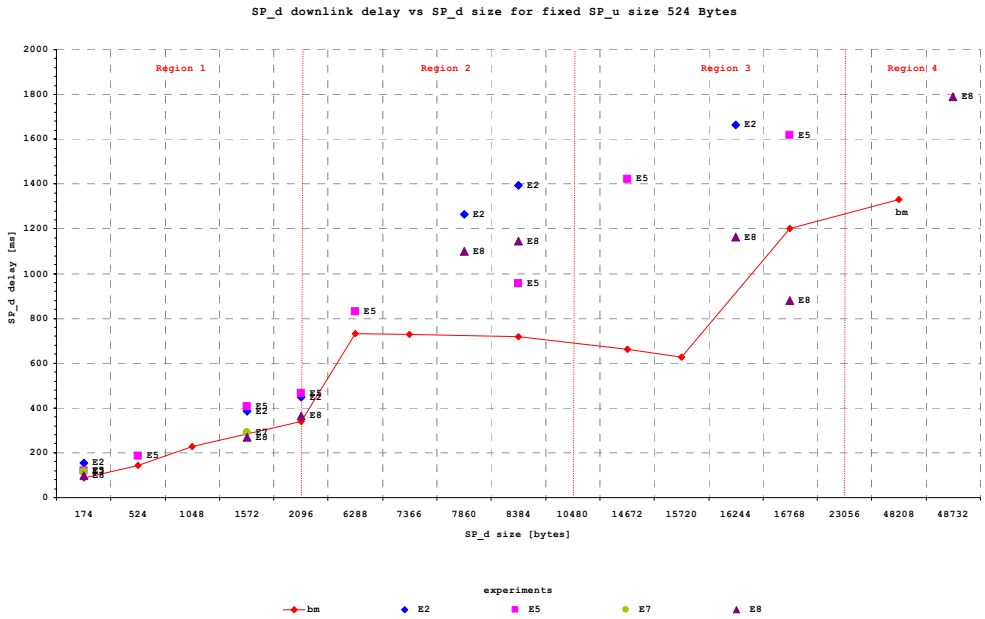



Figure 6.24 SUT downlink delay behaviour versus SP_d size

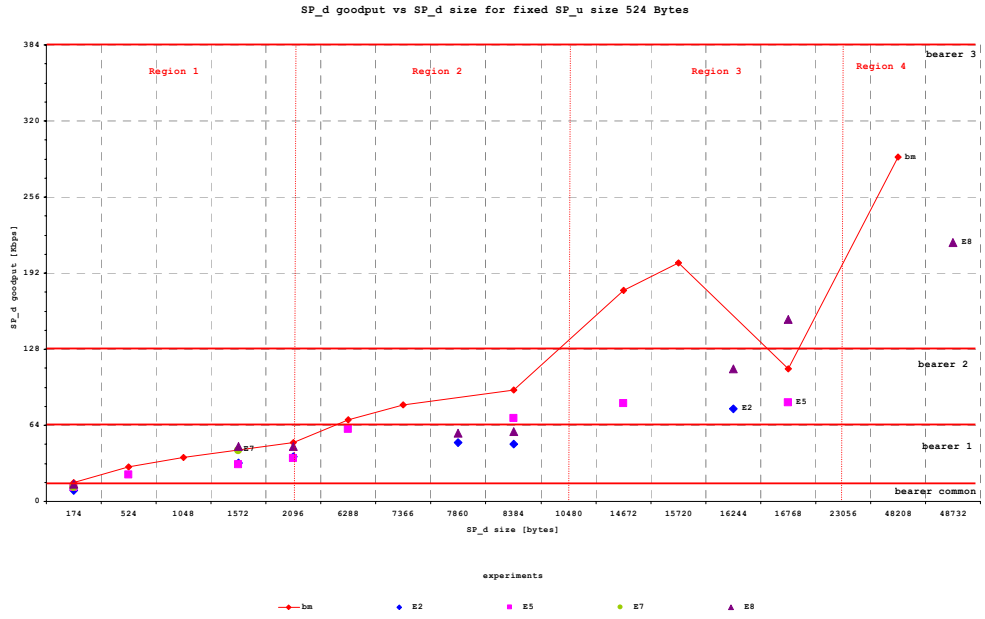



Figure 6.25 SUT downlink goodput behaviour versus SP_d size

### 6.5.2 System Behaviour

In this section, we provide the results of the analysis of the SUT (uplink and downlink) behaviour as observed in different experiments, and presented in conclusive graphs. Unfortunately, the conclusive figures do not provide very clear understanding on the SUT behaviour; they are not complete. Hence, we answer posted questions based on our understanding of the SUT behaviour in different experiments with comparison to the benchmark experiment.

We answer each posted question in two paragraphs. In the first paragraph, we analyze and provide the answer regarding the SUT uplink behaviour. Consequently, the second paragraph provides the answer regarding the SUT downlink behaviour.

> *Question 1: What is the influence of changing the intra communication system (i.e. communication system between the computer system and the UMTS terminal) from USB to Bluetooth?*

Measurement results from Experiments 1 (bm) and 5 (E5), as provided in Figures 6.22 to 6.25, are used to answer this question. In Experiment 1, the USB intra communication system has been used, while in Experiment 5 communication system based on Bluetooth. The red line in both Figures show the SUT behaviour for a USB intra communication system (E1), and the magenta points (squares) show the behaviour of the Bluetooth alternative (E5).

Regarding the SUT uplink delay behaviour, according to the benchmark Experiment 1, the expected average uplink delay is around 190.0 milliseconds (Figure 6.26, SE: SP_$u_{524}$, SP_$d_{174}$). From Figure 6.22 we conclude that the difference in the SUT behaviour for USB and Bluetooth seems to be significant for SP_d sizes of 174, 524 and 1572 Bytes. Moreover, we conclude, that there is no significant difference for other sizes in the delay behaviour experienced in both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4).

We present the raw data for Experiment 5 in Figures 6.27 - 6.30 to analyse the behaviour of the SUT for Bluetooth for SP_d sizes of 174, 524 and 1572 (and additionally 2096) Bytes. It seems to be that this significant difference in behaviour of the SUT does not come from the fact, that different intra communication technology has been used, but from the fact that there were the SUT resource problems while performing E5's measurements. Analysis is as follows.

Figure 6.27 (SE: SP_$u_{524}$ SP_$d_{174}$) shows changes in the SUT behaviour at around 50 and around 150 observations, such that the experienced delay between the observation 50 and 150 is higher than expected. In Figure 6.28 (SE: SP_$u_{524}$, SP_$d_{524}$) it can be observed that this higher delay is experienced almost in the whole sample; only between around 60 and around 230 observation the delay is as expected. Furthermore, Figure 6.29 (SP_$u_{524}$, SP_$d_{1572}$) shows the uplink operating with a high delay (360.0 ms) for the whole sample i.e. 500 observations. Then Figure 6.30 (SE: SP_$u_{524}$, SP_$d_{2096}$) shows the uplink

alternating between the high and lower (i.e. expected) delay. Very important fact is that all four measurement samples (as provided in Figures 6.27-6.30) were taken one after the other in the same timeframe of measurements. The SUT downlink is not changing, which leads to the conclusion that no time synchronisation event could cause the uplink behaviour (see section 5.5 for explanation). The cause for changing the SUT behaviour is not clear, hence it's fair to say that the SUT encountered resources problems due to the background load[64], and these problems were experienced as a higher delay.

Summarizing our analysis for the SUT behaviour in E5 for small packet size, we conclude that the measured behaviour, as presented in Figures 6.27 – 6.29 is not representative. Consequently, we do not consider this SUT behaviour, while answering posted question. Our question will be answered only based on data for SP_d equal to and higher than 2096 Bytes.

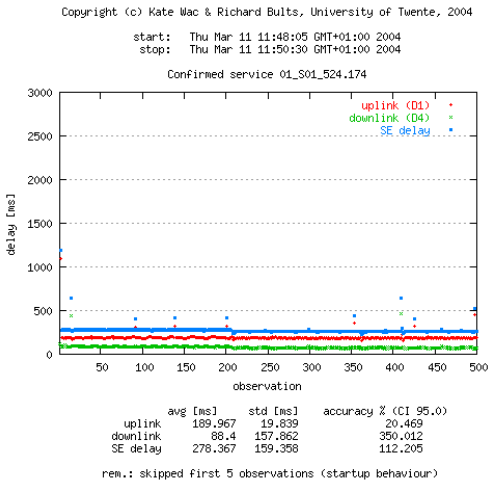



Figure 6.26 benchmark SP_$u_{524}$,SP_$d_{174}$

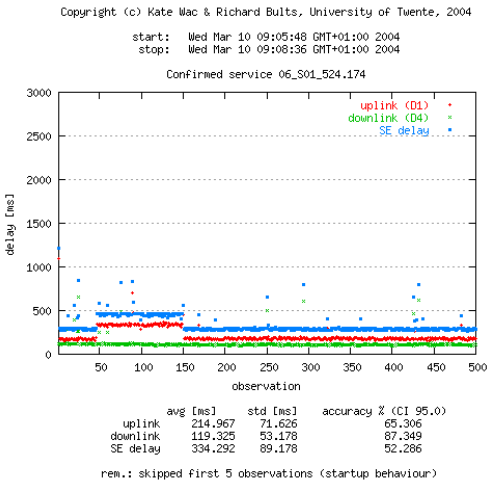



Figure 6.27 E5 SP_$u_{524}$,SP_$d_{174}$

[64] The SUT was in commercial operation during this measurement activity. Background load could be caused by other service users.

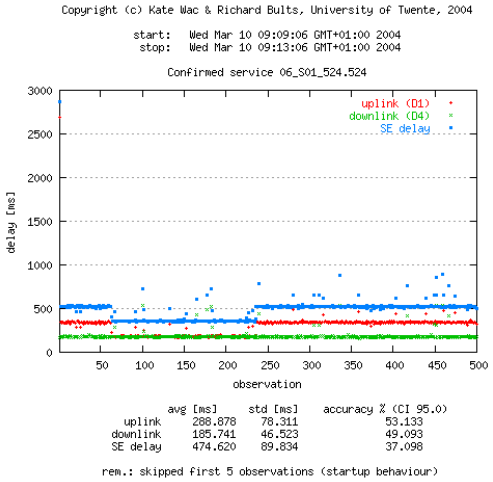



Figure 6.28 E5 SP_u$_{524}$,SP_d$_{524}$

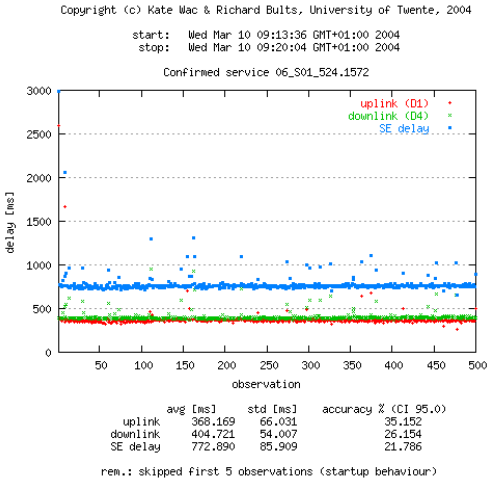



Figure 6.29 E5 SP_u$_{524}$,SP_d$_{1572}$

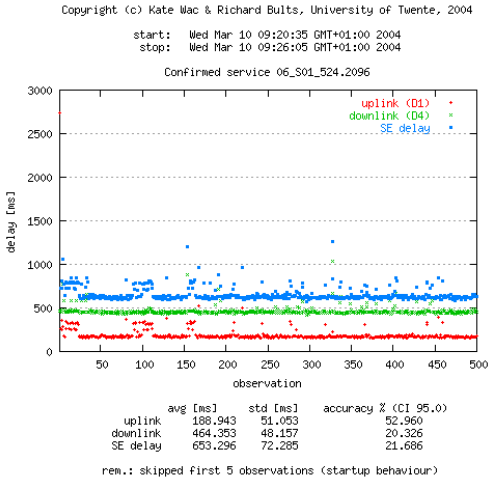



Figure 6.30 E5 SP_u$_{524}$,SP_d$_{2096}$

*Answer 1.1: The SUT uplink behaviour is similar when using the USB and Bluetooth intra communication system.*

The downlink behaviour's analysis, conducted in section 6.3 for Experiment 1 (benchmark), lead to the conclusion that the SUT downlink delay behaviour has 4 remarkable regions. Therefore, now we conduct the analysis on the difference in the SUT downlink behaviour for USB and Bluetooth based on the SUT delay behaviour regions, as presented in Figure 6.24.

In the Region 1 (SP_d ≤ 2096 Bytes) in Experiments 1 and 5, the SUT has assigned common or a dedicated bearer 1 for packets transport. In this region, the delay value is very much similar for both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4). Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 1. This is consistent with the Region 1 description given in section 6.3.

In the Region 2 (2096 < SP_d < 10480 Bytes) in Experiments 1 and 5, the SUT has assigned dedicated bearer 1 or a dedicated bearer 2 for packets transport. To prove this, we provide behaviour analysis for the SP_d size 8384 Bytes. The benchmark experiment shows an average delay of 717.0 ms and for E5 experiment an average delay of 957.0 ms. Goodput calculations indicate that both measurement points are related to the SUT dedicated bearer 2 (benchmark goodput is around 94.0 Kbps, and E5 around 70.0 Kbps). The assigned bearer is the same, but there is a difference in the SUT delay behaviour observed in both experiments; the difference increases with the transported packets size. Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 2. This is consistent with the Region 2 description given in section 6.3.

In the Region 3 (10480 < SP_d < 23056 Bytes) in Experiments 1 and 5, for packets transport the SUT has assigned dedicated bearer 2 or, only in case of Experiment 1, a dedicated bearer 3. To prove this, we provide behaviour analysis for the SP_d size 14672 Bytes. For the benchmark experiment, the average delay is 661.0 ms and goodput is 178.0 Kbps, and for the Experiment 5 the corresponding numbers are: 1423.0 ms and around 83.0 Kbps. The goodput numbers indicate that the benchmark was using the SUT dedicated bearer 3, while the E5 Experiment was still using the SUT dedicated bearer 2. However, the E5 measurement data for SP_d size 16768 Bytes (1617 ms and around 83.0 Kbps) indicates the use of the same SUT bearer as for SP_d size 14672 Bytes. We interpret these results as if the Bluetooth intra communication system's maximum goodput rate is limited to around 83.0 Kbps[65]. Generally, in Experiment 1, there is a randomness of the SUT bearers' assignment in Region 3. This is consistent with the Region 3 description given in section 6.3. There is no randomness of the SUT bearers' assignment in Region 3 in case of Experiment 5. This is different from the expected SUT behaviour, different from the Region 3 description given in section 6.3.

Finally, in the Region 4 (SP_d > 23056 Bytes) in Experiment 1 the SUT has assigned dedicated bearer 3 for packets transport. Unfortunately, we do not have any measurements in this region for Experiment 5, but we suppose that due to the Bluetooth capacity limitation, the SUT would have been assigned at most the dedicated bearer 2 for packet transport. Generally, in Experiment 1, there is SUT dedicated bearer 3 assigned in Region 4. This is consistent with the Region 4 description given in section 6.3. We believe that would be the SUT dedicated bearer 2 assigned in Region 4 in case of Experiment 5. This is different from the expected SUT behaviour, different from the Region 4 description given in section 6.3.

[65] On our request, Vodafone informed us that the maximum capacity of the Nokia 6650 Bluetooth interface is limited to 115 Kbps.

Based on the analysis for the four different delay behaviour regions, we conclude that there is a significant difference between USB and Bluetooth for packets from Regions 3 and 4 (i.e. for all SP_d sizes larger than 10480 Bytes). The delay behaviour observed in case of Bluetooth is always higher than delay observed for USB. The difference in the SUT delay behaviour observed in both experiments increases with the packets size. Due to the Bluetooth capacity's limitation, the SUT would never assign dedicated bearer 3 for packets transport.

*Answer 1.2: The influence of changing the intra communication system from USB to Bluetooth is significant for the SUT downlink delay behaviour when using packet sizes larger then 10480 Bytes. In case of the Bluetooth technology, observed delays are always higher than in case of USB. Bluetooth has a nominal capacity limitation of 115 Kbps and the maximum obtained goodput is 83 Kbps. Hence, Bluetooth is a goodput bottleneck.*

*Question 2: What is the influence of changing the UMTS terminal (interface) from USB-Nokia 6650 to PCMCIA-PC Card?*

Measurement results from Experiment 1 (bm) and 8 (E8), as given in Figures 6.22 to 6.25, are used to answer this question. In Experiment 1, the USB-Nokia 6650 UMTS terminal (interface) has been used, while in Experiment 8 – a PCMCIA-PC Card. The red line in both Figures show the SUT behaviour for the USB-Nokia 6650 UMTS terminal (E1), and the purple points (triangles) show the behaviour of the PCMCIA-PC Card terminal alternative (E8).

Regarding the SUT uplink delay behaviour, according to the benchmark Experiment 1, the expected average uplink delay is around 190.0 milliseconds (Figure 6.26, SE: $SP\_u_{524}$, $SP\_d_{174}$). From Figure 6.21 we conclude that there is no significant difference in the delay behaviour experienced in both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4).

*Answer 2.1: The SUT uplink behaviour is similar when changing the UMTS terminal (interface) from USB-Nokia 6650 to PCMCIA-PC Card.*

The downlink behaviour's analysis, conducted in section 6.3 for Experiment 1 (benchmark), lead to the conclusion that the SUT downlink delay behaviour has four remarkable regions. Therefore, now we conduct the analysis on the difference in the SUT downlink behaviour for Nokia 6650 terminal and the PC Card based on the SUT delay behaviour regions, as presented in Figure 6.24.

In the Region 1 (SP_d ≤ 2096 Bytes) in Experiments 1 and 8, the SUT has assigned common or a dedicated bearer 1 for packets transport. In this region, the delay value is very much similar for both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4). Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 1. This is consistent with the Region 1 description given in section 6.3.

In the Region 2 (2096 < SP_d < 10480 Bytes) in Experiments 1 and 8, the SUT has assigned dedicated bearer 1 or a dedicated bearer 2 for packets transport. To prove this, we provide behaviour analysis for the SP_d size 8384 Bytes. The benchmark experiment shows an average delay of 717.0 ms and for E8 Experiment the average delay is 1147.0 ms. Goodput calculations indicate that the benchmark measurement point is related to the SUT dedicated bearer 2 (goodput around 94.0 Kbps). The E8 measurement point is however related to the SUT dedicated bearer 1 (goodput around 59.0 Kbps). The difference between the two measurement points is caused by the fact that the SUT has not assigned dedicated bearer 1 in case of E8. To prove it we provide raw data figure for SE: SP_$u_{524}$, SP_$d_{7860}$ (Figure 6.31), from which we conclude that no dedicated bearer 2 has been assigned. From raw data figure for SE: SP_$u_{524}$, SP_ $d_{7860}$ (Figure 6.32) we conclude that the SUT has assigned dedicated bearer 2 only after around 450 observations. Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 2. This is consistent with the Region 2 description given in section 6.3.

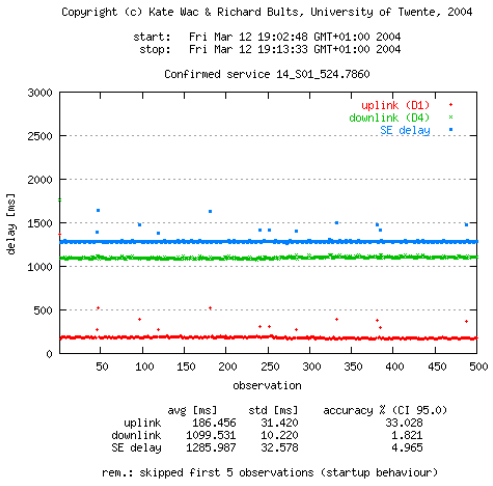



Figure 6.31 E8 SP_$u_{524}$,SP_$d_{7860}$

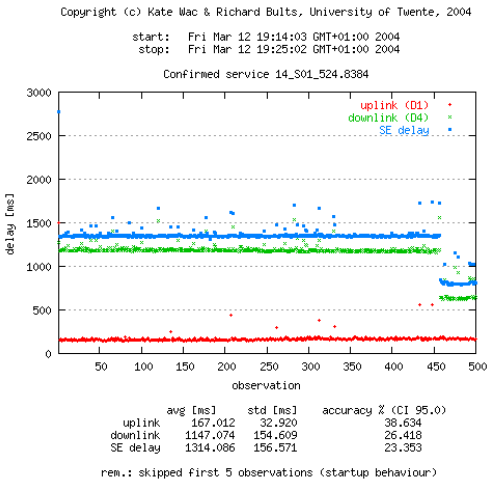



Figure 6.32 E8 SP_$u_{524}$,SP_$d_{8384}$

In the Region 3 (10480 < SP_d < 23056 Bytes) in Experiments 1 and 8, the SUT has assigned dedicated bearer 2 or a dedicated bearer 3 for packets transport. To prove this we provide behaviour analysis for the SP_d size 16768 Bytes. For the benchmark experiment the average delay and goodput are 1200.0 ms and around 112.0 Kbps and for the E8 Experiment respectively 878.0 ms and around 153.0 Kbps. The goodput numbers indicate that the benchmark was using the SUT dedicated bearer 2, while the E8 Experiment has been assigned with the SUT dedicated bearer 3 after about 250 observatories (see Figure 6.33, SE: SP_$u_{524}$, SP_ $d_{16768}$,).

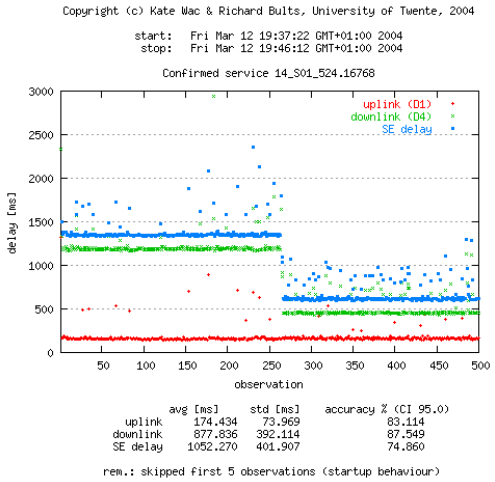



Figure 6.33 E8 SP_$u_{524}$,SP_$d_{16768}$

Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 3. This is consistent with the Region 3 description given in section 6.3.

Finally, in the Region 4 (SP_d > 23056 Bytes) in Experiment 1 and 8 the SUT has assigned dedicated bearer 3 for packets transport. We conclude that in this region, the SUT delay behaviour is similar for both experiments. Generally, in both experiments, there is a SUT dedicated bearer 3 assigned in Region 4. This is consistent with the Region 4 description given in section 6.3.

Based on the analysis for the 4 different delay behaviour regions, we conclude that there is no difference between observed SUT delay in case of Nokia 6650 and PC Card UMTS terminal. We put forward the hypothesis that both technologies behave according to the region's description. In Region 1, in both experiments the SUT common or dedicated bearer 1 has been assigned. In Regions: 2 and 3 there is a randomness in the SUT bearer assignment in both experiments. In Region 4, both has been assigned with the dedicated bearer 3. Hence, we conclude that the SUT delay behaviour is similar for both terminals.

