End-to-end key performance indicators in cellular networks

by

Bengt Magnor Orstad
Erling Reizer

Thesis in partial fulfilment of the degree of Master in Technology in Information and Communication Technology

Agder University College
Faculty of Engineering and Science

Grimstad
Norway

May 2006

Abstract

The continuing growth of customers taking advantage of the available services means greater load on the cellular network. Optimization is the key to ensure that the network can provide a reasonable level of service-quality. Service providers want to examine their network and be assured that their network is performing well. Teleca Wireless Solutions is a company that does this for service providers, and an end-to-end test tool could be proven useful to examine the cellular networks overall performance from an end users point of view. To ensure that one has a tool that does this in an appropriate way, one must ensure, the application is based on testing the right key performance indicators for relevant services. Such services may be popular services like FTP and WEB.

In this thesis, we have researched what affects the end users performance and performed practical end-to-end performance tests in cellular networks. Our goal is to define which key performance indicators are affecting the network's performance at different network layers and for different services. We have paid special attendance to the high latency of the wireless links, and the delay introduced with the radio access bearer establishment. By measurements we have shown that the 3rd generation cellular network UMTS not surprisingly outperforms EDGE regarding commonly used services like HTTP/WEB and FTP. We have discovered that while TCP throughput is good when transferring large files over FTP, the high latency of the wireless link makes the HTTP performance bad compared to potential TCP throughput.

Our work has concluded with which key performance indicators an end-to-end test application should measure for services as HTTP and FTP, to give an overall view of the cellular network's performance. We have proposed enhancements to an already existing end-to-end test tool.

Preface

This thesis has been carried out from January to May 2006 and is the final of the master degree in Information and Communication Technology (ICT) at Agder University College, Faculty of Engineering and Science in Grimstad. The workload of this thesis equals 30 ECTS. The thesis has been carried out on assignment and in co-operation with Teleca Wireless Solutions AS in Grimstad.

We would like to thank our supervisors, Professor Dr. Frank Reichert at Agder University College, and Per Magne Hoff and Per Arne Olsen at Teleca Wireless Solutions for excellent supervision and guidance.

Grimstad, May 2006.

Bengt Magnor Orstad & Erling Reizer.

Table of Contents

List of Figures

List of Tables

1 Introduction and problems

Today's mobile phones are no longer voice-call only equipment. High-bandwidth internet is becoming available through EDGE and the 3rd generation mobile systems. Packet switched data communication is playing a more and more significant role in the mobile network and is seen as a more efficient way to utilize the resources than circuit switched connections. Progression from the first systems to today's UMTS has resulted in more processing power, more availability and more services, due to the continuing growth of customers and newer technology available.

New services for mobile phones, like email, web browsing, audio and video streaming demands more and more from the underlying network. If the network do not deliver what these services needs, the performance and the user experience will be unsatisfactory. Mobile phones are also today often used as dial-up modems to connect portable personal computers to the internet, because of the coverage, quick access and no need for cables of any kind. There are increasing needs for high bit rates, low delay and jitter. These are three parameters that strongly affect the user performance, and are examples of what we call Key Performance Indicators (KPI). There are different KPIs for different services and on different network layers. To identify these will make it easier to optimize the network and applications. Therefore, it is useful for companies who specialize in cellular network optimization or even service providers to have the ability to measure the performance of the network, for the purpose to optimize the network usage and enhance customer experience.

Teleca Wireless Solutions has developed an application for performing tests and measurements of end-user performance. Such measurements are e.g. throughput and delay, for applications like file transfer and web browsing. To measure the end-to-end performance in cellular networks (and wired networks too), is important to discover any problems or issues that needs further investigation. Teleca are interested in enhancing the application and get outlined which indicators that mostly characterize the end-to-end performance. The concept of an end-to-end performance test tool is that one measures the performance from the users' standpoint of the cellular network, i.e. one end. On the other end there may be a computer connected to the internet serving as e.g a fileserver or streaming server. One has no specialized equipment installed in the core network. Obviously one has no detailed information of internal signaling and details in the network itself, one is solely depended on the information measured from the mobile terminal and information extracted from the mobile terminal. To determine which information that will tell us about the performance of the cellular network is a challenging task. There are various parameters and indicators for different services, and it is not given that one parameter, like latency, will affect all services equally.

To identify these parameters is a challenging task that requires both an in depth literature review of existing research and practical measurements. We will perform realistic measurements with Teleca's existing tool in real networks. Delay, especially in the form of access delay and RTT are some parameters we will look into.

One subproblem will then be to look at which AT-commands can be used to extract information from the mobile phone and the network that can affect the performance, like signal strength and coding scheme.

Another subproblem will be to define what parameters or indicators, i.e. KPIs, which affect the user experience of different services, in both GSM/EDGE and UMTS networks. We will also pay special attention to the transmission control protocol (TCP) and research what affects the performance of this much used transport protocol, and also what affects the performance of the wireless network at different layers, from link layer to application layer.

End-to-end performance measuring is difficult, much because one has no direct knowledge of, or equipment installed in, the core network. One is entirely dependent on measuring by testing the network by accessing it like an ordinary end user. We will look into other interesting tools and applications that can be used for measurements. Like stated earlier in this introduction, Teleca are interested in enhancing their application, thus a problem we will discuss is what an enhanced end-to-end test-application should do and report, and why.

1.1 Report outline

Chapter 1 is the introduction to and background for the thesis.

Chapter 2 is the state of the art chapter with a brief introduction to mobile communications technologies, AT-commands, internet protocols and the applications we are using throughout this thesis.

Chapter 3 covers the practical, testing, and results part of the thesis.

Chapter 4 is the summary and discussion part.

Chapter 5 contains the conclusion of this thesis.

1.2 Resources

To our disposition, we had TWSE2E, Teleca's end-to-end agent that measures the end-to-end performance by performing various test scenarios. It runs on a personal computer with a GPRS dialup connection. Our phone used as a modem was the Nokia N70. This phone is both EDGE and 3G capable. Teleca was interested in evaluating how a Nokia phone is suitable for such end-to-end testing, as they traditionally have more experience with Sony Ericsson mobile phones. We had access to SIM cards for access to both the Netcom and Telenor network. Iperf, a TCP and UDP throughput measurer was used for performing measurements. The computer utilizing the modem function of the N70 was a Laptop running Windows XP, and it was connected to the mobile phone via either Bluetooth or USB cable. To our disposition we also had a server located on HiA's network.

N70
NOKIA
Timeline
26/08/2005
We shall fight on the beaches, we shall ...
Be Ye Men of Valour.
We shall fight in the hills; we shall ...
Options
Exit

Figure 1 Nokia N70 GSM, EDGE and 3G phone.

2 State of the Art

This chapter describes and introduces the main themes in our thesis. As the various radio technologies, namely GSM and WCDMA, has different characteristics, some in depth description of them is needed.