*Answer 2.2: The SUT downlink behaviour is similar when changing the UMTS terminal (interface) from USB-Nokia 6650 to PCMCIA-PC Card.*

*Question 3: What is the influence of changing the computer system from notebook to iPAQ (for the Bluetooth intra communication)?*

Measurement results from Experiment 2 (E2) and 5 (E5), as given in Figures 6.22 to 6.25, are used to answer this question. In Experiment 2, the notebook computer system has been used, while in Experiment 5 – an iPAQ. In both experiments, the Bluetooth intra communication system has been used. The blue points (diamonds) in both Figures show the SUT behaviour for the notebook computer system (E2), and the magenta points (squares) show the behaviour of the iPAQ alternative (E5).

Regarding the SUT uplink delay behaviour, based on the analysis of results for E5 conducted while answering Question 1, we discarded the measurement results for SP_d smaller than 2096 Bytes. Therefore, we compare the SUT behaviour for E2 and E5 only for SP_d equal to and bigger than 2096 Bytes. From Figure 6.22 we conclude that there is no significant difference in the delay behaviour experienced in both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4).

*Answer 3.1: The SUT uplink behaviour is similar when changing the computer system from notebook to iPAQ (for the Bluetooth intra communication).*

The downlink behaviour's analysis, conducted in section 6.3 for Experiment 1 (benchmark), lead to the conclusion that the SUT downlink delay behaviour has four remarkable regions. Therefore, now we conduct the analysis on the difference in the SUT downlink behaviour for notebook and iPAQ computer systems (and Bluetooth intra communication), based on the SUT delay behaviour regions, as presented in Figure 6.24.

In the Region 1 (SP_d $\leq$ 2096 Bytes) in Experiments 2 and 5, the SUT has assigned common or a dedicated bearer 1 for a packets transport. In this region, the delay value is very much similar for both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4). Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 1. This is consistent with the Region 1 description given in section 6.3.

In the Region 2 (2096 < SP_d < 10480 Bytes) in Experiments 2 and 5, the SUT has assigned dedicated bearer 1 or a dedicated bearer 2 for packets transport. To prove this, we provide behaviour analysis for the SP_d size 8384 Bytes. In the E2 Experiment we observed an average delay of 1395.0 ms and for E5 Experiment the average delay of 957.0 ms. Goodput calculations indicate that the E2 measurement point is related to the SUT dedicated bearer 1 (goodput around 48.0 Kbps). The E5 measurement point is however related to the SUT dedicated bearer 2 (goodput around 70.0 Kbps). The difference between the two measurement points is caused by the fact that SUT has not assigned dedicated bearer 2 in case of E2. Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 2. This is consistent with the Region 2 description given in section 6.3.

In the Region 3 (10480 < SP_d < 23056 Bytes) in Experiments 2 and 5, the SUT has assigned dedicated bearer 2 for packets transport. To prove this, we provide behaviour analysis for the SP_d size 16244 Bytes for E2 and 16768 Bytes for E5. For Experiment E2, the average delay is 1663 ms and goodput is 76.3 Kbps, and for the Experiment 5 the corresponding numbers are: 1617 ms and around 81 Kbps. The goodput numbers indicate that both E2 and E5 were assigned with the SUT dedicated bearer 2. As in question 1, we interpret these results as if the Bluetooth intra communication system's maximum goodput rate is limited. Generally, there is no randomness of the SUT bearers' assignment in Region 3 in case of Experiments 2 and 5. The SUT delay behaviour observed in both experiments in similar. The fact that behaviour is different from the expected SUT behaviour, i.e., different from the Region 3 description given in section 6.3, can be explained by the Bluetooth's capacity limitation.

Finally, we do not have data for any of Experiments in the Region 4 (SP_d > 23056 Bytes). We suppose that due to the Bluetooth capacity limitation, the SUT would have been assigned at most the dedicated bearer 2 for packet transport in both experiments. Generally, there we believe that there would be no randomness of the SUT bearers' assignment in Region 4 in case of Experiments 2 and 5. The SUT delay behaviour observed in both experiments would be similar. The fact that behaviour would be different from the expected SUT behaviour, i.e., different from the Region 4 description given in section 6.3, could be again explained by the Bluetooth's capacity limitation.

Based on the analysis for the four different delay behaviour regions, we conclude that there is no difference in observed delay behaviour between notebook and iPAQ computer systems. For both computer systems, due to the Bluetooth intra communication capacity's limitation, the SUT would never assign dedicated bearer 3 for packets transport.

*Answer 3.2: The SUT downlink behaviour is similar when changing the computer system from notebook to iPAQ (for the Bluetooth intra communication). In both cases, the Bluetooth is a goodput bottleneck.*

*Question 4: What is the influence of including the Internet component (sub-system) in the SoD structure (as in Figure 4.22)?*

Measurement results from Experiment 1 (E1) and 7 (E7), as given in Figures 6.22 to 6.25, are used to answer this question. In Experiment 1, the Internet component has been excluded from the SoD structure, while in Experiment 7 – it has been included. The red line in both Figures show the SUT behaviour for the case when Internet is excluded (E1), and the green points (circles) show the behaviour of the SUT including Internet, alternative (E7).

*Note: Because in the Experiment 7 there are measurements conducted only for two different SP_d sizes (section 5.2.1) we consider these measurements results only as a raw indication of the SUT behaviour.*

Regarding the SUT uplink delay behaviour, according to the benchmark Experiment 1, the expected average uplink delay is around 190.0 milliseconds (Figure 6.27, SE: $SP\_u_{524}$, $SP\_d_{174}$). From Figure 6.22 we conclude that there is no significant difference in the delay behaviour experienced in both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4).

*Answer 4.1: There is no difference in the SUT uplink behaviour between case when the Internet sub-system is introduced in the SoD configuration and when it is not (for SP_d ≤ 1572 Bytes).*

The downlink behaviour's analysis, conducted in section 6.3 for Experiment 1 (benchmark), lead to the conclusion that the SUT downlink delay behaviour has four remarkable regions. Therefore, now we conduct the analysis on the difference in the SUT downlink behaviour for SoD with and without the Internet component, based on the SUT delay behaviour regions, as presented in Figure 6.24. Unfortunately, for Experiment 7, we have only data for packet sizes from the delay behaviour Region 1.

In the Region 1 (SP_d ≤ 2096 Bytes) in Experiments 1 and 7, the SUT has assigned common or a dedicated bearer 1 for a packets transport. In this region, the delay value is very much similar for both experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4). Generally, in both experiments, there is a randomness of the SUT bearers' assignment in Region 1. This is consistent with the Region 1 description given in section 6.3.

*Answer 4.2: There is no difference in the SUT downlink behaviour between case when the Internet sub-system is introduced in the SoD configuration and when it is not (for SP_d ≤ 1572 Bytes).*

> *Question 5: What is the influence on the uplink behaviour when changing the buffer sizes of the SoD's transport service's and the application protocol?*

Measurement results from Experiment 9 (E9), 10 (E10) and 11 (E11) are used to answer this question. In Experiment 9, the buffer sizes of the SoD transport service and application protocol are equally setup at 64 KBytes. In Experiment 10, the SoD transport service's buffer size equals 64 KBytes and the application protocol buffer equals 32 KBytes. In Experiment 11, the buffer sizes of the SoD transport service and application protocol are equally setup at 32 KBytes. These experiments specified a user unconfirmed test for the uplink of the SUT with fixed SP_u size of 524 Bytes. In the analysis towards the answer to this question, we only focus on a link saturation factor of 2.0. From an application protocol designer's perspective, it is important to understand the behaviour of the SUT for user unconfirmed application protocols that saturate the SUT's uplink.

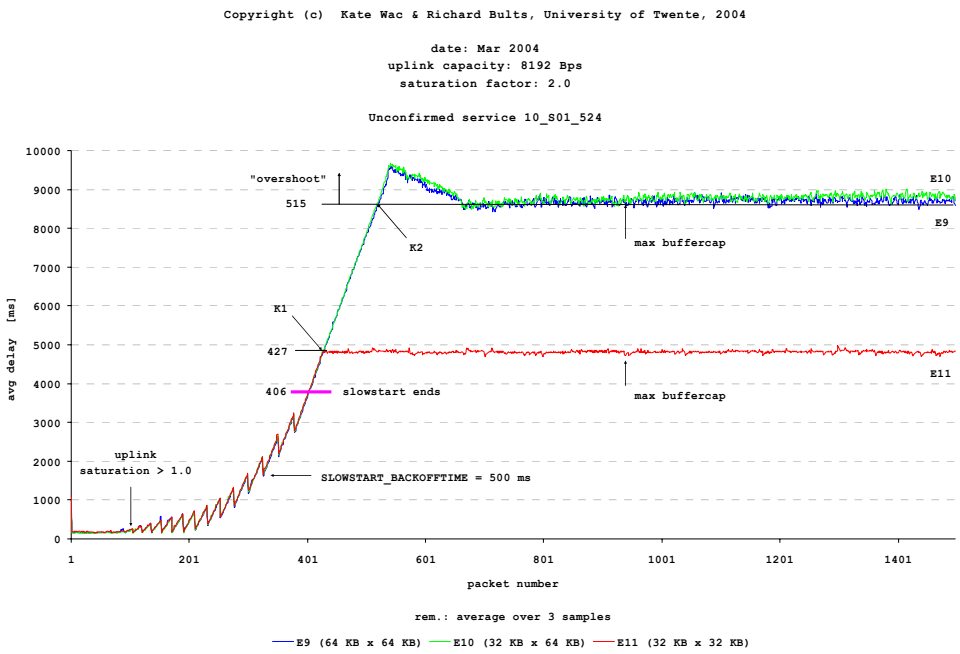



Figure 6.34 SUT's average uplink delay in E9, E10, E11 ($SP\_u_{524}$ and SAT 2.0)

Figure 6.34 presents a combined graph from experiments E9, E10 and E11. Figure presents the SUT's average uplink delay behaviour as a function of the packet number in the measurement sample. Hence, the horizontal axis provides the packet number in the sample. The vertical axis represents the SUT's average uplink delay behaviour for a SP_u

size of 524 Bytes. The red line denotes results from Experiment E11, the green line denotes experiment E10 results and the blue line denotes results obtained from Experiment E9.

As we indicate in the figure, there are two important phases identified for all experiments: *slowstart* and saturation phase. We consider the uplink delay behaviour during the *slowstart* phase to be irrelevant for the analysis of the SUT's behaviour. The *slowstart* phase functions as an initialisation phase for the actual performance measurements.

In all experiments, the *slowstart* phase starts at packet number 1 and ends at number 406. We provide the proof for this number in the following note.

*Note: In section 5.5.5, we introduced the formulae to calculate the "NoOfSlowstartPackets". For Experiments 9, 10 and 11, the SUT uplink capacity ($up\_link\_capacity$) is 64 Kbps (8192 Bps), the saturation factor is 2.0 and the SP_u packet size is 524 Bytes. The protocol overhead is 60 Bytes (per packet).*

$$\max PacketRate = \left\lfloor \frac{up\_link\_capacity * saturation\_factor}{packetSize + protocolOverhead} \right\rfloor$$

$$= \left\lfloor \frac{8192 * 2.0}{524 + 60} \right\rfloor = 28.0$$

$$NoOfSlowstartPackets = \sum_{i=1}^{\max PacketRate} i = \frac{\max PacketRate * (\max PacketRate + 1)}{2}$$

$$NoOfSlowstartPackets = \frac{28 * 29}{2} = 406$$

In all experiments, the workload generator causes the "saw tooth" behaviour shown in all the experiments. This is because after generating $NoOfSlowstartPackets$, workload generator stops for *SLOWSTART_BACKOFFTIME* ms (experimental value of 500ms) to keep the probability of UMTS terminal output buffer overflow as low as possible during the *slowstart* phase (see section 5.5.5 for details).

In all experiments, the saturation phase starts after packet number 406. From this point onwards, the SUT's uplink is offered packets with twice its link packet rate (because saturation factor equals 2.0). Figure 6.34 shows that after this point the behaviour of the SoD is influenced significantly by the size of the transport service buffer and not by the size of the application protocol buffer. This means that the analysis for E9 and E10 are similar and different from analysis of E11.

After the slowstart phase, the average delay observed in E9 and E10 increases linearly up to *K2* (packet number 515). After *K2* there is a buffer overshoot (overflow) phase observed. This resulted in observed additional delay, which may indicate the time for TCP to react, i.e., to activate its flow control mechanism. After the TCP is active, it does

not allow application layer to fill the transport service's buffer completely. TCP regulates the speed, at which an application layer is allowed to send data; such that the transport service buffer does not overflow. In our analysis, we focus on the SUT behaviour for observations of number bigger than 600, as depicted by the black line that crosses point K2. From this point, the average delay remains constant for the rest of the observations. This is because the TCP's mechanism results in stopping the application from offering data to TCP service when the transport service buffer is (almost) full.

Each application packet is accepted for transportation only when there is space in the underlying TCP transport service send buffer. That means that a new packet is accepted from application layer only if other packet has been successfully transported throughout the SUT. Moreover, each new packet accepted sees the transport service buffer (almost) full, and has to wait until its transport starts. The observed packet transport delay contains the delay when packet is stored in buffer before being transported throughout the SUT plus the actual packet transport delay in the SUT. The buffer delay component is dominant in the observed packet transport delay.

In experiment E11 we have observed similar SUT behaviour like in E9 and E10. After the slowstart phase, the average delay for E11 increases linearly up to *K1* (packet number 427). After K1 the average delay remains constant for the rest of the observations. The transport service is implemented by the TCP protocol, which uses a flow control service[66] offering a *speed regulating* effect to an application layer. The effect of the speed regulation is then the observed adjusted transport delay for the rest of the observations. As in case of E9 and E10, the observed packet transport delay in E11 contain the delay packet is stored in buffer before being transported throughout the SUT plus the actual packet transport delay in the SUT. Again, the buffer delay component is a dominant in the observed packet transport delay.

To explain why the change of application buffer size does not have an influence on the observed delay (in E9 and E10), we put forward the following hypothesis. The TCP flow control mechanism does not trigger the application layer, i.e., application protocol buffer, to offer data for transport unless 1) the transport service and the application protocol buffers are empty or 2) transport service buffer is empty or there is a space in this buffer to accommodate application packets for transportation. Therefore, the situation where application buffer is not filled, but application is stopped from sending due to the transport service buffer overflow, is very likely to occur. The feedback given by the TCP layer to the application protocol is based completely on the state of the transport service buffer, application protocol buffer does not play much role in this.

For E9 and E10 it can be observed that, since packets are generated at a higher rate (saturating the SUT), the transport service buffer is always full. Thus increasing the transport buffer from 32 KBytes to 64 KBytes will increase the observed packet delay by almost the factor of two (this is again the evidence that the bottleneck is the SUT). It is because the transport service's buffer delay component will increase by almost the factor

[66] TCP uses a flow control service to its service users to eliminate the possibility of the sender overflowing the receiver's buffer. Flow control is a speed matching service, matching the rate at with the sender is sending to the rate at which the receiving service user is reading [Kuro2001].

of two. The transport service's buffer delay component is a dominant in the observed packet transport delay.

We conclude that increasing the transport buffer would result in the increase the packet delay. Moreover, we conclude that in our experiments, the application protocol's buffer size change does not influence the observed delay. However, the influence of buffer size on the observed goodput has not been yet studied. Therefore, in this paragraph we focus on the influence of the transport service's buffer size on the observed goodput. Based on the conclusion of lack of influence of application protocol's buffer size on observed delay, we presume also the lack of influence of application protocol's buffer size on observed goodput and therefore we do not consider the application buffer size.

In the following paragraphs, we derive the goodput behaviour from the observed delay behaviour in E9, E10 and E11. However, for all experiments, we focus only on the part of the results obtained when the TCP flow control mechanism has been active and the observed average delay remains constant (behaviour observed after the K1 and K2 points).

> *Note: It is obvious that since SUT is saturated, its throughput and goodput characteristics are limited by the SUT capacity, independent of the buffer size. The goal of this section is to prove it.*

In section 6.1 we described the relation between the system response time (r), service time ($\bar{s}$) and waiting time (w):

$$r = w + \bar{s}$$

We assume the service time as constant (C) in this analysis, therefore we rewrite the formula above into:

$$r = w + C$$

The system's response time is a function of waiting time. As we concluded in this section, the system's waiting time is directly associated with the transport service buffer size; large buffer sizes result in long waiting times (compare Figure 6.34 lines E10 and E11). Hence, the system's response time is a function of the transport service buffer size ($Buff_{size}$):

$$r = f(Buff_{size}) + C$$

units of measure: [r] = [ms]

Utilizing Little's formula [Klei1976], we formulate the goodput of the SoD as:

$$SoD_{goodput} = \frac{SP\_u_{length}}{SP\_u_{delay}} * N$$

units of measure: $[SUT\_u_{goodput}] = [b] / [ms] = [Kbps]$

*N* denotes the total number of packets in the SoD, sum of number of packets being transported (*n*) plus these waiting to transport. It is important to notice that the number of packets waiting (in a buffer) for a transport is proportional to the transport service buffer's size. Therefore:

$$N = n + f(Buff_{size})$$

units of measure: [N] = [ dimensionless ]

The average system's response time for a particular SP_u size equals the SP_u transmission delay. This leads to the formula:

$$SoD_{goodput} = \frac{SP\_u_{length}}{\bar{r}} * N = \frac{SP\_u_{length}}{f(Buff_{size}) + C} * N$$

$$SoD_{goodput} = \frac{SP\_u_{length}}{f(Buff_{size}) + C} * \left[ n + f(Buff_{size}) \right]$$

units of measure: [SUT_$u_{goodput}$] = [b] / [ms] = [Kbps]

The SoD goodput seems to be a function of the transport service's buffer size. It is very important to notice that SoD goodput is at the same time an inverse function of the transport service buffer's size and proportional to N - the total number of the packets in the SoD. Hence, the SoD goodput is not really influenced by the transport service's buffer size change, as we prove in the following paragraphs.

We are also interested in the observed efficiency of the SoD. The division of the SoD goodput by the SoD (maximum) capacity (i.e. 64 Kbps for the uplink's dedicated bearer 1) derives the efficiency of the SoD. The formula below supports this analysis:

$$SoD_{efficiency} = \frac{SoD_{goodput}}{SoD_{capacity}} * 100$$

$$SoD_{efficiency} = \frac{SP\_u_{length}}{SoD_{capacity} * \left( f(Buff_{size}) + C \right)} * \left[ n + f(Buff_{size}) \right] * 100$$

units of measure: [SUT_$u_{efficiency}$] = $\left[ \frac{bits}{Kbps * ms} \right] = \left[ \frac{Kbps}{Kbps} \right]$ = [%]

Similar to the SoD goodput, the SoD efficiency is not influenced by the transport service's buffer size change, as we prove in the following paragraphs.

To derive goodput and efficiency values in experiments E9, E10 and E11, we need to understand the value of N. For all experiments it is valid to say, that since packets are generated at a higher rate (saturating the SUT), the transport service's buffer is always full. That would mean that there are always 56 packets in 32 KBytes buffer and correspondingly 112 packets in 64 KBytes size buffer. Moreover, because the SUT is saturated, as we proved in bottleneck analysis (section 6.3), the SUT is able to handle maximum 14 packets being transported at once. This analysis results in the total number of packets in the SoD as 70 packets for 32 KBytes buffer case and 126 packets for 64 KBytes buffer case. Figure 6.35 shows the SoD goodput and efficiency numbers for these two different transport service's buffer sizes.

| Transport service buffer size | SoD goodput [Kbps] | SoD efficiency [%] |
|---|---|---|
| 32 KBytes | 57,3 | 89,5 |
| 64 KBytes | 57,9 | 90,4 |

Figure 6.35 SoD goodput and efficiency versus transport service's buffer size

*Answer 5:* *Increasing the transport service's buffer size results in higher average transport delays. The SUT goodput and efficiency are not influenced by the transport service buffer size change in the cases we studied. However, further reduction (e.g. to 1 KByte) of the transport buffer size may have a significant influence on the goodput and efficiency behaviour. Changing the application protocol buffer size has no influence on delay or goodput.*

## 6.6 Scalability Analysis

In the previous section, we presented the performance evaluation of the MobiHealth transport service in case there is only one transport service user. However, it is interesting to understand the behaviour of the SUT while serving multiple users, i.e., what are the scalability characteristics of the SUT with respect to the number of the concurrent users. Therefore, this section presents the results of the SUT scalability analysis with a focus on the delay and goodput performance characteristics.

For the purpose of analysis, we used the results of scalability Experiments 2, 3 and 4, in which we conducted the indicative measurements for 1, 5 and 10 instances of concurrent service users. Particularly, in Experiment 2 we took the measurements for a single (sole) transport service user. In Experiments 3, we took measurements for 5 concurrent transport service users per geographical location of 10 $m^2$. In addition, in Experiment 4 we took measurements for 10 concurrent transport service users per geographical location of 10 $m^2$. Hence, we take the results from Experiment 2 as a benchmark performance

results for scalability analysis and we focus on the performance characteristics of the SUT as observed from the perspective of a particular (single) user participating in these experiments.

We do not provide a complete analysis of the performance characteristics experienced by the other transport service users participating in the scalability experiments[67]. Measurements graphs of raw data are accumulated in Volume 2, and used to explain the SUT's uplink and downlink behaviour. Moreover, we generated the conclusive graphs based on relevant statistical data from the measurements.

The SUT scalability analysis activities are divided into two steps:

1) Presentation of conclusive delay and goodput graphs (section 6.6.1)
2) Analysis of the delay and goodput conclusive graphs (section 6.6.2)

### 6.6.1 Conclusive Graphs

In this section the behaviour of the SUT's uplink and downlink is presented as observed in scalability Experiments 2, 3 and 4 by a single transport service user. We present the set of combined conclusive graphs, i.e., graphs with results of all these experiments[68]. We present graphs in the following order: firstly, graphs presenting the SUT delay behaviour and then graphs presenting the SUT goodput behaviour. The goodput behaviour of the SUT is derived from the delay behaviour. Graphs are the basis for the SUT behaviour analysis.

Due to the scale of this measurement activity, and the large volume of obtained raw data, we focus only on the SUT's uplink and downlink behaviour for particular group of SEs. Particularly, as for the MobiHealth system perspective, we focus on the SUT uplink behaviour. Therefore, the group of SEs selected for further analysis has the following characteristics: SP_u size arbitrary (is varied) and SP_d size is fixed at 174 Bytes. This choice of the packet size is made with respect to the BANip protocol, where a variable amount of patient monitoring data on the uplink and small amount of control data on the downlink is transported. As discussed in section 6.2.2, even if some SE measurement data has been obtained with a low accuracy (measured by the 95% confidence interval), this data is used for further analysis.

Figures 6.36 and 6.38 present a statistical average of the primary performance parameter *delay* and its accuracy as observed in three scalability experiments (E2, E3, and E4). Figures 6.37, 6.39 present a statistical average of the derived performance parameter *goodput* as observed in the experiments.

*Note: The lines in Figures 6.36 to 6.39 should be interpreted as lines connecting the measurement points for Experiment 2, 3 and 4.*

[67] SUT Performance characteristics are (statistically) identical for all participating users. Hence, it is sufficient to observe one user.
[68] Approach is similar to section 6.2.2, therefore we omitted a description of how do we generate the conclusive graphs.

Figure 6.36 presents the average SUT's uplink delay behaviour as a function of SP_u size. The horizontal axis provides the values of the SP_u size (in Bytes). The left vertical axis presents the corresponding average delay in milliseconds. The right vertical axis presents the statistical accuracy (*Acc*) of the measured delay.

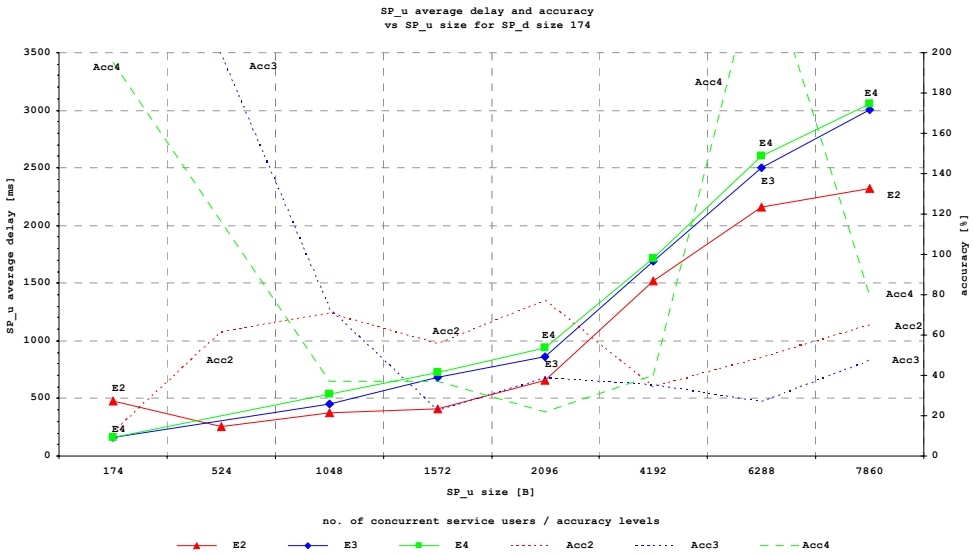



Figure 6.36 SUT uplink delay and accuracy versus SP_u size (1, 5, 10 service instances)

The conclusive graph presenting the goodput behaviour is based on the same SE data set used for delay analysis.

Figure 6.37 presents the SUT's uplink average goodput behaviour (per user) as a function of SP_u size. The horizontal axis provides the values of the SP_u size (in Bytes). The left vertical axis presents the corresponding average goodput in Kbps. The *efficiency* is indicated on the right vertical axis as a percentage of the nominal (maximum) uplink capacity (which is 64 Kbps) for the uplink's dedicated bearer 1. Two red horizontal lines indicate the SUT capacity boundaries for the common (16 Kbps) and dedicated bearer (64 Kbps) in the uplink direction.

Figure 6.38 presents the average SUT's downlink delay behaviour as a function of SP_u size. The horizontal axis provides the values of the SP_u size (in Bytes). The left vertical axis presents the corresponding average delay in milliseconds. The right vertical axis presents the statistical accuracy (*Acc*) of the measured delay.

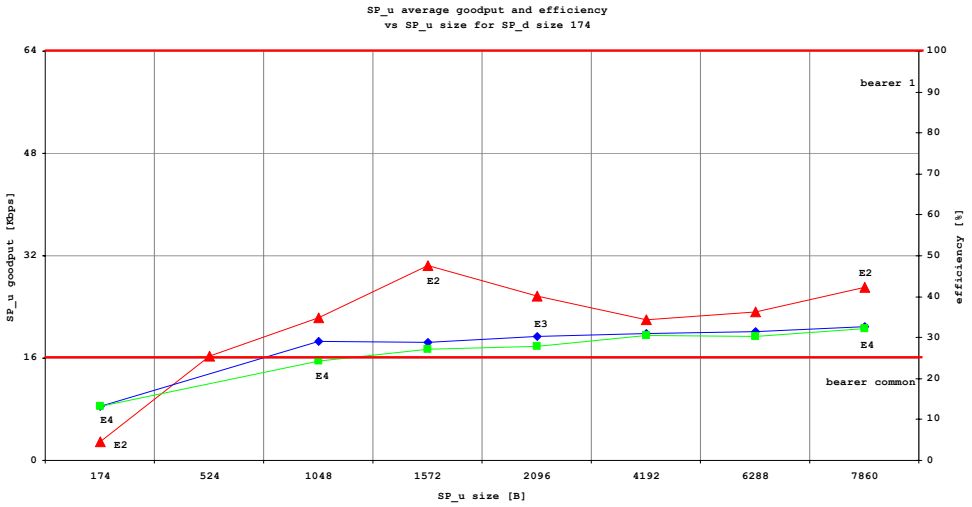



Figure 6.37 SUT uplink goodput and efficiency versus SP_u size (1, 5, 10 service instances)

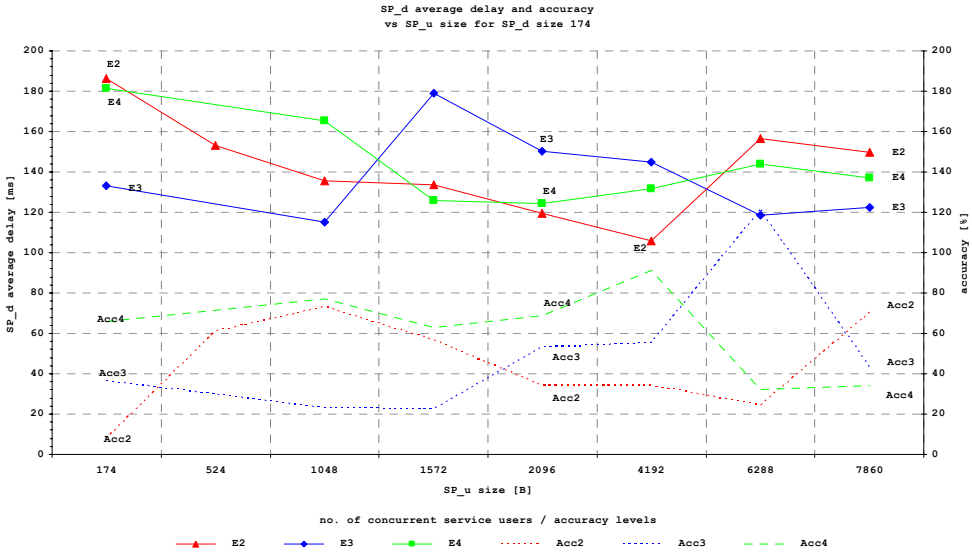



Figure 6.38 SUT downlink delay and accuracy versus SP_u size (1, 5, 10 service instances)

Figure 6.39 presents the SUT's downlink average goodput behaviour as a function of SP_u size. The horizontal axis provides the values of the SP_u size (in Bytes). The left vertical axis presents the corresponding average goodput in Kbps. Two red horizontal lines indicate the SUT capacity boundaries for the common (16 Kbps) and dedicated bearer 1 (64 Kbps) in the downlink direction.

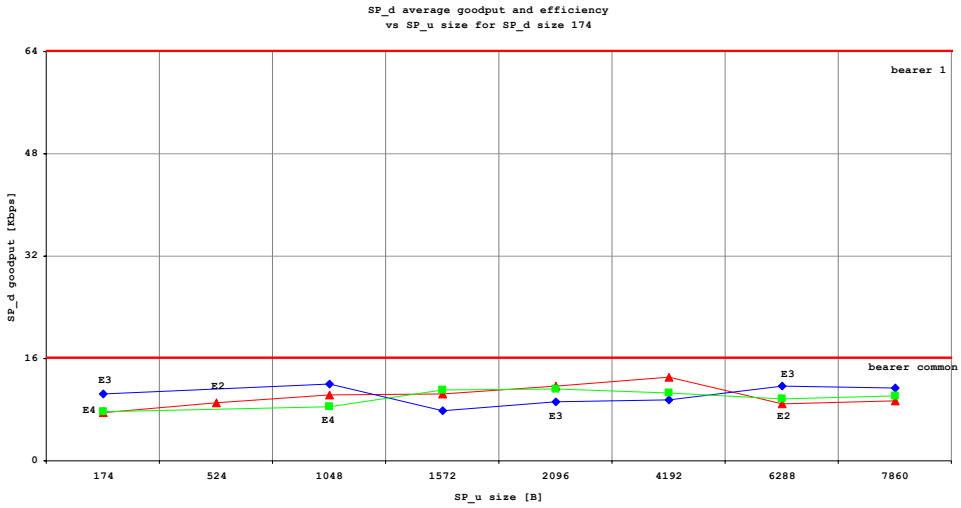



Figure 6.39 SUT downlink goodput per user versus SP_u size (1, 5, 10 service instances)

### 6.6.2 System Behaviour

In this section, we provide the results of the analysis of the SUT scalability characteristics as observed in scalability experiments, and presented in conclusive graphs. Unfortunately, the conclusive figures do not provide very clear understanding of the SUT behaviour; they are not complete. Hence, we conduct the scalability analysis based on our understanding of the SUT behaviour in Experiments 3 and 4 (multiple service users) with comparison to the Experiment 2 (sole user case). Scalability analysis is conducted separately for uplink and downlink.

To facilitate description, the following abbreviations are introduced:

d2, g2 meaning the average delay (d2) and goodput (g2) behaviour experienced by a single user in case of Experiment 2

d3, g3 meaning the average delay (d3) and goodput (g3) behaviour experienced by the transport service user in case of Experiment 3

d4, g4 meaning the average delay (d4) and goodput (g4) behaviour experienced by the transport service user in case of Experiment 4

The uplink delay behaviour's analysis is based on Figure 6.36 and lead to the following conclusions.

1) the SUT uplink average delay behaviour experienced by the service user is less predictable with increasing number of concurrent users; the variance of the delay increases. We believe that there are two possible reasons of this behaviour: 1) influence of intra-communication parameter i.e. Bluetooth interference, or/and 2) there are buffer delays experienced in the SUT. This is a subject of further research.

2) the difference between d2 and d3, d4 is significant for the SE: SP_u 174 Bytes and SP_d 174 Bytes. This is due to the fact, that during the measurements in E2, there was no dedicated bearer 1 assigned in the SUT, as it has been in case of E3 and E4 (see Figure 6.40). Due to this, the accuracy of obtained $SP\_u_{174}$ delay (calculated over the whole delay sample) for E3 and E4 is very low.

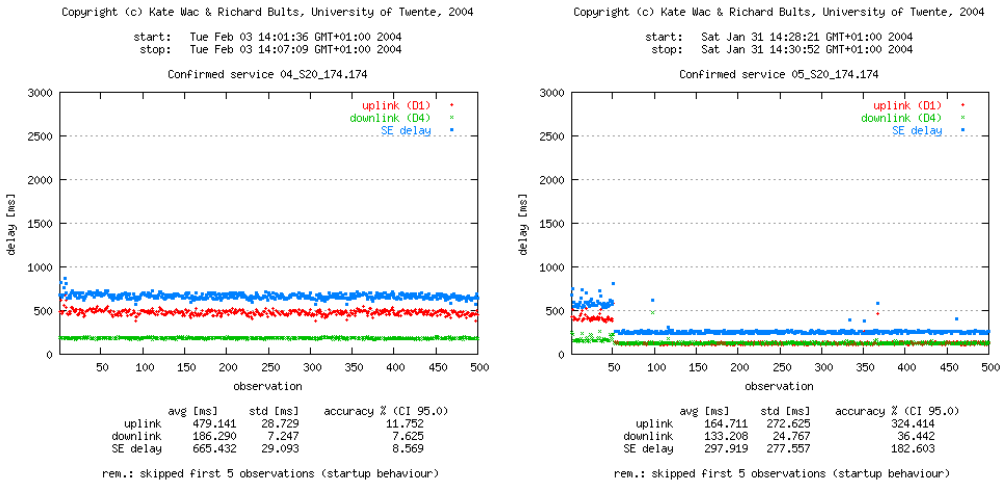



Figure 6.40 E2 and E3 for SE: $SP\_u_{174}$ $SP\_d_{174}$

3) For packets SP_u bigger than 174 Bytes, there is a significant difference between d2 and d3 (or d4). However, delay d2 is always lower than d3, and d3 is always lower than d4. The difference between d3 and d4 is always around 100 milliseconds.

4) The difference between d2 and d3 (or d4) is increasing for the packet SP_u increasing (and bigger than 174 Bytes). For SP_u 1048 Bytes the difference between d2 and d3 (or d4) is 100ms (200ms), while for 7860 Bytes is 700ms for both: d3 and d4.

5) The jitter is comparable for E2 and E3 (for SP_u bigger than 174 Bytes). Jitter for E4 seems to have no pattern in its behaviour, and is unpredictable, especially for a large packets. The uplink bearer alternating behaviour of the SUT; switching between the common bearer and the dedicated bearer 1, can cause the high jitter. This behaviour is unpredictable (see Figure 6.41) and occurs with higher probability for bigger packet sizes and for increasing number of concurrent users (E3 and E4).

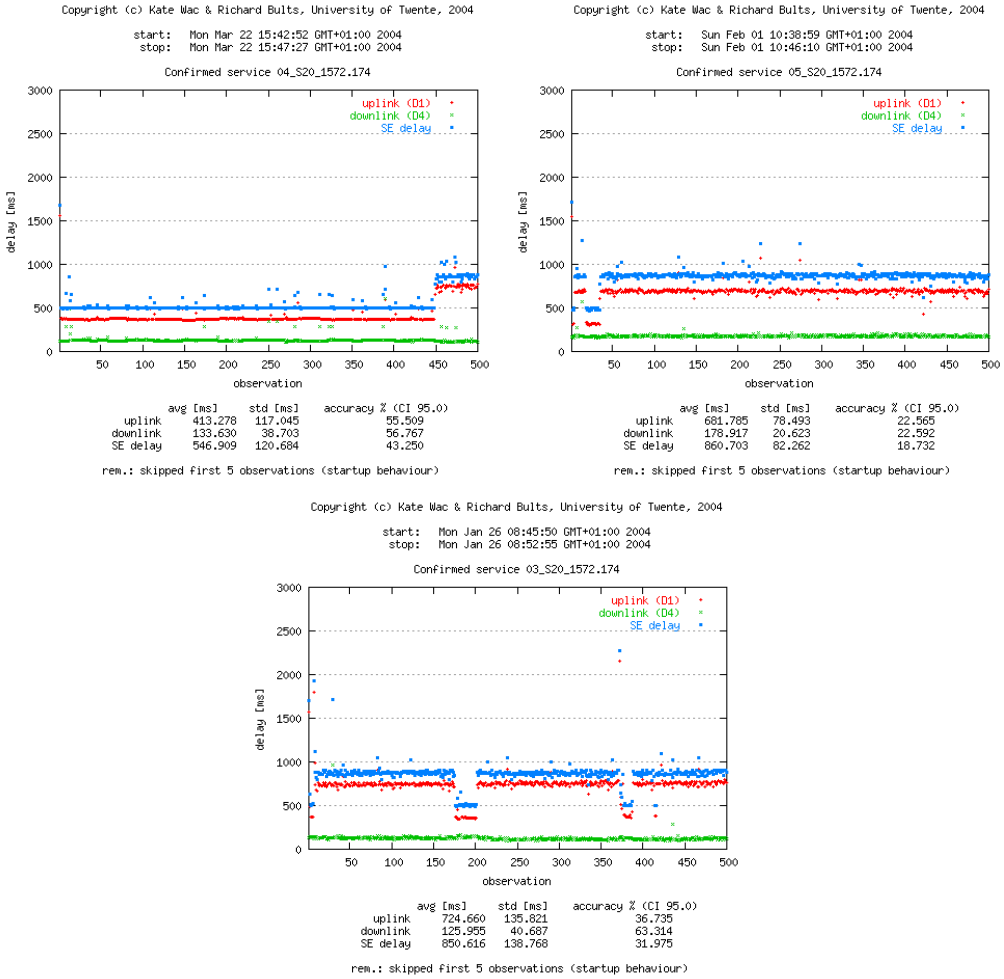



Figure 6.41 E2, E3 and E4 uplink bearer alternating behaviour

Figure 6.38 forms the basis of the downlink behaviour's delay analysis. We derive the following conclusions:

1) The downlink delay value is very much similar for all experiments (assumed 20ms to 40ms timing accuracy tolerances due to the instrumentation of the measurements; section 5.5.4).

2) The jitter seems to have no pattern in its behaviour, and is unpredictable for all SP_u sizes. Jitter is less predictable for bigger SP_u sizes and for the higher number of concurrent users.

The uplink goodput behaviour's analysis is based on Figure 6.37 and lead to the following conclusions:

1) the SUT uplink average goodput behaviour observed by a service user is lower as the number of concurrent users increases.
Generally, for a stable SUT delay, throughput and goodput (hence efficiency) per user, are not expected to change with number of users. We believe in an upper limit of users that can freely use the SUT in one geographical location, and we expect that this upper limit relate to the internal resources of the SUT. Consequently, the fact that for 5, 10 concurrent users we observe the degradation in SUT performance, we believe that for this number of users, we reach the upper limit of SUT usage and there are experienced buffer delays in the SUT.

2) the difference between g2 and g3, g4 is significant for the SE: SP_u 174 Bytes and SP_d 174 Bytes, due to the fact, that during the measurements in E2, there was no dedicated bearer 1 assigned in the SUT, as it has been in case of E3 and E4 (see Figure 6.40).

3) For packets SP_u bigger than 174 Bytes, there is a significant difference between g2 and g3 (or g4). The difference between g3 and g4 is small. The efficiency of the SUT usage for E2 is always higher than for E3 or E4, but it is never higher than 50%.

4) The difference between g2 and g3 (or g4) is increasing for the packet SP_u bigger than 174 Bytes and up to 1572 Bytes. For SP_u 1572 Bytes the difference between g2 and g3 (or g4) is the biggest and equals 10 Kbps. As it can be seen in Figure 6.38, for this packet size in E2 the SUT has assigned dedicated bearer 1, while for E3 and E4 the SUT has assigned common bearer or dedicated bearer 1 with high probability of bearer alternating behaviour. The difference in efficiency obtained between E2 and E3 (or E4) is the highest for the SP_u 1572 Bytes and equals 20%.

5) The difference between g2 and g3 (or g4) is variable for packet SP_u size bigger than 1572 Bytes. For big packet sizes in E2, E3 and E4 the SUT has assigned common bearer or dedicated bearer 1 with higher probability of bearer alternating behaviour, which result in low experienced goodput for all cases. Therefore, the efficiency of the SUT is low for bigger packet sizes and equals around 40%.

The downlink goodput behaviour's analysis is based on Figure 6.39 and leads to the following conclusions:

1) When taking into consideration the accuracy of measurements, it is fair to say that the SUT downlink average goodput behaviour experienced by the service user while concurring with other users is the same regardless of the number of other users.

2) Figure 6.39 may give an impression that for all cases: E2, E3 and E4 a common bearer is assigned by the SUT. This impression however is false; the SUT behaviour is alternating between the common bearer and dedicated bearer 1 (as discussed in the previous part of this section).

3) The goodput experienced by a user is lower than 16 Kbps regardless of the assigned bearer in the SUT.

4) The efficiency of the SUT usage is low, between 10% and 20% for all cases.

# 7 Conclusions, Recommendations and Future Work

Section 7.1 presents the main achievements and conclusions of this thesis and also provides answers to the research questions defined in chapter 1. Section 7.2 presents our recommendations for the existing MobiHealth system (version 3.1.1). Section 7.3 discusses potential subjects of interest for future research.

## 7.1 Conclusions

This section provides comprehensive answers (conclusions) to the research questions defined in chapter 1. The research questions provide the structure of this section; we recall each question first and answer it accordingly. An answer may also include a particular 'achievement' that was necessary to obtain the answer.

*Research question 1:*

*"What performance evaluation methodology can be applied to assess a transport system supporting the MobiHealth BANip?"*

Performance evaluation of transport systems is in many cases based on mathematical models and simulation of these systems. Performance evaluation of real-world transport systems by means of measurements is an alternative method. Given the fact that we had the availability over Vodafone's (pre) commercial UMTS network in the Netherlands, we made the choice to use measurements for the performance evaluation of this transport system.

We believe that the performance evaluation of a transport system by means of measurements can only be successful if a methodical approach towards the execution of the measurements activity is used. We knew about the existence of [Hoek1997]. It presents a precise and systematic approach towards performance evaluation of telecommunication networks (i.e. transport systems) by means of measurements. They base their approach on the methodical approach for performance evaluation of computer systems by [Jain1991].

Our requirements for the performance evaluation methodology (section 3.1) match the methodical phases discussed in [Hoek1997] and [Jain1991]. Hence, we adopt the methodologies of [Hoek1997] and [Jain1991] and use them to produce our performance evaluation methodology (achievement 1). It is a combination of measurements (section 3.2) and modelling (section 3.3) methodologies and consists of nine phases (section 3.4). Figure 7.1 presents our performance evaluation methodology.

| | |
|---|---|
| *1. State the Goals and System Definition* | |
| *2. List Services and their Outcomes* | |
| *3. Select Performance Criteria (i.e. Metrics)* | |
| *4. List System and Workload Parameters* | |
| *5. Select Factors and their Levels* | |
| *6. Select System and Workload Parameters* | |
| *7. Design and Execute the Experiments* | |
| *8. Analyse, Evaluate and Interpret the Data* | |
| | *a. Select Model Representation* |
| | *b. Parameterise the Model* |
| | *c. Validate and Verify the Model* |
| *9. Present the Results* | |

Figure 7.1 Performance evaluation methodology

*Research question 2:*

*What are the performance criteria that should be considered for measurements, modelling and evaluation of the transport system?"*

The third phase of our performance evaluation methodology addresses this question (section 4.5). Prior to this phase however, we have to execute phase one and two first.

Section 4.1 presents Vodafone's (pre) commercial UMTS network in the Netherlands (V3GNL) as the transport system of interest, and lists the objectives (goals) for the performance evaluation activity. These objectives are:

1) Characterize the quantitative behaviour of V3GNL as a MobiHealth transport service,
2) determine the optimal BANip packet size and packet rate for a specified (maximum) delivery time, and
3) determine if the packet sizes of the current BANip implementation are chosen adequately.

Our methodology provides a conceptual framework to decompose a transport system into sub-systems that are relevant to the performance measurements activity. Section 4.2 describes the framework and defines the V3GNL network as the System Under Test (SUT); i.e. the transport (sub) system of interest. This section also describes the SUT in relation to other sub-systems of the transport system, and with that concludes the first phase of the methodology.

Section 4.3 describes the implementation of phase two (List Services and their Outcomes). From a performance evaluation perspective, we are interested in the service(s) of the SUT. The external observable behaviour of the SUT defines these services. The SUT's IP service is of our interest because this service supports the MobiHealth BANip (TCP/IP based application protocol). The IP service characteristics

are analysed (section 4.3.2) to obtain the service outcomes. We conclude that the SUT offers an asymmetrical IP datagram service with a nominal uplink transport capacity of 64 Kbps and a nominal downlink transport capacity of 384 Kbps). In addition, the service capacity correlates to uplink and downlink transmission of datagrams. The SUT reacts to changes in the volume and/or rate of the datagrams offered at its Service Access Points (SAPs). It supports two (UMTS) bearers for uplink (0-16 Kbps and 16-64 Kbps) and four for downlink (0-16 Kbps, 16-64 Kbps, 64-128 Kbps and 128-384 Kbps).

Section 4.5 describes the third phase (Select Performance Criteria) of our methodology and provides the answer to this research question. In this section, we use the ITU-T 3 x 3 matrix approach as a generic method to identify the SUT performance criteria in a systematic way. The focal point of the objectives (see above) is aimed at the SUT's communication function *user information transfer* and its performance criterion *speed*. The *speed* criterion is linked to the primary performance parameter *delay*, which has two derived performance parameters *jitter* and *goodput*.

*Research question 3:*

*What model can be used to represent the transport system and what are the conditions under which the model is valid?*

Phase 8a-c of our performance evaluation methodology addresses this question. Prior to this phase however, we have to execute phases 4-7 first.

Listing the system and workload parameters applicable to the measurements activity is the goal of phase 4. System parameters include both the hardware and software parameters, which generally do not vary among various installations of the system. Workload parameters are characteristics of user's requests, which vary from one installation to the next. Section 4.7 lists the system and workload parameters.

Phase 5 of our methodology focuses on selecting factors (i.e. parameters to be varied) and assigning values (i.e. levels) to the system and workload parameters. Section 4.7 describes this activity and concludes with an overview of the selected workload and system parameters and their assigned values (Figure 4.21 and Figure 4.22 respectively).

Phase 6 performs the selection of system and workload parameters. Section 5.1 describes the selection activity. It explains how the list of parameters and their values defined in phase 5 is used as input for this activity. The selection criteria are based on the relevance of these parameters for our goal of the SUT's performance evaluation. Figure 5.7 provides an overview of the selected system and (associated) workload parameters.

The activities of phase 7 (Design and Execute the Experiments) of our methodology imply the use of an evaluation system to perform the performance measurements. We designed and implemented our own (JAVA based) distributed evaluation system capable of automatic generation of different workloads for a particular set of system parameters (achievement 2). The evaluation system's components (i.e. workload generator and measurement function) facilitate one-to-one (i.e. server/client) scenarios or many-to-one

(i.e. servers/client) scenarios. Section 5.2 describes the functional model of the evaluation system. In addition, section 5.3 provides the most important evaluation system's implementation details. Section 5.4 describes the design and execution of various experiments; each experiment has a different set of system and workload parameters.

The execution of the experiments results in high quantities of raw measurement data. We developed an elementary statistical application (achievement 3) to support the data evaluation activity of phase 8. This application processes the raw data to derive statistical information, and visualises (by means of graphs) the raw data in conjunction with the statistical information (e.g. Figure 6.1). The preliminary evaluation of the measurements results provides insight in the quality of the measurements. We conclude phase 7 if the preliminary evaluation of the measurements data results in a raw data set of "good" quality. Section 5.5 provides detailed information about the measurements design and evaluation of data.

All the preceding activities are necessary to answer research question 3. The answer is directly related to phase 8a-c of our methodology. The conclusion of the first activities of phase 8 (Analyse, Evaluate and Interpret the Data) provides the foundation for model representation selection (section 6.1). The high-level knowledge of the SUT (i.e. transport system of interest) as a system, limits the level of sophistication of the model. In addition, several boundary conditions are applicable to the conception of the model:

1) focus on the SUT's uplink behaviour
2) one fixed packet size (524 Bytes) used for parameterisation, validation and verification
3) only light-load scenarios are considered to avoid buffer overflow and to guarantee a stable operation for the unconfirmed service scenario.

We developed a D/G/3 queuing model in section 6.1 that represents the uplink of the SUT. In addition, the model is parameterised, validated and verified based on a specific set of performance measurement data (see boundary conditions 2+3).

*Research questions 4 and 5:*

Both research questions aim to find the optimal BANip PDU size (and PDU transmission rates in case of research question 5) for UMTS based transport systems. We answer both questions based on the measurements results of the SUT (i.e. transport system of interest). Moreover, we use a top down approach to analyse the complete BANip PDU transport chain from a time (delay) perspective. The execution time of the activities prior to the construction of a BANip PDU (i.e. MobiHealth SDU assembly), and the relationship between BANip PDU sizes and the SUT's transmission time are the focal points of the analysis. Therefore, to find the optimal BANip PDU size, we consider the transport service utilisation and total time required to deliver a MobiHealth SDU from the MBU to the BEsys. Furthermore, we do not consider any issues regarding an optimal BANip overhead/payload ratio.

The MobiHealth project does not define the delay requirements for transmission of BAN sensor-system data (encapsulated in a MobiHealth SDU) from the MBU to the BEsys. Therefore, we specify two scenarios that investigate the consequences of using small and large MobiHealth SDUs, which are transported by BANip PDUs. In each scenario, we answer research questions 4 and 5. In the first scenario, every SDU holds an individual BAN sensor-system sample. This SDU is transmitted immediately from the MBU to the BEsys. In the second scenario, a SDU holds aggregate BAN sensor-system samples. This SDU is transmitted from the MBU to the BEsys after an a priori specified number of aggregated samples (packetizer aggregation number). We analyse both scenarios in the following paragraphs.

Scenario 1: individual samples

In this scenario, a MobiHealth SDU holds one BAN sensor-system sample. The BANip protocol transports (via the SUT) this SDU immediately to the BEsys, and we assume an application buffer size of one BANip PDU[69]. A MobiHealth SDU size equals 19 Bytes (packetizer aggregation number is 1) with a sampling frequency of 256 Hz. Furthermore, we assume a deflation factor of 52%[70] for this (small) SDU size. Therefore, the total amount of data (i.e. $HTTP_{PDU}$) transported by the SUT equals 27 Bytes, and the BANip PDU size is 20 Bytes (chapter 2 provides a calculation example for this HTTP PDU size). Finally, we assume that if a newly generated MobiHealth SDU cannot be placed in the application buffer, this SDU is dropped (i.e. lost).

We focus on the estimation of the total elapsed time from the MobiHealth SDU assembly (includes sample aggregation) at the MBU side, to the complete reception of this SDU by the BEsys. This elapse time is represented by the SDU delay and consists of three components: SDU assembly time (Ta), transport service propagation time (Tp) and HTTP PDU transmission time (Ts). Recall the description of the "BANip PDU format and exchange" in chapter 2. The BANip and HTTP protocol are unified and behave as an unconfirmed service. In chapter 5, we use workload generator data to represent a SDU encapsulated by a BANip PDU, which in turn is encapsulated by a HTTP PDU. Therefore, it is valid to use the obtained performance measurements data for our analysis of the SDU delay.

> *Note: We assume the execution time to be zero for all other activities related to the transmission of a MobiHealth SDU (e.g. BANip PDU or HTTP PDU assembly and disassembly). In addition, the assembly process of a new SDU is executed in parallel to the transmission of the previously assembled SDU.*

Figures 7.2 and 7.3 present the SDU delay components. We use these figures to answer research questions 4 and 5.

[69] The amount of arriving BANip PDUs per unit of time is smaller than the amount of BANip PDUs that the SUT is able to handle in unit time, therefore the SUT is stable. If not, the SUT is overloaded and the number of BANip PDUs residing in the application buffer will grow to infinity; situation that is inappropriate for time analysis.
[70] The measured average deflation factor for a MobiHealth SDU of 3800 Bytes is 52%. The correlation between the deflation factor and other MobiHealth SDU sizes is unknown.

*Note: The proportions between the SDU delay components in the figures are indicative.*

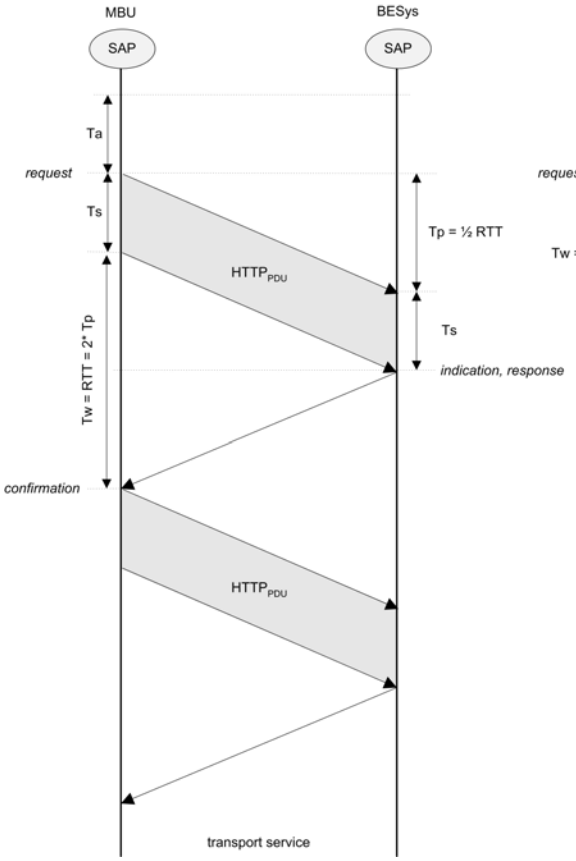



Figure 7.2 User confirmed service

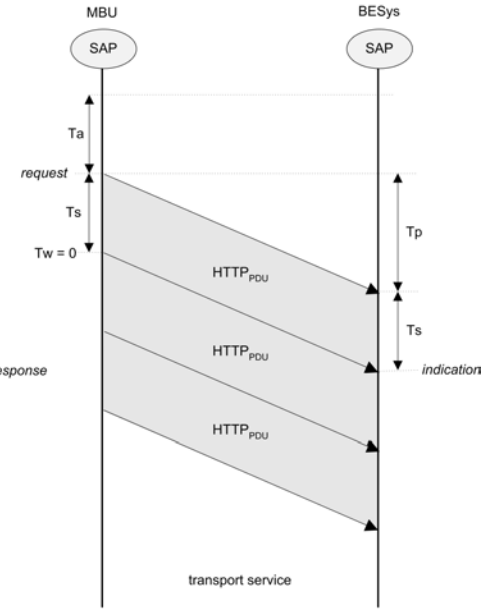



Figure 7.3 User unconfirmed service

The values of the SDU delay components are:

Ta = $P_{agg}$ / sampling frequency
= 1 / 256 = 4 ms
Tp = ½ RTT[71]
= 86 ms (derived in section 6.3)
Ts = $HTTP_{PDU}$ transmission time
= 0.138 * 27 = 4 ms (derived in section 6.3)

[71] The uplink and downlink bearers of the SUT are equal. The nominal capacity for both bearers is 64 Kbps.

Hence, the delay for the first SDU to arrive at the BEsys is:

Ta + Tp + Ts = 94 ms

If Ta ≤ Ts + 2*Tp, Figure 7.2 shows that the SDU delay for a BANip confirmed service is equal to 2*Ts + 3*Tp (266 ms) for all other (i.e. the second and following) SDUs. In contrast, Figure 7.3 shows no increase of SDU delay for all other SDUs in case of a BANip unconfirmed service.

At the BEsys side, we define the transport service idle time as the waiting time Tw. It indicates the time span between the end of the reception of a HTTP PDU and the start of the reception of a new HTTP PDU.

*Note: Because the BANip PDU and HTTP PDU assembly and disassembly times are assumed zero, and we ignore for simplicity reasons the BANip and HTTP protocol overhead transmission times, it is valid to interchange a HTTP PDU and BANip PDU with a MobiHealth SDU. This assumption applies to the analysis of the first and second scenario.*

*Research question 4:*

*Consider the MobiHealth BANip as a user confirmed application protocol. What is the optimal PDU size of a UMTS based transport system?*

Figure 7.2 presents the SDU delay components in case the BANip protocol is based on a confirmed service.

*Note: We assume that in a confirmed service the size of the data send in the response Service Primitive is very small compared to the data sent in the request Service Primitive (Chapter 2, Figure 2.15 presents the confirmed service concept). Therefore, we assume that the transmission time of the response Service Primitive equals zero. This assumption holds for analysis of the first and second scenario.*

We observe that Tp is the major SDU delay component. At the MBU side, Ta and Ts are equal. Hence, there is always a (newly assembled) SDU ready to be transmitted when the transmission of the current SDU is finished. However, in case of a confirmed service a (newly assembled) SDU is transmitted only when a confirmation message (i.e. confirm Service Primitive) is received by the MBU from the BEsys.

We conclude that for Ta ≤ Ts + 2*Tp, the waiting time is:

Tw = 2 * Tp
= 172 ms

From a BEsys perspective, the waiting time (Tw) to receive a SDU (encapsulated in the BANip PDU and HTTP PDU) is longer than the actual transmission time (Ts) of this SDU. The underlying transport service is busy with the transmission of a SDU for $\frac{Ts}{Ts+Tw}$ = 3% of the total time Ts + Tw.

We conclude that for a confirmed service, the transport service is not used efficiently for a BANip PDU size of 20 Bytes. Recall, this PDU size is determined by the size of one MobiHealth BAN sensor-system sample.

From a MobiHealth system end-user perspective, the utilisation of the transport service is secondary to the perceived SDU delay. We extend the conclusion above by providing the SDU delay for a BANip PDU size of 20 Bytes; the SDU delay is equal to 266 ms.

Section 7.2 provides recommendations for the MobiHealth system end-user and BANip application protocol designer. The end-user is able to select a required SDU delay and the protocol designer can derive the corresponding BANip PDU size.

*Research question 5:*

*Consider MobiHealth BANip as a user unconfirmed application protocol. What is the optimal PDU size and PDU transmission rate of a UMTS based transport system?*

Figure 7.3 presents the SDU delay components in case the BANip protocol is based on a unconfirmed service. Recall, that we focus only on the second and following SDU transmissions.

At the MBU side, if Ta ≤ Ts, then there is always a (newly assembled) SDU ready to be transmitted when the transmission of the current SDU is finished (back-to-back transmission).

We conclude that for Ta ≤ Ts, the waiting time is Tw = 0 ms.

From a BEsys perspective, the waiting time (Tw) to receive a SDU (encapsulated in the BANip PDU and HTTP PDU) is zero. The underlying transport service is busy with the transmission of a SDU for $\frac{Ts}{Ts+Tw}$ = 100% of the total time Ts + Tw.

We conclude that for an unconfirmed service, the transport service is used efficiently for a BANip PDU size of 20 Bytes. Recall, this PDU size is determined by the size of one MobiHealth BAN sensor-system sample. However, from a transport system perspective, any BANip PDU size will yield 100% transport system efficiency.

From a MobiHealth system end-user perspective, the utilisation of the transport service is secondary to the perceived SDU delay and SDU transmission rate. We extend the conclusion above by providing the SDU delay and SDU transmission rate for a BANip PDU size of 20 Bytes. The SDU delay is 94 ms with a transmission rate of 256 SDUs per second. We elaborate further on this scenario of *MobiHealth SDUs streaming* in section 7.2.

*Note: We do not consider any issues regarding BANip and lower level protocols overhead. Although from a BANip service user perspective this scenario seems feasible, lower level transport services may not be able to support this scenario. At the BANip service level the data rate is 21 Kbps (256 SDUs/s * 10 Bytes * 8), while for example the data rate at the SUT IP service access point is 96 Kbps if we include protocol overhead (BANip, HTTP, TCP).*

Section 7.2 provides recommendations for the MobiHealth system end-user and BANip application protocol designer. The end-user is able to select a required SDU delay and the protocol designer can derive the corresponding BANip PDU size.

Scenario 2: aggregated samples

In this scenario, a MobiHealth SDU holds 200 BAN sensor-system samples (MobiHealth version 3.1.1 implementation), and the BANip protocol transports (via the SUT) this SDU immediately to the BEsys, and we assume an application buffer size of one BANip PDU. A MobiHealth SDU size equals 3800 Bytes (packetizer aggregation number is 200) with a sampling frequency of 256 Hz. The deflation factor is 52% for this SDU size. Therefore, the total amount of data ($HTTP_{PDU}$) transported by the SUT equals 1841 Bytes, and the BANip PDU size is 1834 Bytes (chapter 2 provides a calculation example for this HTTP PDU size). Finally, we assume that if a newly generated MobiHealth SDU cannot be placed in the application buffer, this SDU is dropped (i.e. lost).

We focus on the estimation of the total elapsed time from the MobiHealth SDU assembly (includes sample aggregation) at the MBU side, to the complete reception of this SDU by the BEsys. This elapsed time is represented by the SDU delay and consists of three components: SDU assembly time (Ta), transport service propagation time (Tp) and HTTP PDU transmission time (Ts).

*Note: We assume the execution time to be zero for all other activities related to the transmission of a MobiHealth SDU (e.g. BANip PDU or HTTP PDU assembly and disassembly). In addition, the assembly process of a new SDU is executed in parallel to the transmission of the previous assembled SDU.*

Figures 7.4 and 7.5 present the SDU delay components. We use these figures to answer research questions 4 and 5.

*Note: The proportions between the SDU delay components in the figures are indicative.*

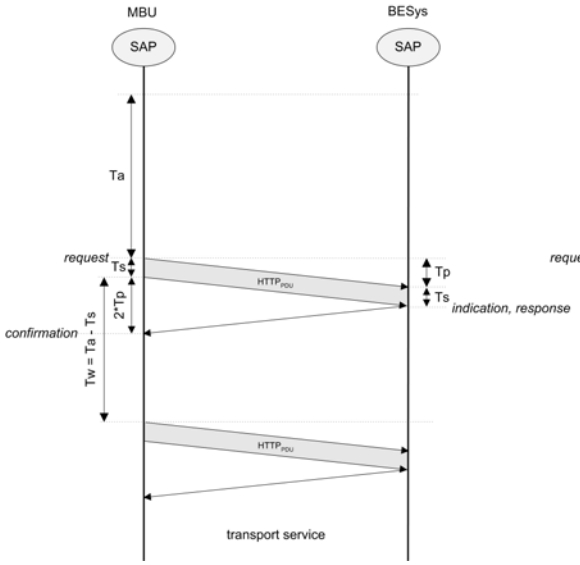



Figure 7.4 User confirmed service

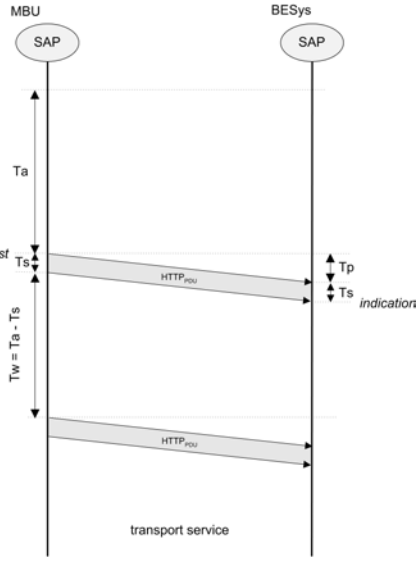



Figure 7.5 User unconfirmed service

The values of the SDU delay components are:

Ta = $P_{agg}$ / sampling frequency
= 200 / 256 = 782 ms
Tp = ½ RTT[72]
= 86 ms (derived in section 6.3)
Ts = $HTTP_{PDU}$ transmission time
= 0.138 * 1841 = 255 ms (derived in section 6.3)

Recall, the assembly of a new SDU starts in parallel with the transmission of the previous assembled SDU, and observe that Ta > Ts + 2*Tp.

Hence, the delay for the first SDU to arrive at the BEsys is:

Ta + Tp + Ts = 1123 ms

If Ta > Ts + 2*Tp, Figure 7.4 and 7.5 show that the SDU delay for a BANip confirmed and unconfirmed service is equal to Ta + Tp + Ts (1123 ms) for all other (i.e. the second and following) SDUs.

[72] The uplink and downlink bearers of the SUT are equal. The nominal capacity for both bearers is 64 Kbps.

At the BEsys side, we define the transport service idle time as the waiting time Tw. Recall, that it indicates the time span between the end of the reception of a HTTP PDU and the start of the reception of a new HTTP PDU. Because the BANip PDU and HTTP PDU assembly and disassembly times are assumed zero, and for simplicity reasons we ignore the BANip and HTTP protocol overhead transmission times, it is valid to interchange a HTTP PDU with a MobiHealth SDU.

*Research question 4:*

*Consider the MobiHealth BANip as a user confirmed application protocol. What is the optimal PDU size of a UMTS based transport system?*

Figure 7.4 presents the SDU delay components in case the BANip protocol is based on a confirmed service.

We observe that Ta is the major SDU delay component. Hence, there is never a (newly assembled) SDU ready to be transmitted when the transmission of the current SDU is finished. In case of a confirmed service, a (newly assembled) SDU is transmitted only when a confirmation message (i.e. confirm Service Primitive) is received by the MBU from the BEsys. Because Ta > Ts + 2*Tp, this message is received before Ta elapses.

We conclude that for a MobiHealth SDU holding 200 BAN sensor-system samples, Ta > Ts + 2*Tp, and the waiting time is:

$$\begin{aligned} Tw &= Ta - Ts \\ &= 527 \text{ ms} \end{aligned}$$

From a BEsys perspective, the waiting time (Tw) to receive a SDU (encapsulated in the BANip PDU and HTTP PDU) is longer then the actual transmission time (Ts) of this SDU. The underlying transport service is busy with the transmission of a SDU for $\frac{Ts}{Ts + Tw}$ = 33% of the total time Ts + Tw.

We conclude that for a confirmed service, the transport service is not used efficiently for a BANip PDU size of 1834 Bytes. Recall, this PDU size is determined by the size of 200 MobiHealth BAN sensor-system samples. Hence, from a transport system perspective, 1834 Bytes is not an optimal BANip PDU size.

From a MobiHealth system end-user perspective, the utilisation of the transport service is secondary to the perceived SDU delay. We extend the conclusion above by providing the SDU delay for a BANip PDU size of 1834 Bytes; the SDU delay is equal to 1123 ms.

Section 7.2 provides recommendations for the MobiHealth system end-user and BANip application protocol designer. The end-user is able to select a required SDU delay and the protocol designer can derive the corresponding BANip PDU size.

*Research question 5:*

*Consider MobiHealth BANip as a user unconfirmed application protocol. What is the optimal PDU size and PDU transmission rate of a UMTS based transport system?*

Figure 7.5 presents the SDU delay components in case the BANip protocol is based on a unconfirmed service. Recall, that we focus only on the second and following SDU transmissions.

We observe that Ta is the major SDU delay component, and Ta > Ts. Hence, there is never a (newly assembled) SDU ready to be transmitted when the transmission of the current SDU is finished.

We conclude that for a MobiHealth SDU holding 200 BAN sensor-system samples, Ta > Ts, and the waiting time is:

$$\begin{aligned} Tw &= Ta - Ts \\ &= 527 \text{ ms} \end{aligned}$$

From a BEsys perspective, the waiting time (Tw) to receive a SDU (encapsulated in the BANip PDU and HTTP PDU) is longer then the actual transmission time (Ts) of this SDU. The underlying transport service is busy with the transmission of a SDU for $\frac{Ts}{Ts + Tw}$ = 33% of the total time Ts + Tw.

We conclude that for an unconfirmed service, the transport service is not used efficiently for a BANip PDU size of 1834 Bytes. Recall, this PDU size is determined by the size of 200 MobiHealth BAN sensor-system samples. However, from a transport system perspective, any BANip PDU size will not yield 100% transport system efficiency.

From a MobiHealth system end-user perspective, the utilisation of the transport service is secondary to the perceived SDU delay and SDU transmission rate. We extend the conclusion above by providing the SDU delay and SDU transmission rate for a BANip PDU size of 1834 Bytes; the SDU delay equals 1123 ms and its transmission rate equals 1.3 SDUs per second.

At the BANip service level the data rate is 19 Kbps (1.3 SDUs/s * 1834 Bytes * 8), while the data rate at the SUT IP service access point is 19,5 Kbps if we include protocol overhead (BANip, HTTP, TCP). The SUT is able to support bit rates up to 63,8 Kbps (=64Kbps nominal uplink capacity – IP and PPP protocol overhead)at the IP SAP. Hence, the SUT is capable to support SDU rates up to 1.3 SDU/s.

Section 7.2 provides recommendations for the MobiHealth system end-user and BANip application protocol designer. The end-user is able to select a required SDU delay and the protocol designer can derive the corresponding BANip PDU size.

*Research question 6:*

*What are the relevant performance issues related to the transport system?*

The final phase of our performance evaluation methodology focuses on the presentation of the results. We use these results to provide an answer to this question. Sections 6.3-6.6 cover the activities of the final phase and provide an assessment of the SUT's performance. The assessment is based on the measurements results and the developed model. We deduct five SUT related performance issues: capacity switching behaviour, goodput, delay variation, influence of system parameters on the SUT's behaviour and scalability characteristics.

Capacity switching behaviour

The capacity performance characteristics of the SUT are changing in time. We conclude from the measurements data, that the data size and rate offered to the SUT causes capacity changing behaviour. We specify the capacity switching behaviour as 'bearer assignment' or 'bearer switching'.

For an initial data transport request, the SUT assigns the common bearer (shared bearer with low capacity) by default to both uplink and downlink directions. For additional data transport requests, it assigns a dedicated bearer (with a higher capacity) that 'matches' the required transmission rate and volume. Matching in this context means that the SUT recognizes the volume and rate of an ingress data stream, and responds with assigning a dedicated bearer. There are two bearers for the uplink direction: common bearer 16 Kbps and dedicated bearer 1 of 64 Kbps. We distinguish four bearers in the downlink direction: common bearer 16 Kbps and dedicated bearers 1, 2 and 3 respectively: 64 Kbps, 128 Kbps and 384 Kbps (nominal capacities).

The bearer assignment behaviour is always gradual, i.e., the SUT assigns the adjacent bearer (to the current one) to both directions. This behaviour is considered to be normal if it converges to a dedicated bearer and this bearer is going to be used for the remaining part of a data transfer. This behaviour is validated for 99.8% of the executed performance measurements. The rest (0.2%) of the measurements showed an alternating behaviour of the downlink.

The exact conditions, under which a bearer assignment takes place, became not clear to us. Therefore, we indicate different SUT behaviour regions in which there is a relation between the data size transported and the assigned bearer. For the uplink direction, the SUT assigns common bearer for data sizes smaller than 174 Bytes and dedicated bearer 1 for data sizes larger than 174 Bytes. We distinguish four behaviour regions for the downlink. For data sizes smaller than 2096 Bytes, the SUT assigns the common bearer or dedicated bearer 1. For data sizes in the range of 2096 Bytes to 10480 Bytes, dedicated bearer 1 or dedicated bearer 2 is assigned. For data sizes in the range of 10480 Bytes to 23056 Bytes, dedicated bearer 2 or dedicated bearer 3 is assigned. Finally, for data sizes larger than 23056 Bytes, the SUT assigns dedicated bearer 3. Hence, the observed SUT

goodput and delay variation performance parameters depend directly on the bearer assignment behaviour.

Goodput

We typify the UMTS based SUT as 'goodput bottleneck' system. For the uplink direction, the maximum goodput is 57 Kbps and for the downlink approximately 300 Kbps. The bottleneck in the downlink direction however, differs in case the Bluetooth option of the pre-commercial Nokia 6650 UMTS terminal (both are system parameters) is used. Theoretically, the Nokia terminal has a nominal capacity of 115 Kbps for a Bluetooth link, but the measurements proved that only 83 Kbps was obtained. We assume a Bluetooth implementation problem of the Nokia terminal is the cause of this limitation.

Delay variation

We observed significant delay variation (i.e. jitter) in the SUT. This has a negative influence on the delay performance characteristics of the (MobiHealth) transport system as a whole; recall, the SUT is a subsystem of this (larger) transport system. We identify three possible sources of jitter: 1) SUT bearer assignment, 2) packet loss in one of the SUT's subsystems (causing UMTS or TCP retransmissions), and 3) resource problems in one of the SUT's subsystems.

Influence of system parameters on SUT behaviour

We defined *system parameters of influence* as the parameters that influence the behaviour of the SUT. The communication system used between the terminal and a computer system was denoted as *intra communication system* and is comparable to the MobiHealth intra BAN communication system. The conclusions provided below are therefore usable for the MobiHealth intra-BAN communication system's performance evaluation.

We consider the following system parameters of influence: intra communication system, UMTS terminal, computer system, transport service and application buffer size, and inclusion of the Internet subsystem in the transport system configuration[73]. The default system parameters were: notebook computer system, USB intra communication system, (pre-commercial) Nokia 6650 UMTS terminal, transport service and application buffer size each of 64 Kbytes, and the UTnet subsystem included in the transport system's configuration.

*intra communication system*

The SUT uplink behaviour is similar when changing the intra communication system from USB to Bluetooth (for the same UMTS terminal). The influence of the change is significant for the SUT downlink behaviour when using packet sizes larger than 524

[73] We consider a (MobiHealth) transport system that consists of two sub-systems: the SUT and the Internet (including UTnet)

Bytes. The Bluetooth instantiation of the intra communication system was identified as a goodput bottleneck. The maximum obtained goodput for downlink communication was 83 Kbps (nominal capacity 115 Kbps).

*UMTS terminals (interface)*

The SUT's uplink and downlink behaviour is similar for a Nokia 6650 UMTS terminal (USB interface to computer system) and a PC Card terminal (PCMCIA interface to computer system).

*computer system*

The SUT's uplink and downlink behaviour is similar when changing computer systems from a notebook to an iPAQ (Bluetooth intra communication). However, for both systems, the Bluetooth intra communication is a goodput bottleneck for the downlink.

*transport service and application buffer sizes*

Increasing the transport service's buffer size (i.e. TCP send/receive buffers) results in higher average data transport delays. The SUT goodput and efficiency are not influenced for transport service buffer sizes of 32 Kbytes and 64 KBytes. Changing the application protocol buffer size (i.e. socket buffer) has no influence on the SUT's delay or goodput.

*influence of the Internet*

There is no difference in the SUT uplink and downlink behaviour between case when the Internet subsystem is introduced in the transport system configuration and when it is not (for packet sizes between 174 and 1572 bytes).

Scalability

We also conducted indicative performance measurements to assess the scalability performance characteristics of the SUT. We do not provide complete analysis of the performance characteristics experienced by all transport service users participating in the scalability experiments. SUT performance characteristics are (statistically) identical for all participating users. Hence, it is sufficient to observe one user.

Generally, for a stable SUT (e.g. no loss of data), the delay, throughput and goodput per user are expected not to change significantly, when the number of users increases. However, we observe that if the SUT is concurrently used by 10 end-users, an end-user experiences significant system performance degradation compared to a single end-user scenario. In the uplink, the SUT performance degradation is observed as an increase (sometimes double) of delays and in addition, a significant increase of delay variation. Moreover, the observed SUT goodput decreases by 50%.

This behaviour can be explained by the fact that the SUT supports only a limited number of concurrent users per geographical location of 10 $m^2$ due to SUT resources limitations.

Therefore, for an increasing number of users an increased delay variation may be observed. Another possible reason of this behaviour is the use of pre-commercial Nokia 6650 UMTS terminals. These terminals may experience Bluetooth or UMTS communication related buffering problems. We conclude that the upper limit of SUT users is 10 per geographical location of 10 $m^2$; however, this is subject for future research. The area of a geographical location and amount of concurrent users supported at a single location must be investigated.

## 7.2 Recommendations

This section provides recommendations for the MobiHealth system end-user and BANip application protocol designer. The end-user is able to select a required SDU delay and the protocol designer can derive the corresponding BANip PDU size. To find this optimal BANip PDU size we also consider the SDU rate and transport service throughput as important criteria for the protocol designer, as a proof that the transport service can support certain SDU delays and BANip PDU sizes.

*Note: We use the second (and following) SDU transport delay values throughout this section.*

The MobiHealth project does not define the delay requirements for transmission of BAN sensor-system data (encapsulated in a MobiHealth SDU) from the MBU to the BEsys. Therefore, the recommendations we give in this section indicate the possible range of SDU delays/rates and the corresponding BANip PDU sizes that can be supported by the transport service.

SDU delay

In section 7.1, we analysed two scenarios that investigated the consequences of using small and large MobiHealth SDUs, which are transported by BANip PDUs. We identified the SDU delay components: Ta, Tp and Ts.

For a BANip confirmed service:

if $Ta \leq Ts + 2*Tp$ (scenario 1)
then SDU delay = $2*Ts + 3*Tp$

if $Ta > Ts + 2*Tp$ (scenario 2)
then SDU delay = $Ta + Tp + Ts$

For a BANip unconfirmed service:

if Ta ≤ Ts (scenario 1)
then SDU delay = 2*Ts + Tp

if Ta > Ts (scenario 2)
then SDU delay = Ta + Tp + Ts

A SDU is deflated and encapsulated in a BANip PDU. If we assume zero deflation time, then Ta is a function of the SDU size[74] and Ts is a function of the BANip PDU size; Tp is constant. Figure 7.6 presents the SDU delay perceived by the MobiHealth system end-user as a function of the BANip PDU size for confirmed and unconfirmed service scenarios. The horizontal axis represents the BANip PDU size (Bytes) corresponding to all possible packetizer aggregation numbers [1..255]. The vertical axis represents the corresponding SDU delay (milliseconds). The red line represents the SDU delay for a confirmed service scenario and the blue line represents the SDU delay for an unconfirmed service scenario. The vertical red line indicates the BANip PDU size (1834 Bytes) for the current MobiHealth system implementation.

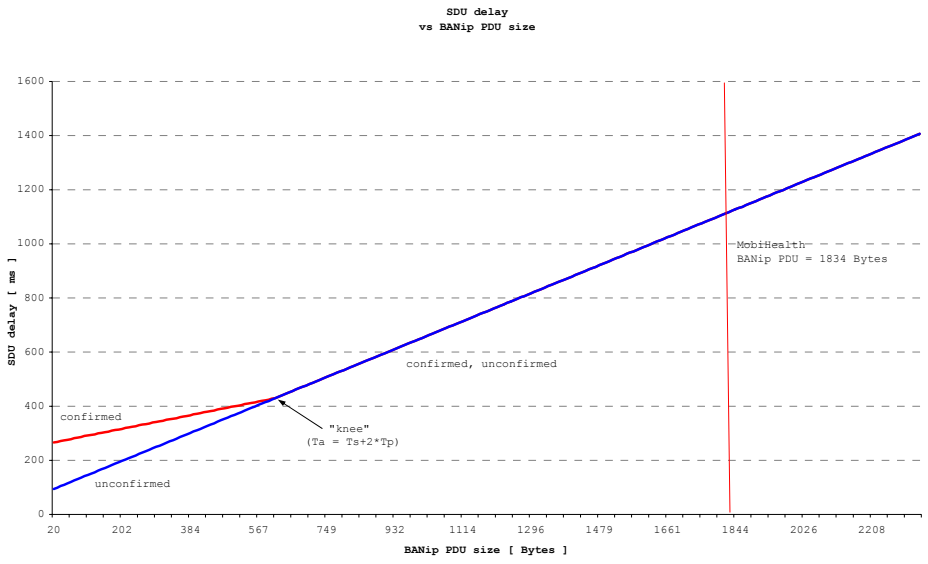



Figure 7.6 SDU delay versus BANip PDU size for confirmed and unconfirmed service

[74] Ta is a function of Pagg (Ta = Pagg / samplingFreq) and therefore an indirect function of the SDU size

We conclude from Figure 7.6, that for unconfirmed and confirmed services the SDU delay is proportional to the BANip PDU size. However, for a confirmed service there is a "knee" at BANip PDU size of 612 Bytes (corresponds to packetizer aggregation number 66). The SDU delay for this "knee" equals Ta+Ts+Tp (430 ms).

SDU rate

The SDU delay analysis may be interesting for a MobiHealth system end-user, but we believe that the supported SDU rate analysis is more important to the end-user. The end-user may raise the following questions: "What sampling frequency of the BAN sensor-system is supported by the MobiHealth system?", and "Is real-time transmission of BAN sensor-system samples possible?" We translate these questions to "What is the required throughput of a MobiHealth transport system to support the BANip PDU rate obtained from the end-user requirements?" Firstly, a SDU rate analysis is performed to obtain the required input information to answer the throughput question.

*BANip confirmed service scenario*

The supported SDU (transmission) rate by the BANip (service) depends on the SDU assembly time Ta:

if $Ta \leq Ts + 2*Tp$,
then SDU rate = $1 / (Ts + 2*Tp)$,
else SDU rate = $1 / Ta$

The SDU rate is a function of the BANip PDU transmission time, therefore a function of the BANip PDU size. The red line in Figure 7.7 represents the SDU rate perceived by the MobiHealth system end-user as a function of the BANip PDU size for a confirmed service scenario. Region 1 corresponds to a SDU rate equal to $1/(Ts + 2*Tp)$ and region 2 to a SDU rate equal to $1/Ta$. The horizontal axis represents the BANip PDU size (Bytes) and the left vertical axis the corresponding SDU rate (SDUs/sec). In addition, the blue line in Figure 7.7 represents the perceived SDU goodput (at the BANip SAP) as a function of the BANip PDU size; goodput is a product of SDU rate * BANip PDU size. The right vertical axis presents the corresponding SDU goodput in Kbps.

We conclude from Figure 7.7, that there is a "knee" at BANip PDU size of 612 Bytes. The SDU rate and goodput for this "knee" are respectively 3,9 SDUs/sec and 19 Kbps. For region 2, the SDU rate and goodput stabilizes respectively at 1 SDU/sec and 19 Kbps. In addition, for MobiHealth BANip PDU size of 1834 Bytes, the SDU rate is 1,3 SDUs/sec and the corresponding goodput is 19 Kbps.

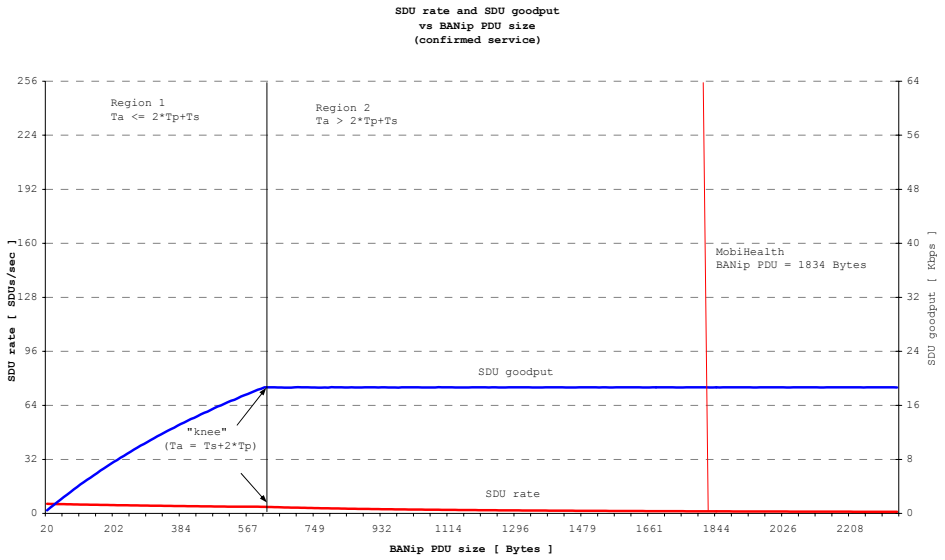



Figure 7.7 SDU rate and SDU goodput versus BANip PDU size for confirmed service

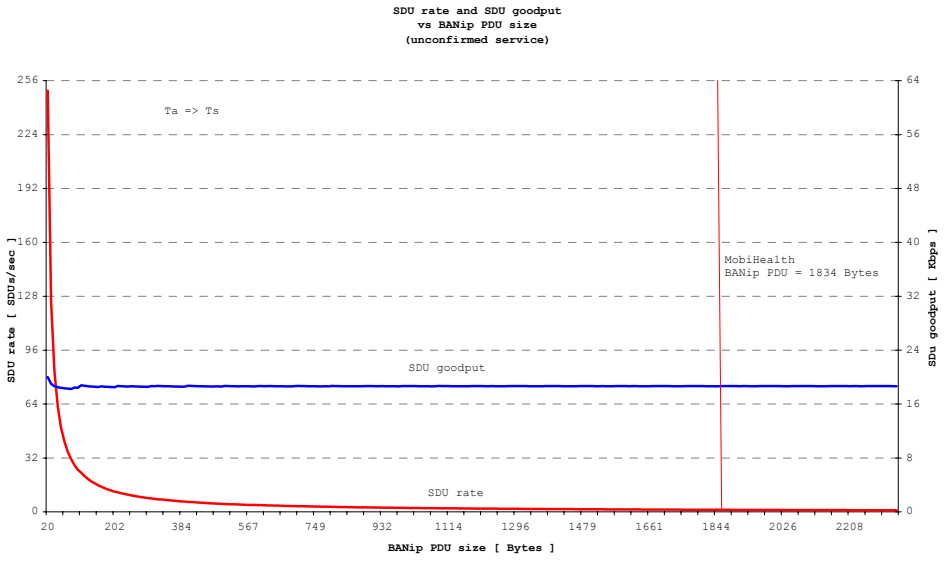



Figure 7.8 SDU rate and SDU goodput versus BANip PDU size for unconfirmed service

*BANip unconfirmed service scenario*

Similar to the BANip confirmed service scenario, the supported SDU (transmission) rate for the unconfirmed service scenario depends on the SDU assembly time Ta:

if Ta ≤ Ts,
then SDU rate = 1 / Ts,
else SDU rate = 1 / Ta

Figure 7.8 presents the SDU rate and the SDU goodput perceived by the MobiHealth system end-user as a function of the BANip PDU size for an unconfirmed service scenario. The horizontal axis represents the BANip PDU size (Bytes) and the left vertical axis the corresponding SDU rate (SDUs/sec). In addition, the right vertical axis presents the corresponding SDU goodput (at the BANip SAP) in Kbps.

We conclude from Figure 7.8, that the SDU rate is inversely proportional to Ta (and therefore BANip PDU size), and it decreases from 256 to 1 SDU/sec. Simultaneously, the corresponding goodput decreases from approximately 20 Kbps to 19 Kbps. In addition, for MobiHealth BANip PDU size of 1834 Bytes, the SDU rate is 1,3 SDUs/sec and the corresponding goodput is 19 Kbps.

Transport system throughput

The SDU rate analysis performed in the previous paragraphs provides the required input information to answer the question: “What is the required throughput of a MobiHealth transport system to support the BANip PDU rate obtained from the end-user requirements?”

The required throughput is determined by the MobiHealth SDU rate, and the encapsulation of the SDUs by the BANip and the lower levels protocols. Figure 7.9 presents the BANip protocol stack used in the MobiHealth system. Transporting SDUs from the MBU to the BEsys requires the use of five (lower level) protocols before they are actually transmitted by a UMTS (datalink layer service) based transport system. The lower levels protocols add additional data (protocol overhead) to a SDU. For example, a 10 Byte SDU becomes a 75 Byte PPP PDU[75] at the datalink layer SAP of the transport system. A SDU rate of 256, represent a 21 Kbps bit rate at the BANip (unconfirmed service) SAP, but at the datalink layer SAP the bit rate equals 154 Kbps (!).

[75] (PPP + IP + TCP + HTTPchunking + BANip) overhead + SDU = 8 + 20 + 20 + 7 + 10 + 10 = 75 Bytes

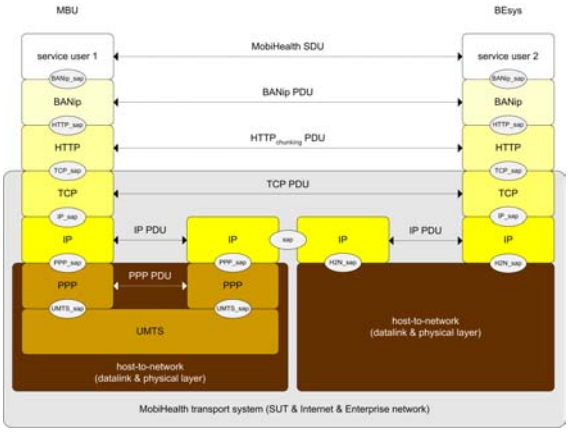



Figure 7.9 BANip protocol stack

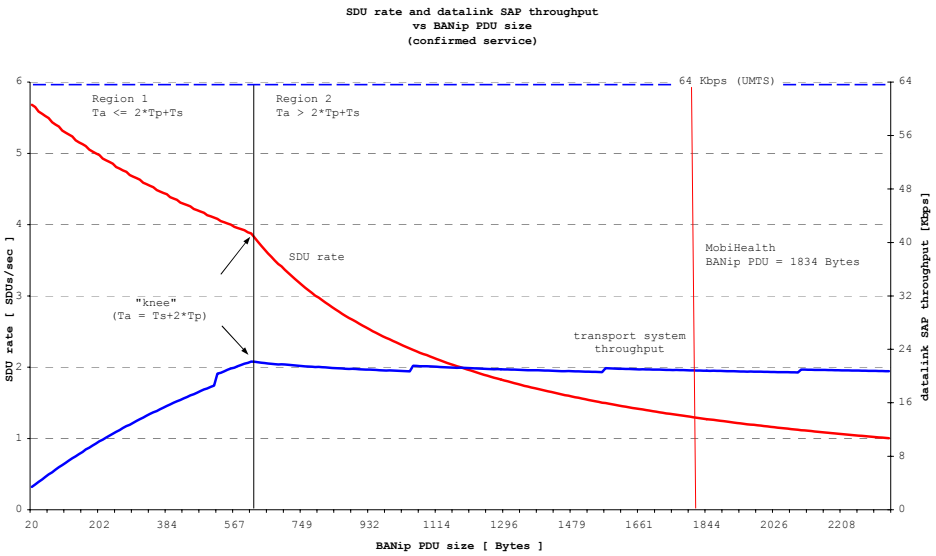



Figure 7.10 SDU rate & DL_SAP throughput vs. BANip PDU size for confirmed service

*BANip confirmed service scenario*

We derive Figure 7.10 from Figure 7.7. The red line in Figure 7.10 represents the SDU rate perceived by the MobiHealth system end-user as a function of the BANip PDU size for a confirmed service scenario. The horizontal axis represents the BANip PDU size (Bytes) and the left vertical axis the corresponding SDU rate (SDUs/sec). In addition, the blue line in Figure 7.10 represents the required throughput of the transport system's datalink layer SAP (right vertical axis) as a function of the BANip PDU size. The black vertical line indicates the regions we identified in SDU rate analysis. The blue dashed (horizontal) line indicates the nominal capacity of a UMTS-based transport system.

We conclude from Figure 7.10, that the maximum supported SDU rate by a confirmed service based BANip equals 5.7. The throughput required at the transport system's datalink layer SAP varies from 3 Kbps up to 21 Kbps. In this case, a UMTS-based transport system may perfectly support the SDU rates up to 5.7. In contrast, the MobiHealth system requires a transport system's datalink layer SAP throughput of 21 Kbps, which is also supported by a UMTS-based transport system.

*BANip unconfirmed service scenario*

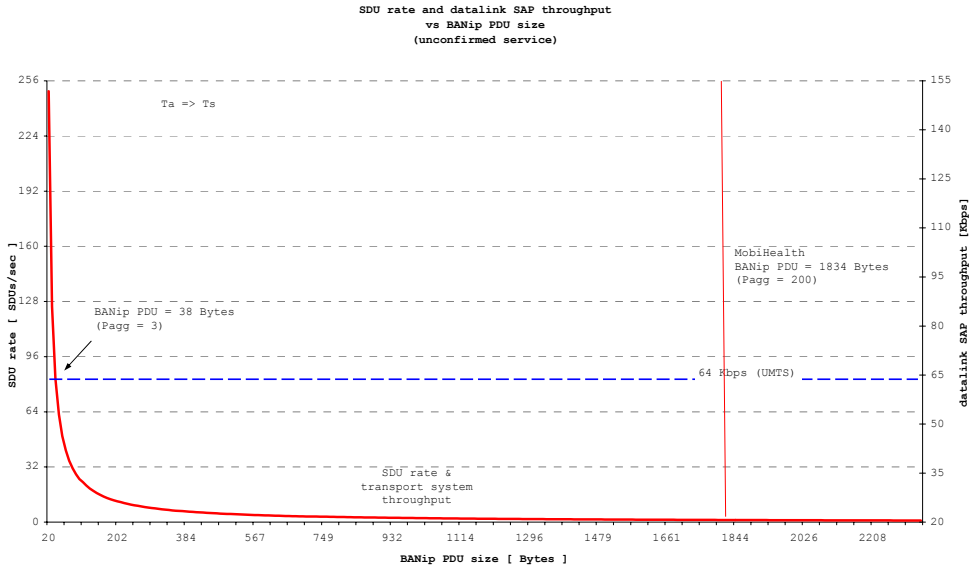



Figure 7.11 SDU rate & DL_SAP throughput vs. BANip PDU size for unconfirmed service

We derive Figure 7.11 from Figure 7.8. The red line in Figure 7.10 represents the SDU rate perceived by the MobiHealth system end-user as a function of the BANip PDU size for a unconfirmed service scenario. The horizontal axis represents the BANip PDU size (Bytes) and the left vertical axis the corresponding SDU rate (SDUs/sec). In addition, the same red line in Figure 7.11 represents the required throughput at the transport system's datalink layer SAP (right vertical axis) as a function of the BANip PDU size. The blue dashed (horizontal) line indicates the nominal capacity of a UMTS-based transport system.

We conclude from Figure 7.11, that the throughput required at the datalink layer SAP of the transport service varies from 21 Kbps up to 154 Kbps. A UMTS-based transport system can only support data rates up to 64 Kbps. This corresponds to a minimum BANip PDU size of 38 Bytes and a supported SDU rate of 83.3 (Pagg = 3) while 256 was required! Hence, a UMTS-based transport system cannot support real-time transport of BAN sensor-system samples.

On September 23, 2004, we executed experiments with version 3.1.1 of the MobiHealth system. We changed the aggregation number (Pagg) of the BAN sensor-system samples at the MBU and observed the MobiHealth system behaviour; i.e. is the system continuing to function properly. The experiment was conducted for Pagg values of: 50, 25, 10, 5 and 1. The BEsys successfully received the data, and we viewed it (real-time) at the end-user application (PL2). The experiment failed however for the Pagg values 5 and 1. We believe that the MBU was suffering from timing problems (Pagg = 5) or the SDU rate (Pagg = 1) offered at the SUT datalink layer SAP of the transport system was simply too high.

In spite of changing Pagg however, at the PL2 end-user application we did not see any timing improvement. We discovered that the timing characteristics of the data communication process between the BEsys (in particular the BSS component) and the PL2 application is completely independent from the MBU to BEsys data communication.

In order to obtain proof, the data communication between the BEsys and the PL2 application must be changed to reflect the changes. Alternatively, the data communication from the MBU to the BEsys can be analysed to discover changes in the BANip packet inter arrival times when using smaller Pagg values then the current 200.

Conclusions

We conclude that an unconfirmed service type of BANip application protocol is the best choice to support MobiHealth SDU rates up to 83 SDUs/sec. The MobiHealth BAN sensor-system (Mobi4 3e1as) is configured for 256 samples per second. Real-time transport of sensor-system data (i.e. SDU) for this high sampling frequency is not supported by a UMTS based transport system (e.g. V3GNL) because of its limited nominal capacity of 64 Kbps.

The optimal BANip PDU size for the maximum supported SDU rate of 83 SDUs/sec (Pagg = 3) is 38 Bytes. In addition, the end-user experiences a SDU delay of an estimated 105 ms (Ta = 12, Tp = 86, Ts = 0,138 * 45 Bytes). In contrast, the BANip PDU size of the current MobiHealth system is 1834 Bytes. The corresponding SDU rate and SDU delay are 1.3 SDUs/sec and 1123 ms.

Note: We used the following assumptions to derive the conclusions: The execution time is zero for all other activities related to the transmission of a MobiHealth SDU (e.g. BANip PDU or HTTP PDU assembly and disassembly). In addition, the assembly process of a new SDU is executed in parallel to the transmission of the previous assembled SDU. Furthermore, the sample aggregation number (Pagg) = 3, gross SDU size = 20 Bytes, net SDU size = 10 Bytes (deflation factor 52%) and the total protocol overhead = 65 Bytes.

## 7.3 Future work

This thesis describes the performance evaluation activity of a transport system supporting the MobiHealth BANip application protocol. We made some assumptions to limit the scope of the activity. For example, we considered only Vodafone's (pre)commercial UMTS network in the Netherlands as an alternative transport system for the MobiHealth system. In addition, we studied only the reliable transport service delivery and the speed-related performance criteria of the system. Every single assumption can therefore be seen as an area of future work. However, we only indicate the most important areas (from our perspective) in this section.

Delay and jitter behaviour

We observed in section 7.1 that the SUT (i.e. V3GNL) has in general high delay variations (i.e. jitter), and indicated the UMTS bearer assignment within the SUT as one of the possible causes. We recommend further study on the delay and jitter calculations performed by the elementary statistical application (achievement 3). This work must aim to neutralise the effect of the bearer assignments in statistical calculations of the delay. Moreover, we recommend further investigation of possible jitter sources by means of deploying measurements probes inside the SUT; effectively decomposing the SUT into sub-systems.

Measurements and evaluation under realistic workload environment

We were the only users of the SUT's UMTS sub-system during our performance evaluation of the SUT. Therefore, this performance evaluation is considered as a benchmark for analysis of alternative transport systems. We recommend a repetition of the performance evaluation activities for alternative UMTS based transport systems that contain background traffic (i.e. other users). These evaluations can be used to study the influence (possible degradation) of other network users on the performance characteristics experienced by a single user.

Handling performance characteristics dynamics

We developed an evaluation methodology to evaluate statically the performance characteristics of the transport system. We introduced a 20 x 20 matrix containing (400) values for the 'size' workload parameter to measure performance parameters of the transport system's uplink and downlink channels (chapter 4). We filled the cells of this matrix with measurements for speed-related performance parameters: delay, jitter and goodput.

We want to make the measurements and performance evaluation dynamic, i.e. we want to implement cyclic repetition of the measurements and evaluation activities. In general, the

performance of commercial UMTS (or GPRS) based transport systems may change dynamically. The performance can change due to different numbers of system users, handovers, system (components) upgrades, etc. There is no one absolute set of performance measurements of a particular transport system. Hence, we want to periodically perform measurements and update the values of the performance parameters in the cells of the matrix. The performance evaluation of the transport system must be performed on the most recent content of the matrix.

To elaborate further on the idea of dynamic measurements and evaluation, an operational MobiHealth system must have access to the content of the 20 x 20 matrix. Our idea is that the MobiHealth SDU size can be changed dynamically based on the actual performance characteristics of the underlying transport system. This implies an adjustment of the number of sensor system samples packetised (Pagg parameter) in a SDU. This dynamic approach aims at the situation where an operational MobiHealth system benefits from the underlying transport system's dynamic performance characteristics.

To deal with the dynamicity of the transport system characteristics, we introduce entities responsible for two functions: 1) holding of the most current values of the performance parameters and 2) upgrading these values in a timely manner.

The 20 x 20 matrix mentioned in the previous paragraphs is denoted as a Transport Service Performance Object (TSPO). We consider this matrix as an object due to the (possible) use in a future version of the MobiHealth system. Similarly, to the matrix, the TSPO object is determined by 20 different 'size' workload parameters in the uplink and 20 sizes in downlink transport system's direction. In this matrix, every cell holds the respective transport system's speed-related performance parameters (delay, jitter and goodput). Each cell in the TSPO object holds besides the performance parameters also the accuracy- and dependability-related results. We apply the ITU 3 x 3 matrix (chapter 4) to each cell of the TSPO object and denote it as a Performance Criteria Object (PCO). Figures 7.12 and 7.13 present the TSPO object and the PCO object respectively.

| **Downlink size** / **Uplink size** | **174** | **⋮** | **48208** |
|---|---|---|---|
| **174** | PCO | ... | PCO |
| **…** | ... | ... | ... |
| **8122** | PCO | ... | PCO |

Figure 7.12 Transport Service Performance Object

| **performance criterion** / **function** | **speed (S)** | **accuracy (A)** | **dependability (D)** |
|---|---|---|---|
| **access (A)** | A-S PPT | A-A PPT | A-D PPT |
| **user information transfer (UIT)** | **UIT-S PPT** | UIT-A PPT | UIT-D PPT |
| **disengagement (D)** | D-S PPT | D-A PPT | D-D PPT |

Figure 7.13 Performance Criteria Object based on the 3 x 3 matrix approach

There are three performance criteria in the PCO object: speed, accuracy and dependability. Each performance criterion can be applied to the following function: access, user-information transfer and disengagement. Therefore, there are nine cells in the PCO objects. Each cell represents a Performance Parameters Table (PPT), holding the corresponding values of the performance parameters. Our performance evaluation methodology covers only one cell in the PCO object. We focussed on the speed-related performance characteristics of the SUT for the user-information function.

Figure 7.14 presents the corresponding UIT-S PPT table (i.e. Performance Parameters Table for the Speed performance criterion and the User Information Transfer function), which holds the measured values of performance parameters: delay, jitter and goodput. We indicate that future work is needed on the validity of the performance parameters of the other (eight) PPTs.

We finish our discussion on future work areas proceeding from our work presented in this thesis. However, we are fully aware that our work just covered the proverbial “top of the iceberg”.

| **Performance parameter** | **delay** [ms] | **jitter** [%] | **goodput** [Kbps] |
|---|---|---|---|
| **Value** | 120 | 10 | 35 |

Figure 7.14 UIT-S Performance Parameters Table

# 8 References

[BoV2001] Book of Visions, Wireless World Research Forum, 2001, WWRF

[Bult2004] R. Bults, K.Wac, A.Van Halteren, D. Konstantas, V. Jones, I. Widya, "Body Area Networks for Ambulant Patient Monitoring Over Next Generation Public Wireless Networks" 3rd IST Mobile and Wireless Communications Summit 2004 27-30 June, Lyon, France

[Doko2003] Dokovsky N., Aart van Halteren, Ing Widya, "BANip: enabling remote healthcare monitoring with Body Area Networks", Third International Workshop, FIDJI 2003, Luxembourg-Kirchberg, Luxembourg, November 27-28, 2003, Series: Lecture Notes in Computer Science, Vol. 2952 Guelfi, Nicoals; Astesiano, Egidio; Reggio, Gianna (Eds.) 2004, ISBN: 3-540-21091-1

[Eyse2001] Eysenbach G., What is E-health? Journal of Medical Internet Research 2001; June 18; 3(2): e20, JMIR 2001

[Hoek1997] Hoeksema F.W., van der Veen J.T., van Beijnum B.J., *Measurements, A Methodical Approach to Performance Measurement, Experiments: Measure and Measurement Specification*, University of Twente, CTIT, July 1997

[Have1998] Haverkort B.R, Performance of Computer Communication Systems, A Model-Based Approach, John Wiley & Sons, England, 1998

[HTTPfaq] HTTP 1.1 chunking, http chunking

[ITU1993] ITU-T Recommendation I.350, *General aspects of Quality of Service and Network Performance in Digital Networks, including ISDNs*, March 1993

[Jain1991] Raj Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modelling, John Wiley & Sons, New York, 1991

[Jova2003] E. Jovanov, et al, "Stress Monitoring Using a Distributed Wireless Intelligent Sensor System", in *IEEE Engineering in Medicine and Biology Magazine*, May/June 2003

[Jone2001] Val Jones, Richard Bults, Dimitri Konstantas, Pieter AM Vierhout, Healthcare PANs: Personal Area Networks for trauma care and home care, Proceedings Fourth International Symposium on Wireless Personal Multimedia Communications (WPMC), Sept. 9–12, 2001, Aalborg, Denmark

[Knop2003] Knoppel D.F, *Performance improvement of a Body Area Network interconnect protocol (master thesis)*, University of Twente, EWI-APS, November 2003

[Kons2004] Dimitri Konstantas, Aart Van Halteren, Richard Bults, Katarzyna Wac, Ing Widya, Nicolai Dokovsky, George Koprinkov, Val Jones and Rainer Herzog, ``Mobile Patient Monitoring: the MobiHealth System," proceedings of *International Congress on Medical and Care Compunetics,* NCC, The Hague, June 2-4, 2004

[Kuro2001] J.F. Kurose and K.W. Ross, Computer Networking, A Top-Down Approach Featuring the Internet, Addison Wesley Longman Inc., 2001

[Klei1976] L. Kleinrock, Queuing Systems, Vol. 2, Computer Applications, John Wiley, NY, 1976

[MobiHeal] The MobiHealth Project (IST-2001-36006)

[Naug1996] P. Naughton, The Java handbook, Osborne McGraw-Hill. Berkeley, California, USA, 1996

[RFC793] TRANSMISSION CONTROL PROTOCOL, RFC 793

[RFC2616] Hypertext Transfer Protocol - HTTP/1.1, RFC 2616

[SunWeb] Sun Microsystems

[Tardis] HC Mingham-Smith Ltd., Tardis

[TechWeb] TechWeb, TechEncyclopedia

[TPILB] “This Page Intentionally Left Blank”-Project, TPILB-Project

[Tracert] MicroSoft Windows XP Professional Product Documentation, Tracert

[Viss2000] Vissers C.A., Ferreira Pires L., Quartel D.A.C., van Sinderen M.J., *The architectural design of distributed systems*, University of Twente, Nov. 2000

[Webster] Merriam-Webster's Online Dictionary, dictionary

[WinWeb] WinGuides Network for Windows, NTP Time Server

[YIPChi] YIP Chi Lap [Beta], Ph.D., Dep. of Computer Science, The University of Hong Kong, System clock resolution for different operating systems

[ZigBee] ZigBee Alliance, “IEEE 802.15.4, ZigBee standard”

# Performance evaluation of a Transport System supporting the MobiHealth BANip: Methodology and Assessment

# APPENDIXES

# 9 Appendixes

## Appendix A Ethereal dump of HTTP 1.1 PDU

```
Frame 15 (590 bytes on wire, 590 bytes captured)
Ethernet II, Src: 02:e0:78:07:8c:41, Dst: 00:0c:6e:82:06:59
Internet Protocol, Src Addr: 62.140.137.30 (62.140.137.30), Dst Addr:
130.89.144.130 (130.89.144.130)
Transmission Control Protocol, Src Port: 11975 (11975), Dst Port: 32858 (32858),
Seq: 2507588779, Ack: 3186703796, Len: 524
Data (524 bytes)

0000  00 0c 6e 82 06 59 02 e0 78 07 8c 41 08 00 45 00   ..n..Y..x..A..E.
0010  02 40 5a 75 40 00 2c 06 17 bd 3e 8c 89 1e 82 59   .@Zu@.,...>....Y
0020  90 82 2e c7 80 5a 95 76 c4 ab bd f1 3d b4 80 10   .....Z.v....=...
0030  0c 48 88 f6 00 00 01 01 08 0a 00 03 49 6b 00 03   .H..........Ik..
0040  d3 49 37 38 39 0d 0a 00 01 00 00 00 01 00 00 07   .I789...........
0050  7f 78 da dd d7 f9 73 55 e5 19 07 f0 c7 4b 08 31   .x....sU.....K.1
0060  84 00 01 02 04 08 67 b9 27 1b 49 b8 49 6e 62 1a   ......g.'.I.Inb.
0070  6e a3 ad 76 a0 c6 c2 74 8a 2d 48 80 18 50 59 ac   n..v...t.-H..PY.
0080  0d 75 19 46 71 86 29 53 ea d4 49 5d 50 0b 6d 29   .u.Fq.)S..I]P.m)
0090  d8 b2 28 a5 94 4d a6 04 11 51 08 a1 81 24 ac 36   ..(..M...Q...$.6
00a0  e4 06 0a 14 2c ce 60 63 69 19 40 e1 db f7 7c cf   ....,.`ci.@...|.
00b0  73 19 fa 2f 34 3f 7e e6 79 de fb 9e 73 be ef 92   s../4?~.y...s...
00c0  16 19 f4 7f f2 27 95 36 76 b6 89 c8 e2 e1 1d 17   .....'.6v.......
00d0  eb 7f 78 af 48 44 3c 07 2b db 7d 0a 29 95 ca 7e   ..x.HD<.+.}.)..~
00e0  1b ef 91 b2 94 a2 32 c8 c6 22 36 7e da 1e 50 85   ......2.."6~..P.
00f0  cc b0 51 4a fa 83 52 a5 5c 75 b0 a1 ed ce c6 2a   ..QJ..R.\u.....*
0100  59 e5 60 39 c9 33 f4 a4 a1 98 4c b2 d1 8b c3 7f   Y.`9.3....L.....
0110  d9 1e 50 b5 cc b1 31 91 74 55 49 24 64 63 1a 69   ..P...1.tUI$dc.i
0120  93 52 48 ea 2d bc 4c 7a 5c 29 49 8a 6d f4 90 1a   .RH.-.Lz\)I.m...
0130  95 92 65 a5 83 26 d2 6e 43 f3 0d a5 c8 08 07 0f   ..e..&.nC.......
0140  73 12 4b 94 52 e5 63 0b 1e e9 05 a5 34 b9 69 63   s.K.R.c.....4.ic
0150  34 1b 17 28 a5 cb 3b 0e 8e b3 ea 47 4a 03 e4 b0   4..(..;....GJ...
0160  8d 9f 92 d6 b5 05 94 21 df b3 71 37 1b 4f 2b 0d   .......!..q7.O+.
0170  96 23 36 ee 22 f5 18 9a 67 28 53 fe 66 a3 8a 8d   .#6."...g(S.f...
0180  7f 56 1a 26 c7 6c fc 36 78 ab 4a 59 d2 c7 c1 f3   .V.&.l.6x.JY....
0190  1d 3e 9d 51 1a 29 cd 16 2e 07 9f c3 4c 62 ae a1   .>.Q.).......Lb..
01a0  6c a9 b1 f4 ad 56 2b 59 b2 ce c6 12 d2 40 25 47   l....V+Y.....@%G
01b0  62 0e 26 b3 f1 df 66 ac 39 86 c2 72 cd 46 06 e9   b.&...f.9..r.F..
01c0  9c 52 8e ac b7 b1 8f 8d ed 4a 79 72 c0 c6 20 56   .R.......Jyr.. V
01d0  ad 55 2a f0 3f c7 ad b6 04 3d 61 a8 50 86 da 28   .U*.?....=a.P..(
01e0  66 55 91 52 b1 7c ee c0 23 ad 52 8a c8 6f 1c 7c   fU.R.|..#.R..o.|
01f0  ce c6 64 33 af c7 98 af 37 2d 6c 23 ed 6d 0b 28   ..d3....7-l#.m.(
0200  2a ad 16 2e 92 7e af 54 21 17 6c 0d e6 15 43 b3   *....~.T!.l...C.
0210  99 af d5 26 85 a4 66 a5 2a 59 66 61 21 e9 a4 52   ...&..f.*Yfa!..R
0220  4c 9e b1 50 d3 91 78 a0 d9 cc 57 b9 85 4e 56 1d   L..P..x...W..NV.
0230  35 34 8b f9 9a 68 61 52 10 13 a5 90 94 d8 b8 4a   54...haR.......J
0240  fa 85 52 92 5c b3 b0 98 63 bd 66 a8 9e f9         ..R.\...c.f...

Frame 17 (590 bytes on wire, 590 bytes captured)
Ethernet II, Src: 02:e0:78:07:8c:41, Dst: 00:0c:6e:82:06:59
Internet Protocol, Src Addr: 62.140.137.30 (62.140.137.30), Dst Addr:
130.89.144.130 (130.89.144.130)
Transmission Control Protocol, Src Port: 11975 (11975), Dst Port: 32858 (32858),
Seq: 2507589303, Ack: 3186703796, Len: 524
Data (524 bytes)

Frame 19 (590 bytes on wire, 590 bytes captured)
Ethernet II, Src: 02:e0:78:07:8c:41, Dst: 00:0c:6e:82:06:59
Internet Protocol, Src Addr: 62.140.137.30 (62.140.137.30), Dst Addr:
130.89.144.130 (130.89.144.130)
Transmission Control Protocol, Src Port: 11975 (11975), Dst Port: 32858 (32858),
Seq: 2507589827, Ack: 3186703796, Len: 524
Data (524 bytes)
```

## Appendix A (cont.) Ethereal dump of HTTP 1.1 PDU

```
Frame 21 (430 bytes on wire, 430 bytes captured)
Ethernet II, Src: 02:e0:78:07:8c:41, Dst: 00:0c:6e:82:06:59
Internet Protocol, Src Addr: 62.140.137.30 (62.140.137.30), Dst Addr:
130.89.144.130 (130.89.144.130)
Transmission Control Protocol, Src Port: 11975 (11975), Dst Port: 32858 (32858),
Seq: 2507590351, Ack: 3186703796, Len: 364
Data (364 bytes)

0000  00 0c 6e 82 06 59 02 e0 78 07 8c 41 08 00 45 00   ..n..Y..x..A..E.
0010  01 a0 5a 78 40 00 2c 06 18 5a 3e 8c 89 1e 82 59   ..Zx@.,..Z>....Y
0020  90 82 2e c7 80 5a 95 76 ca cf bd f1 3d b4 80 18   .....Z.v....=...
0030  0c 48 db 81 00 00 01 01 08 0a 00 03 49 89 00 03   .H..........I...
0040  d3 67 a2 64 ee f7 11 4c e5 fb fa ea 54 40 51 49   .g.d...L....T@QI
0050  19 8b f1 7c ab 05 5d 89 7c 8d 29 c2 29 7e 8e 9a   ...|..].|.).)~..
0060  ae 44 be d6 8e c1 84 7d 3e bd d3 95 c8 d7 dc 31   .D.....}>......1
0070  68 64 00 ee 8b 27 f2 65 62 f2 24 a9 3a 9e d8 bf   hd...'.eb.$.:...
0080  76 e4 62 32 f3 35 30 9e d8 bf 7a 72 b0 91 21 1f   v.b2.50...zr..!.
0090  13 4f ec 5f 7d 73 f1 0a 97 c2 c6 78 e2 7c 74 73   .O._}s.....x.|ts
00a0  50 43 7a 23 9e 38 1f 25 07 37 d8 f8 54 3c 71 3e   PCz#.8.%.7..T<q>
00b0  36 79 48 e5 7a 6c 88 27 ce c7 7b c2 38 4f 6a 8d   6yH.zl.'..{.8Oj.
00c0  07 37 98 34 a9 73 d1 c1 3d e7 86 52 ba a4 86 71   .7.4.s..=..R...q
00d0  89 1b df 88 ee 80 06 c8 ec 30 c2 a4 e4 ee e0 26   .........0.....&
00e0  9a 21 43 5c c4 d8 68 2b 0d 96 2b 0e fe 41 ba 14   .!C\..h+..+..A..
00f0  0f 28 53 ce ba d8 c3 c6 b7 ba 13 f7 fb a3 2e 1e   .(S.............
0100  24 0d 39 9d b8 df 4f 70 d1 cc c6 86 ee c4 fd 7e   $.9...Op.......~
0110  a2 8b 4a 56 ed 56 ca 96 42 17 17 58 55 6f 1a 1b   ..JV.V..B..XUo..
0120  98 af 32 17 bb 48 95 4a 8e b4 b8 68 62 63 b5 52   ..2..H.J...hbc.R
0130  d8 bf 33 65 f1 54 f8 c4 d0 b3 cc 97 39 02 be c9   ..3e.T......9...
0140  aa 66 a5 3c 19 ed e0 0c c7 fa a3 52 81 7c ea e0   .f.<.......R.|..
0150  99 e0 06 a3 54 28 5b 1d fd 9f 6f ad a1 85 cc 57   ....T([...o....W
0160  b3 83 f9 1c eb 8a 52 44 3e 73 b0 98 f4 c0 99 80   ......RD>s......
0170  4a 65 a3 83 8f 48 f7 18 7a 81 f9 ea 71 10 21 3d   Je...H..z...q.!=
0180  a8 54 21 7f 77 b0 99 f4 98 52 a5 cc 74 b1 8e bf   .T!.w....R..t...
0190  b8 51 a9 4a 4e 38 58 c5 aa bd 86 16 31 5f d3 1c   .Q.JN8X.....1_..
01a0  fc 8a b3 5f ad 54 fd 5f db 62 65 2d 0d 0a         ..._.T._.be-..


HTTP encapsulation:
37 38 39     = chunk size in hex notation (e.g. 789x = 1929 bytes)
0d 0a        = CRLF

BANIP PCI:
00 01        = BANip protocol version (always 1x)
00 00 00 01 = message type (Sensor-Data = 1, Registration = 2,
              RequestSubscription: registration = 9, response = 10
00 00 07 7f = payload length (e.g. 86x)
```

# Appendix B XML description of TMSI Mobi4

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by
 Rende Luitjes (HP) revised by Paul Buijsman 30/10/2003 -->
<MBU xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=
 "http://www.mobihealth.org/schema/BANDescription.xsd" ID="" Name=""
 Manifacturer="MobiHealth" Vendor="UT" Version="$Revision: 1.14 $"
 Model="trauma-3e1as">
 <SensorSet ID="" transcodingChain="Filter;Packet;EDR;Deflate">
  <Device Type="composed" ID="" Config="" Frequency="">
   <Device Type="mobi4" ID="0924030007" Config="00:a0:96:20:d6:7a"
    Frequency="256">
    <Sensor Name="ECG1" ByteLength="3" AmplifierType="1" Unit="uV"
     Resolution="0.0715" Visualization="graph" Min="-6000"
     Max="6000" Highpassfilter="0.8" Active="true"/>
    <Sensor Name="ECG2" ByteLength="3" AmplifierType="1" Unit="uV"
     Resolution="0.0715" Visualization="graph" Min="-6000"
     Max="6000" Highpassfilter="0.8" Active="true"/>
    <Sensor Name="ECG3" ByteLength="3" AmplifierType="1" Unit="uV"
     Resolution="0.0715" Visualization="graph" Min="-6000"
     Max="6000" Highpassfilter="0.8" Active="true"/>
    <Sensor Name="Resp" ByteLength="3" AmplifierType="3" Unit="uV"
     Resolution="1.4305" Visualization="graph" Min="-10000"
     Max="10000" Highpassfilter="5" Active="true"/>
    <Sensor Name="SaO2" ByteLength="1" AmplifierType="11" Unit="%"
     Resolution="1" Visualization="numeric" Min="90" Max="100"
     Highpassfilter="0" Active="true"/>
    <Sensor Name="Pleth" ByteLength="1" AmplifierType="11" Unit=""
     Resolution="1" Visualization="graph" Min="0" Max="255"
     Highpassfilter="0" Active="true"/>
    <Sensor Name="HeartRate" ByteLength="1" AmplifierType="11"
     Unit="BPM" Resolution="1" Visualization="numeric" Min="30"
     Max="150" Highpassfilter="0" Active="true"/>
    <Sensor Name="SensorStatus" ByteLength="1" AmplifierType="11"
     Unit="" Resolution="1" Visualization="numeric" Min="0"
     Max="255" Highpassfilter="0" Active="true"/>
    <Sensor Name="Marker" ByteLength="1" AmplifierType="9" Unit=""
     Resolution="1" Visualization="numeric" Min="0" Max="1"
     Highpassfilter="0" Active="true"/>
    <Sensor Name="SawTooth" ByteLength="1" AmplifierType="10"
     Unit="" Resolution="1" Visualization="graph" Min="0"
     Max="65" Highpassfilter="0" Active="true"/>
   </Device>
   <!-- The sum of the ByteLenghts of the Sensors shoud be <255 -->
   <Device Type="manualinput.trauma" ID="manual_trauma"
    Frequency="0" Path="">
      <Device Type="manualinput.trauma" ID="manual_trauma" Frequency="0"
       Path="">
      <Sensor Name="BP SYS" ByteLength="1" AmplifierType="7" Unit="mmHg"
       Resolution="1" Active="true"/>
      <Sensor Name="BP DIA" ByteLength="1" AmplifierType="7" Unit="mmHg"
       Resolution="1" Active="true"/>
      <Sensor Name="Fluid Type" ByteLength="30" AmplifierType="138"
       Unit="" Resolution="1" Active="true"/>
      <Sensor Name="Fluid Amount" ByteLength="4" AmplifierType="3"
       Unit="ml" Resolution="1" Active="true"/>
      <Sensor Name="Left Pupil Reaction" ByteLength="1" AmplifierType="4"
       Unit="0/1" Resolution="1" Active="true"/>
      <Sensor Name="Right Pupil Reaction" ByteLength="1" AmplifierType="4"
       Unit="0/1" Resolution="1" Active="true"/>
      <Sensor Name="Left Pupil Size" ByteLength="30" AmplifierType="138"
       Unit="" Resolution="1" Active="true"/>
      <Sensor Name="Right Pupil Size" ByteLength="30" AmplifierType="138"
       Unit="" Resolution="1" Active="true"/>
```

## Appendix B (cont.) XML description of TMSI Mobi4

```
      <Sensor Name="Injury Type" ByteLength="30" AmplifierType="138"
       Unit="" Resolution="1" Active="true"/>
      <Sensor Name="Text" ByteLength="40" AmplifierType="138" Unit=""
       Resolution="1" Active="true"/>
      <Sensor Name="PatientID" ByteLength="10" AmplifierType="138" Unit=""
       Resolution="1" Active="true"/>
      <Sensor Name="FirstName" ByteLength="30" AmplifierType="138" Unit=""
       Resolution="1" Active="true"/>
      <Sensor Name="LastName" ByteLength="30" AmplifierType="138" Unit=""
       Resolution="1" Active="true"/>
   </Device>
  </Device>
 </SensorSet>
</MBU>
```

# Appendix C DeflationEncoder compression factor calculation

**Ethereal file: 3e1as_sensors_20040507**

| **timestamp** | **BANip PDU** | **MH SDU** |
|---|---|---|
| 0,54 | 1.929 | 1.919 |
| 1,46 | 1.914 | 1.904 |
| 2,34 | 1.902 | 1.892 |
| 2,82 | 1.913 | 1.903 |
| 3,90 | 1.883 | 1.873 |
| 4,42 | 1.938 | 1.928 |
| 5,48 | 1.924 | 1.914 |
| 5,84 | 1.922 | 1.912 |
| 6,96 | 1.899 | 1.889 |
| 7,84 | 1.931 | 1.921 |
| 8,39 | 1.974 | 1.964 |
| 9,28 | 2.005 | 1.995 |
| 9,84 | 1.945 | 1.935 |
| 10,90 | 2.009 | 1.999 |
| 12,28 | 1.922 | 1.912 |
| 12,92 | 1.937 | 1.927 |
| 13,83 | 1.939 | 1.929 |
| 14,84 | 1.894 | 1.884 |
| 15,34 | 1.905 | 1.895 |
| 16,34 | 1.885 | 1.875 |
| 16,90 | 1.899 | 1.889 |
| 17,76 | 1.896 | 1.886 |
| 18,32 | 1.913 | 1.903 |
| 19,34 | 1.881 | 1.871 |
| 20,32 | 1.902 | 1.892 |
| 21,76 | 1.887 | 1.877 |

| | | | |
|---|---|---|---|
| total time interval | 49.688 | bytes | |
| average MH SDU size | 1.801 | bytes | (normalized for 1 sec) |

| | | |
|---|---|---|
| Aggregate MH SDU per time interval of 20 sec | 97.280 | bytes (gross) |
| BANip PCI overhead | 256 | bytes |
| | 97.536 | bytes (gross) |

| | | |
|---|---|---|
| Ethereal: MH SDU bytes per EDR time interval of 20 sec. | 46.831 | bytes (deflated) |
| EDR: MH SDU bytes per EDR time interval of 20 sec. | 97.280 | bytes (gross) |
| **average deflation factor** | **51,86** | **%** |

# Appendix D System Parameters of Influence

| No | Operating System | CPU | RAM |
|---|---|---|---|
| notebook (nb) | Microsoft Windows XP Professional, Version 2002, SP1 | Pentium III, mobile CPU 1GHz | 640 MB |
| pc1 | Microsoft Windows XP Professional, Version 2002, SP1 | Pentium 4, 2.4 GHz | 512 MB |
| pc2 | Microsoft Windows XP Professional, Version 2002, SP1 | Pentium III, 600 MHz | 128 MB |
| iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13 | | 64 MB |

Table D.1 Computer systems

# Appendix E V3GNL Leased line infrastructure

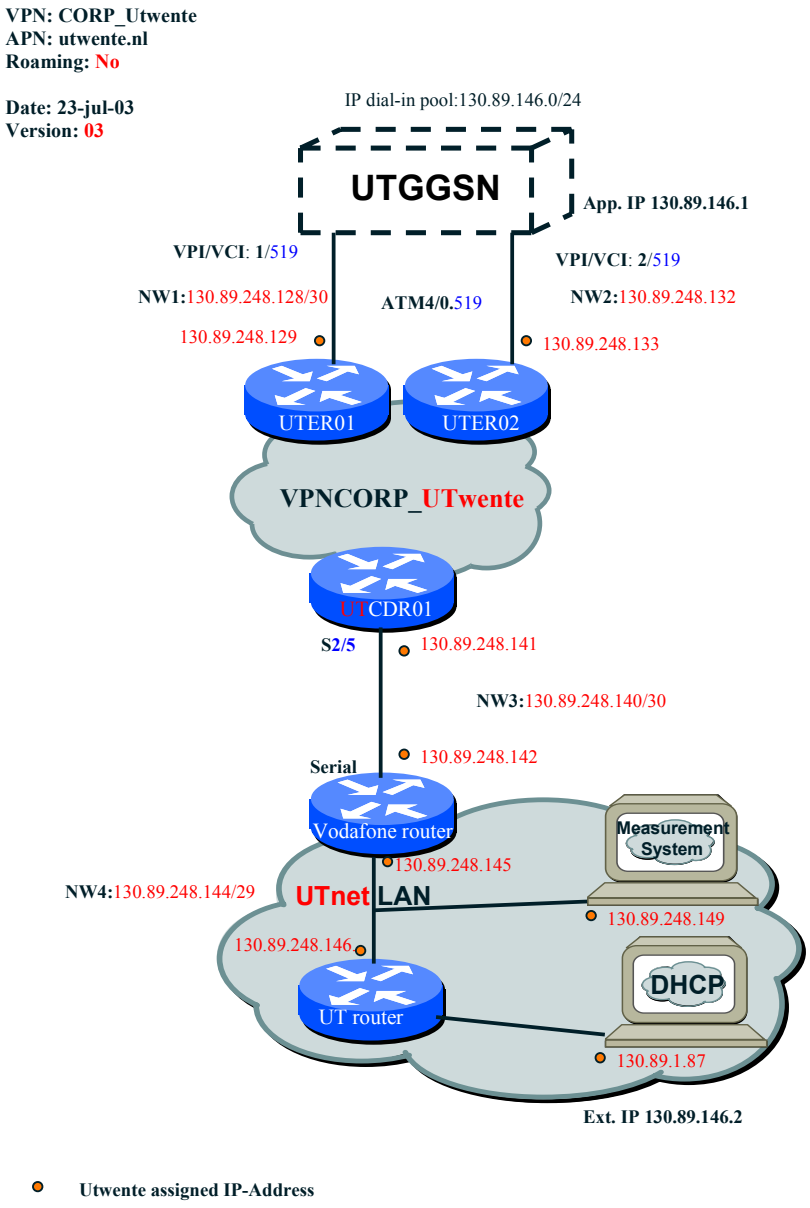



Figure E.1 V3GNL leased line to UTnet

## Appendix F UML class diagram notation

The notation is taken from the following online tutorial: http://www.parlezuml.com/tutorials/java/class/index_files/frame.htm. The notation elements used in Java class diagram are given below.

1) Class is represented by a rectangle with one or more compartments. The topmost (and always mandatory) compartment contains the name of the class.

2) Attributes i.e. variables declared within a class

   [visibility] [/] attribute_name [multiplicity] [: type [= default_value]]

3) Operations i.e. functions or sub procedures in Java

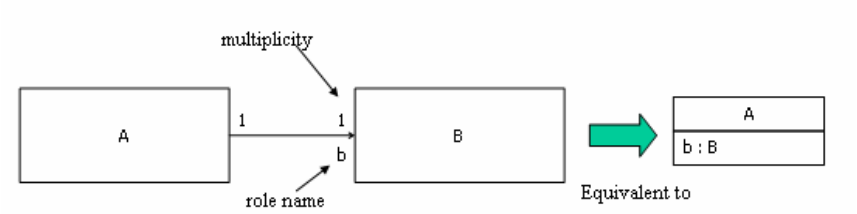
   [visibility] op_name ( [ [in|out] parameter : type [, more params] ] ) [: return_type]

4) Visibility

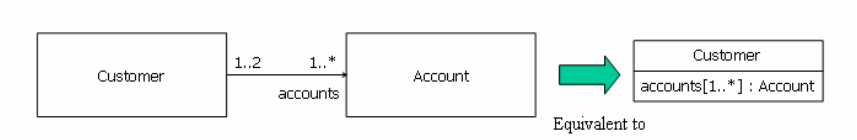
| | | | |
|---|---|---|---|
| + : | public | - : | private |
| # : | protected | ~ : | package |

5) Scope
   class scope (i.e. static) members are underlined

6) Association relation

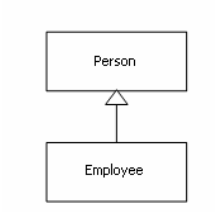



7) Multiplicity and collections

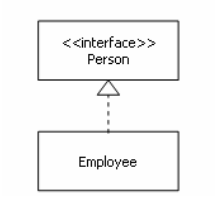


## Appendix F (cont.) UML class diagram notation

8) Generalization i.e. inheritance




9) Realization i.e. implementation of the interface of another class



## Appendix G Java class diagram refinement

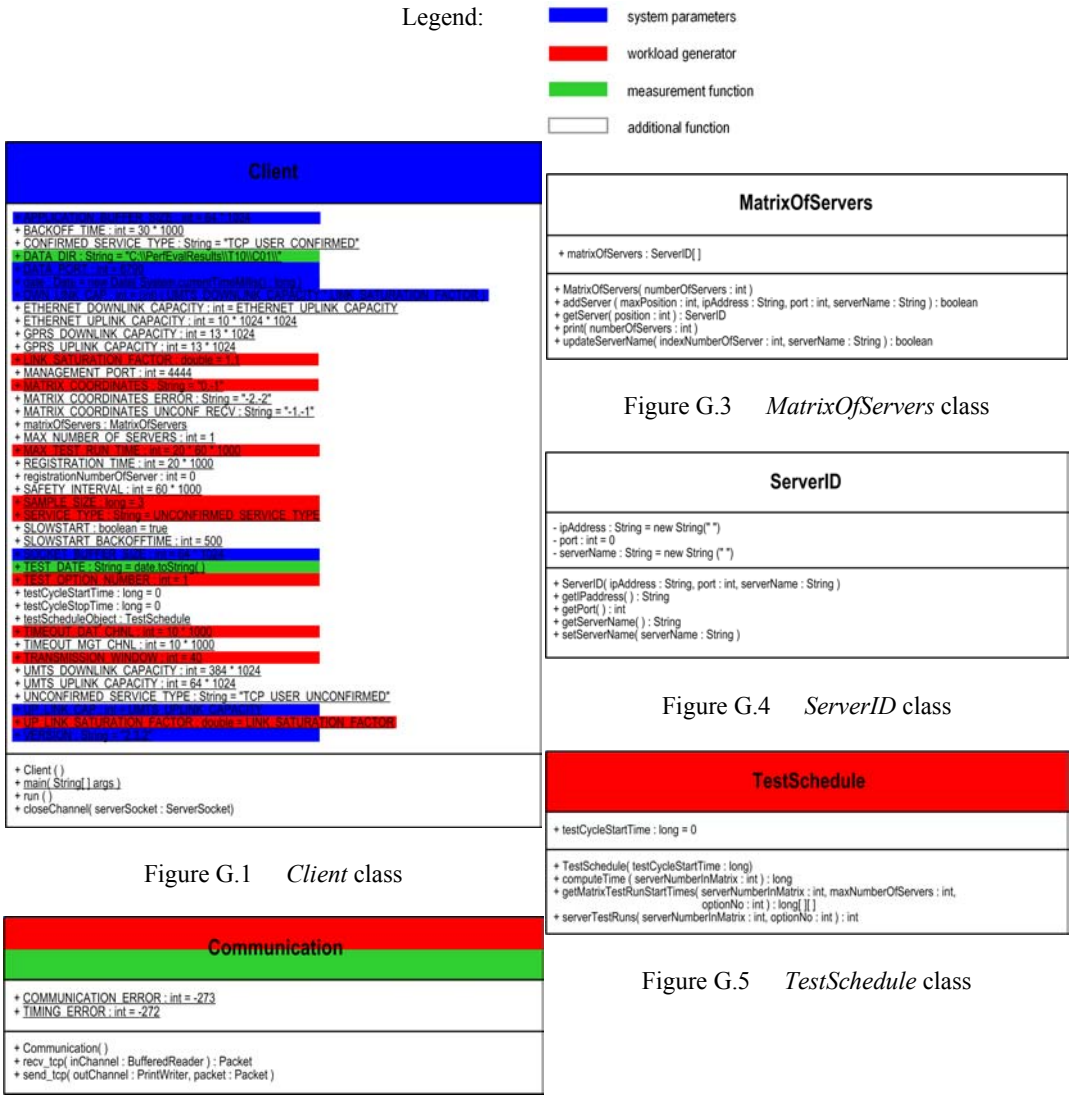



Figure G.1 *Client* class

Figure G.2 *Communication* class

Figure G.3 *MatrixOfServers* class

Figure G.4 *ServerID* class

Figure G.5 *TestSchedule* class

## Appendix G (cont.) Java class diagram refinement


Figure G.6 *ServerControl* class


Figure G.7 *ConfirmedServiceTest* class

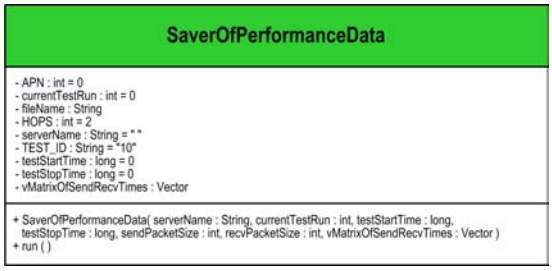



Figure G.8 *SaverOfPerformanceData* class

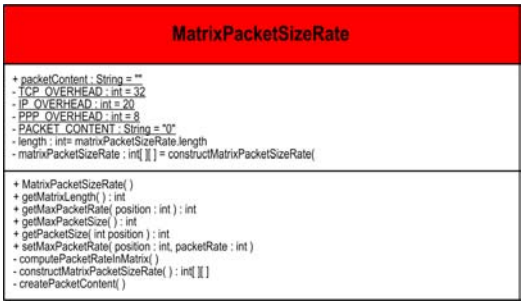



Figure G.9 *MatrixPacketSizeRate* class

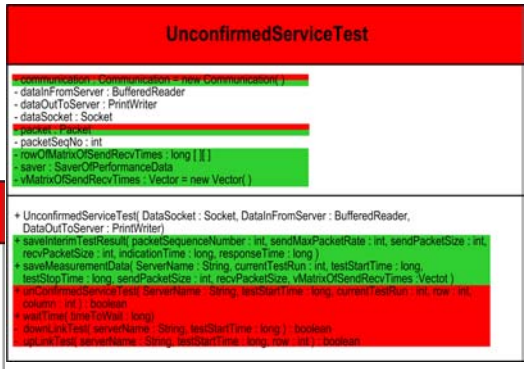



Figure G.10 *UnconfirmedServiceTest* class


Figure G.11 *Packet* class

# Appendix H Measurement experiments specification

## Experiment 2

| | | | | |
|---|---|---|---|---|
| TestID | 04 | | | |
| APN | utwente.nl | | | |
| HOPS | Server <> Client | | | |
| Test description | Confirmed Service, 8x8, UMTS-UT/BT, Server/Client, 1 Terminal | | | |
| Extra comms | UMTS | | | |
| Intra comms | Bluetooth 1.0 | | | |
| Clock sync | ntpdate | | 0,01 seconds/test run | ntp.utwente.nl |
| Equipment: | | | | |
| Identifier | name | HW-platform | SW-platform | |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 | |
| ServerId 20 | Enschede-20 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| Terminal | Nokia 8 | Nokia 6650 | PR4 | |

Figure H.1 Specification of the Experiment 2

| **500** | 174 | 1572 | 2096 | 7860 | 8384 | 16244 | 16768 | 48732 |
|---|---|---|---|---|---|---|---|---|
| 174 | 0,0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | NOP |
| 524 | 1,0 | 1 | 2 | 3 | 4 | 5 | NOP | NOP |
| 1048 | 2,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 1572 | 3,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 2096 | 4,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 4192 | 5,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 6288 | 6,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 7860 | 7,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |

Figure H.2 Test run matrix for Experiment 2

## Appendix H (cont.) Measurement experiments specification

### Experiment 3

| | | | | |
|---|---|---|---|---|
| TestID | 05 | | | |
| APN | utwente.nl | | | |
| HOPS | Server <> Client | | | |
| Test description | Confirmed Service, 8x8, UMTS-UT/BT, Server/Client, 5 Terminals in area around $10m^2$ | | | |
| Extra comms | UMTS | | | |
| Intra comms | Bluetooth 1.0 | | | |
| Clock sync | ntpdate | | 0,026 seconds/test run | ntp.utwente.nl |
| Equipment: | | | | |
| Identifier | Name | HW-platform | SW-platform | |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 | |
| ServerId 3 | Enschede-16 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 8 | Enschede-17 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 13 | Enschede-18 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 19 | Enschede-19 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 20 | Enschede-20 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| Terminals | Nokia 1,2,3,8,9 | Nokia 6650 | PR4<br>*Note:Terminal Nokia 1 was swapped by Nokia 4 on February 2, 2004 due to repetitive communication errors in test SEs* | |

Figure H.3 Specification of the Experiment 3

| **500** | 174 | 1572 | 2096 | 7860 | 8384 | 16244 | 16768 | 48732 |
|---|---|---|---|---|---|---|---|---|
| 174 | 0,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 524 | 1,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 1048 | 2,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 1572 | 3,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 2096 | 4,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 4192 | 5,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 6288 | 6,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 7860 | 7,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |

Figure H.4 Test run matrix for Experiment 3

## Appendix H (cont.) Measurement experiments specification

### Experiment 4

| | | | | |
|---|---|---|---|---|
| TestID | 03 | | | |
| APN | utwente.nl | | | |
| HOPS | Server <> Client | | | |
| Test description | Confirmed Service, 8x8, UMTS-UT/BT, Server/Client, 10 Terminals in area around 10m$^2$ | | | |
| Extra comms | UMTS | | | |
| Intra comms | Bluetooth 1.0 | | | |
| Clock sync | Ntpdate | | 0.010 seconds/test run | ntp.utwente.nl |
| Equipment: | | | | |
| Identifier | Name | HW-platform | SW-platform | |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 | |
| ServerId 03 | Enschede-3 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 08 | Enschede-8 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 13 | Enschede-13 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 14 | Enschede-14 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 15 | Enschede-15 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 16 | Enschede-16 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 17 | Enschede-17 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 18 | Enschede-18 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 19 | Enschede-19 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| ServerId 20 | Enschede-20 | iPAQ 3870 | Linux 2.4.19-rmk6-pxa1-hh13, CVM J2ME personal profile 1.0 | |
| Terminals | Nokia 1..10 | Nokia 6650 | PR4 | |

Figure H.5 Specification of the Experiment 4

| **500** | 174 | 1572 | 2096 | 7860 | 8384 | 16244 | 16768 | 48732 |
|---|---|---|---|---|---|---|---|---|
| 174 | 0,0 | 0.1 | 0.2 | 0.3 | 0.4 | NOP | NOP | NOP |
| 524 | 1,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 1048 | 2,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 1572 | 3,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 2096 | 4,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 4192 | 5,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 6288 | 6,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |
| 7860 | 7,0 | 1 | 2 | 3 | 4 | NOP | NOP | NOP |

Figure H.6 Test run matrix for Experiment 4

## Appendix H (cont.) Measurement experiments specification

### Experiment 5

| | | | | |
|---|---|---|---|---|
| TestID | 06 | | | |
| APN | utwente.nl | | | |
| HOPS | Server <> Client | | | |
| Test description | Confirmed Service, 8x8, UMTS-UT/BT, Server/Client, 1 Terminal | | | |
| Extra comms | UMTS | | | |
| Intra comms | Bluetooth 1.0 | | | |
| Clock sync | Tardis | | 2.89 s/day clock drift | ASUS notebook |
| Equipment: | | | | |
| Identifier | Name | HW-platform | SW-platform | |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 | |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0, BT dongle | |
| Terminal | Nokia 11 | Nokia 6650 | PR4 | |

Figure H.7 Specification of the Experiment 5

| **500** | 174 | 524 | 1572 | 2096 | 6288 | 8384 | 14672 | 16768 |
|---|---|---|---|---|---|---|---|---|
| 174 | 0,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 524 | 1,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1048 | 2,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1572 | 3,0 | 1 | 2 | 3 | 4 | 5 | 6 | NOP |
| 2096 | 4,0 | 1 | 2 | 3 | 4 | 5 | NOP | NOP |
| 4192 | 5,0 | 1 | 2 | 3 | 4 | 5 | NOP | NOP |
| 6288 | 6,0 | 1 | 2 | 3 | 4 | 5 | NOP | NOP |
| 7860 | 7,0 | 1 | 2 | 3 | 4 | 5 | NOP | NOP |

Figure H.8 Test run matrix for Experiment 5

## Appendix H (cont.) Measurement experiments specification

### Experiment 6

| | | | | |
|---|---|---|---|---|
| TestID | 07 | | | |
| APN | utwente.nl | | | |
| HOPS | Server <> Proxy <> Client | | | |
| Test description | Confirmed Service, 8x8, UMTS-UT/USB, Server/Proxy/Client, 1 Terminal | | | |
| Extra comms | UMTS | | | |
| Intra comms | USB | | | |
| Clock sync | ntpdate | | 11.669 s/day clock drift | ntp.utwente.nl |
| Equipment: | | | | |
| Identifier | Name | HW-platform | SW-platform | |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 | |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0 | |
| Terminal | Nokia 1 | Nokia 6650 | PR4 | |

Figure H.9 Specification of the Experiment 6

| 500 | 174 | 1572 | 2096 | 7860 | 8384 | 16244 | 16768 | 48732 |
|---|---|---|---|---|---|---|---|---|
| 174 | 0,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 524 | 1,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1048 | 2,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1572 | 3,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2096 | 4,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4192 | 5,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6288 | 6,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 7860 | 7,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Figure H.10 Test run matrix for Experiment 6

## Appendix H (cont.) Measurement experiments specification

### Experiment 7

| | | | | |
|---|---|---|---|---|
| TestID | 08 | | | |
| APN | web.vodafone.nl | | | |
| HOPS | Server <> Client | | | |
| Test description | Confirmed Service, 8x8, UMTS-Vodafone/USB, Server/Client, 1 Terminal | | | |
| Extra comms | UMTS | | | |
| Intra comms | USB | | | |
| Clock sync | ntpdate | | 1.9 s/day | ntp.utwente.nl |
| Equipment: | | | | |
| Identifier | Name | HW-platform | SW-platform | |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 | |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0 | |
| Terminal | Nokia 1 | Nokia 6650 | PR4 | |

Figure H.11 Specification of the Experiment 7

Due to the time constraint only two packet sizes are executed, therefore the measurement results are considered as indicative.

| **500** | 174 | 1572 | 2096 | 7860 | 8384 | 16244 | 16768 | 48732 |
|---|---|---|---|---|---|---|---|---|
| 174 | 0,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |
| 524 | 1,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |
| 1048 | 2,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |
| 1572 | 3,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |
| 2096 | 4,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |
| 4192 | 5,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |
| 6288 | 6,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |
| 7860 | 7,0 | 1 | NOP | NOP | NOP | NOP | NOP | NOP |

Figure H.12 Test run matrix for Experiment 7

## Appendix H (cont.) Measurement experiments specification

### Experiment 8

| | | | |
|---|---|---|---|
| TestID | 14 | | |
| APN | utwente.nl | | |
| HOPS | Server <> Client | | |
| Test description | Confirmed Service, 8x8, UMTS-UT, Server/Client, 1 PCMCIA Card | | |
| Extra comms | UMTS | | |
| Intra comms | None | | |
| Clock sync | Tardis 2000 Service v1.5 | -1.892 s/day | ASUS notebook |
| Equipment: | | | |
| Identifier | Name | HW-platform | SW-platform |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0, BT dongle |
| Terminal | PC card | PCMCIA UMTS | |

Figure H.13 Specification of the Experiment 8

| **500** | 174 | 1572 | 2096 | 7860 | 8384 | 16244 | 16768 | 48732 |
|---|---|---|---|---|---|---|---|---|
| 174 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 524 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1048 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1572 | 3 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2096 | 4 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4192 | 5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6288 | 6 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 7860 | 7,0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Figure H.14 Test run matrix for Experiment 8

## Appendix H (cont.) Measurement experiments specification

### Experiment 9

| | | | |
|---|---|---|---|
| TestID | 09 | | |
| APN | utwente.nl | | |
| HOPS | Server <> Client | | |
| Test description | Unconfirmed Service (uplink only), 524B, UMTS-UT/USB, Server/Client, 1 Terminal<br>Buffer size: application 64 KBytes, socket 64 KBytes | | |
| Extra comms | UMTS | | |
| Intra comms | USB | | |
| Clock sync | Tardis 2000 V1.5 | 2.089 s/day | ntp.utwente.nl |
| Equipment: | | | |
| Identifier | Name | HW-platform | SW-platform |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0 |
| Terminal | Nokia 11 | Nokia 6650 | PR4 |

Figure H.15 Specification of the Experiment 9

Number of observations: 30

| packet size: 524 B<br>40 s trans. window | T09 | T10 | T11 |
|---|---|---|---|
| SAT 0.5 | 1 | 2 | 3 |
| SAT 0.6 | 1 | 2 | 3 |
| SAT 0.7 | 1 | 2 | 3 |
| SAT 0.8 | 1 | 2 | 3 |
| SAT 0.9 | 1 | 2 | 3 |
| SAT 1.0 | 1 | 2 | 3 |
| SAT 1.1 | 1 | 2 | 3 |
| SAT 1.2 | 1 | 2 | 3 |
| SAT 1.5 | 1 | 2 | 3 |
| SAT 2.0 | 1 | 2 | 3 |

Figure H.16 Test run matrix for Experiments: 9

## Appendix H (cont.) Measurement experiments specification

### Experiment 10

| | | | |
|---|---|---|---|
| TestID | 10 | | |
| APN | utwente.nl | | |
| HOPS | Server <> Client | | |
| Test description | Unconfirmed Service (uplink only), 524B, UMTS-UT/USB, Server/Client, 1 Terminal<br>Buffer size: application 32 KBytes, socket 64 KBytes | | |
| Extra comms | UMTS | | |
| Intra comms | USB | | |
| Clock sync | Tardis 2000 V1.5 | 2.089 s/day | ntp.utwente.nl |
| Equipment: | | | |
| Identifier | Name | HW-platform | SW-platform |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0 |
| Terminal | Nokia 11 | Nokia 6650 | PR4 |

Figure H.17 Specification of the Experiment 10

Number of observations: 30

| packet size: 524 B<br>40 s trans. window | T09 | T10 | T11 |
|---|---|---|---|
| SAT 0.5 | 1 | 2 | 3 |
| SAT 0.6 | 1 | 2 | 3 |
| SAT 0.7 | 1 | 2 | 3 |
| SAT 0.8 | 1 | 2 | 3 |
| SAT 0.9 | 1 | 2 | 3 |
| SAT 1.0 | 1 | 2 | 3 |
| SAT 1.1 | 1 | 2 | 3 |
| SAT 1.2 | 1 | 2 | 3 |
| SAT 1.5 | 1 | 2 | 3 |
| SAT 2.0 | 1 | 2 | 3 |

Figure H.18 Test run matrix for Experiments: 10

## Appendix H (cont.) Measurement experiments specification

### Experiment 11

| | | | |
|---|---|---|---|
| TestID | 11 | | |
| APN | utwente.nl | | |
| HOPS | Server <> Client | | |
| Test description | Unconfirmed Service (uplink only), 524B, UMTS-UT/USB, Server/Client, 1 Terminal<br>Buffer size: application 32 KBytes, socket 32 KBytes | | |
| Extra comms | UMTS | | |
| Intra comms | USB | | |
| Clock sync | Tardis 2000 V1.5 | 2.089 s/day | ntp.utwente.nl |
| Equipment: | | | |
| Identifier | Name | HW-platform | SW-platform |
| ClientId 01 | Utip194 | P4, 2.4 GHz, 512 MBmem | WindowsXP, JDK 1.3.0 |
| ServerId 01 | Freelander | mP3, 1 GHz, 640MBmem | WindowsXP, JDK 1.3.0 |
| Terminal | Nokia 11 | Nokia 6650 | PR4 |

Figure H.19 Specification of the Experiment 11

Number of observations: 30

| packet size: 524 B<br>40 s trans. window | T09 | T10 | T11 |
|---|---|---|---|
| SAT 0.5 | 1 | 2 | 3 |
| SAT 0.6 | 1 | 2 | 3 |
| SAT 0.7 | 1 | 2 | 3 |
| SAT 0.8 | 1 | 2 | 3 |
| SAT 0.9 | 1 | 2 | 3 |
| SAT 1.0 | 1 | 2 | 3 |
| SAT 1.1 | 1 | 2 | 3 |
| SAT 1.2 | 1 | 2 | 3 |
| SAT 1.5 | 1 | 2 | 3 |
| SAT 2.0 | 1 | 2 | 3 |

Figure H.20 Test run matrix for Experiments: 11

## Appendix I Tardis application

Tardis (http://www.kaska.demon.co.uk/tardis.htm) is a shareware utility for Windows that makes synchronises automatically on scheduled basis the PC's clock. It supports NTP and many other protocols, including GPS (The Global Positioning System), and various Radio clocks.
There are two versions of Tardis:

1) The Service version of Tardis 2000 that runs as a Windows NT/2000/XP/2003 "service", just like other services that come as standard with Windows.
2) The application version of Tardis 2000 does the same thing for Windows 95/98/ME/NT/2000/XP/2003.

Both versions act as both a server and client for the supported protocols. I.e. they can get the time from a timeserver with the 'correct' time and then make the 'correct' time available to local clients. The example Tardis configuration is as shown in Figures below.

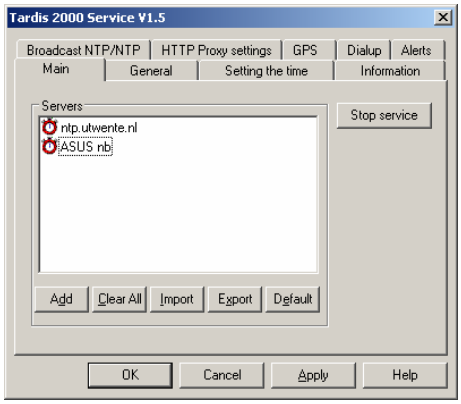



Figure I.1 NTP server(s) configuration

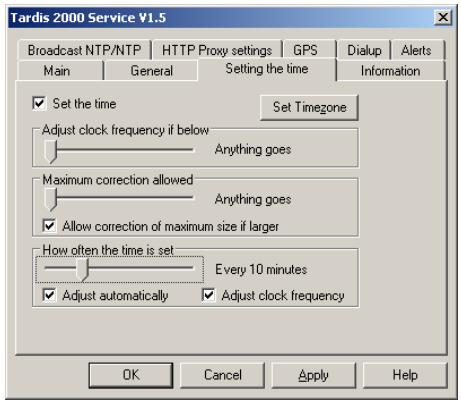



Figure I.2 Setting the time configuration

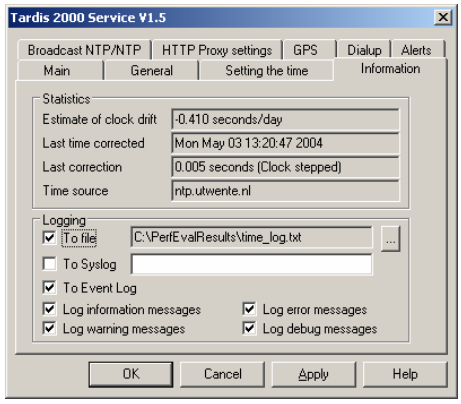



Figure I.3 Logging clock sync information

## Appendix J ntpdate manual

### NAME

ntpdate - set the date and time by way of NTP

### SYNOPSIS

**/usr/sbin/ntpdate** [ **-bBdoqsuv** ] [ **-a** *key#* ] [ **-e** *authdelay* ] [ **-k** *keyfile* ] [ **-m** ] [ **-o** *version* ] [ **-p** *samples* ] [ **-t** *timeout* ] [ **-w** ] *server*

### DESCRIPTION

The ntpdate utility sets the local date and time. To deter mine the correct time, it polls the Network Time Protocol (NTP) servers on the hosts given as arguments. This utility must be run as root on the local host. It obtains a number of samples from each of the servers and applies the standard NTP clock filter and selection algorithms to select the best of these.

The reliability and precision of ntpdate improve dramatically with a greater number of servers. While a single server may be used, better performance and greater resistance to inaccuracy on the part of any one server can be obtained by providing at least three or four servers, if not more.

The ntpdate utility makes time adjustments in one of two ways. If it determines that your clock is off by more than 0.5 seconds it simply steps the time by calling **gettimeofday**. If the error is less than 0.5 seconds, by default, it slews the clock's time with the offset, by way of a call to **adjtime**. The latter technique is less disruptive and more accurate when the offset is small; it works quite well when ntpdate is run by cron every hour or two. The adjustment made in the latter case is actually 50% larger than the measured offset. This adjustment tends to keep a badly drifting clock more accurate, at some expense to stability. This trade-off is usually advantageous. At boot time, however, it is usually better to step the time. This can be forced in all cases by specifying the -b option on the command line.

The ntpdate utility declines to set the date if an NTP server daemon like **xntpd** is running on the same host. It can be run on a regular basis from **cron** as an alternative to running a daemon. Doing so once every one to two hours results in precise enough timekeeping to avoid stepping the clock.

### OPTIONS

The following options are supported:

-a *key#* Authenticate transactions, using the key number, *key#*.

-b Step the time by calling **gettimeofday**.

-B Force the time to always be slewed using the **adjtime** system call, even if the measured offset is greater than +-128 ms. The default is to step the time using **settimeofday** if the offset is greater than +-128 ms. If the offset is much greater than +-128 ms in this case, that it can take a long time (hours) to slew the clock to the correct value. During this time the host should not be used to synchronise clients.

## Appendix J (cont.) ntpdate manual

-d Display what will be done without actually doing it. Information useful for general debugging is also printed.

-e *authdelay* Specify an authentication processing delay, *auth- delay* in seconds. See **xntpd** for details. This number is usually small enough to be negligible for purposes of ntpdate. However, specifying a value may improve timekeeping on very slow CPU's.

-k *keyfile* Read keys from the file *keyfile* instead of the default file, /etc/ntp.keys. *keyfile* should be in the format described in **xntpd**.

-m Join multicast group specified in *server* and synchronise to multicast NTP packets. The standard NTP group is 224.0.1.1.

-o *version* Force the program to poll as a version 1 or version 2 implementation. By default ntpdate claims to be an NTP version 3 implementation in its outgoing packets. However, some older software declines to respond to version 3 queries. This option can be used in these cases.

-p *samples* Set the number of samples ntpdate acquires from each server. *Samples* can be between 1 and 8 inclusive. The default is 4.

-q Query only. Do not set the clock.

-s Log actions by way of the **syslog** facility rather than to the standard output - a useful option when running the program from **cron**.

-t *timeout* Set the time ntpdate spends, waiting for a response. *Timeout* is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

-u Use an unprivileged port to send the packets from. This option is useful when you are behind a firewall that blocks incoming traffic to privileged ports, and you want to synchronise with hosts beyond the firewall. The -d option always uses unprivileged ports.

-v Be verbose. This option causes ntpdate's version identification string to be logged.

-w Wait until able to synchronise with a server. When the -w option is used together with -m, ntpdate waits until able to join the group and synchronise.

## Appendix J (cont.) ntpdate manual

**FILES**

/etc/inet/ntp.keys Contains the encryption keys used by ntpdate.

**ATTRIBUTES**

See attributes for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---|---|
| Availability | SUNWntpu |

**SEE ALSO**

cron**,** xntpd**,** adjtime**,** gettimeofday**,** settimeofday, syslog**,** attributes

**NOTES**

The technique of compensating for clock oscillator errors to improve accuracy is inadequate. However, to further improve accuracy would require the program to save *state* from previous runs.

# Appendix K Description Final Research Project

**Description Final Research Project**

| **Name:** | **CHAIR/Group** | **Supervisor UT** |
|---|---|---|
| Katarzyna Wac<br>ing. Richard Bults | APS | Dr. ir. A.T. van Halteren |

**Title:** Performance evaluation of a transport service supporting the MobiHealth BAN Interconnect Protocol: Methodology and Assessment.

**Abstract**
MobiHealth is a European project (IST-2001-36006) that explores the possibilities of 2.5G and 3G networks to support new m-health services. Standard TCP/IP based Internet networking protocols are used to transport the m-health service data over these next generation broadband wireless networks.

The MobiHealth m-health service is based on a telematics system that consists of three major building blocks: 1) BAN (Body Area Network), 2) BEsys (Back-End system) and 3) transport service (i.e. data communication system). The BAN can be used for example as a healthcare monitoring tool to measure vital sign data of ambulatory patients. The BEsys controls a BAN and makes the vital sign data available to a (remote) healthcare provider (e.g. hospital). Transportation of vital sign data and control data is handled by the MobiHealth BAN Interconnect Protocol which in turn is supported by the transport service.

The usability of the MobiHealth service depends on a minimum guaranteed Quality of Service (QoS) of the transport service. The transport service is provided by a data communication system. This system consists of various subsystems with discrete performance characteristics (e.g. speed, delay) that influence the overall system performance.

**Objective**
The objective of this final project is to apply an appropriate performance evaluation of a transport service supporting the MobiHealth BAN Interconnect Protocol. The assignment includes the methodology for the setup of a measurements strategy to assess the performance characteristics of a particular (i.e. reliable, unreliable) transport service under predefined loads. Measurement results are used to create a high level abstract model of the transport service, in order to assess the scalability of the MobiHealth system. Of particular interest is the scalability of the MobiHealth system in relation to the number of BANs in the system.

**Roles**
Student ing. R.G.A. Bults is responsible for the methodology phase of the assignment, while student K.E.Wac is responsible for the assessment phase of the assignment.

**Approach**
- Study relevant transport service technologies (e.g. GPRS/UMTS, UDP, TCP, wireless TCP/IP)
- Survey and selection of performance evaluation methodology
- Perform performance measurements on selected MobiHealth transport service
- Derive a performance model of the transport system and assess the scalability of the transport service
- Write thesis

**External/Internal**
Internal

| **Period** | **Graduation committee** |
|---|---|
| 1th Feb, 2003 – 1st November, 2004 | (chairman) Prof. Dr. ir. D. Konstantas |
| **Date of report** | (1st coordinator) Dr. ir. A.T. van Halteren |
| October 2004 | (2nd coordinator) Dr. ir. V. Nicola |
| **Approval of supervisor UT** | |