2.1 Cellular network overview

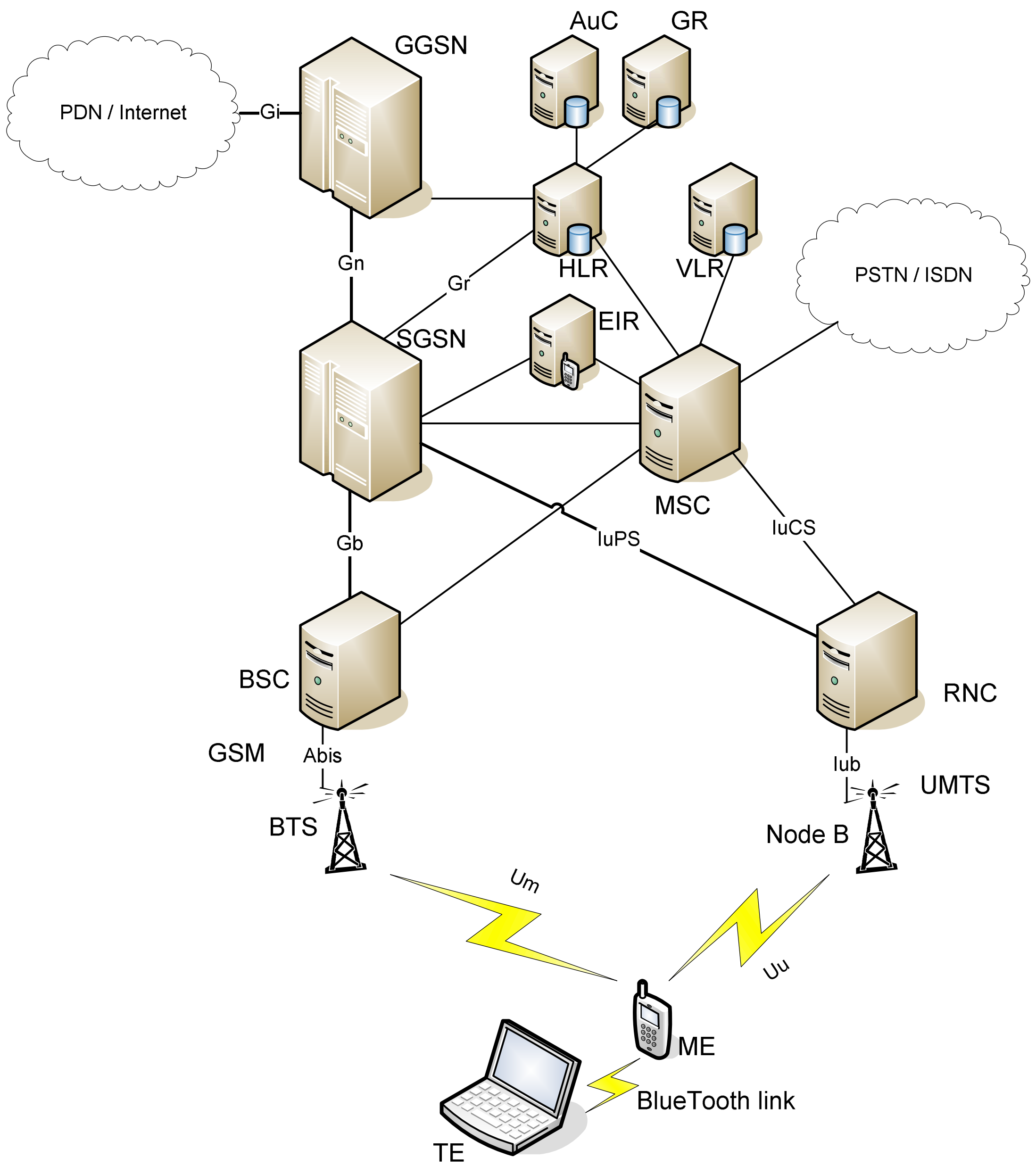


Figure 2 GSM & UMTS network overview

2.2 Radio technologies

In this chapter we will shortly describe the two radio access technologies relevant to this thesis, GSM and UMTS/WCDMA.

2.2.1 GSM

Global System for Mobile Communications is the second generation mobile telephony system, or 2G. The GSM group was founded in 1982 and is the leading cellular-system with over a billion end users and a market share of 70% [1]. GSM is a cellular technology, which means that terminals (cellular phones or TE) searches for cells and connects to the base station with the best radio-signal. GSM uses frequency-divided duplex (FDD), which means one frequency for up- and downlink. Time divided multiple access, or TDMA, is used for multiple access [2].

GSM®
GLOBAL SYSTEM FOR
MOBILE COMMUNICATIONS

2.2.1.1 Architecture

As seen in Figure 4, the GSM system consists of three subsystems, the radio subsystem (RSS), the network and switching subsystem (NSS) and the operation subsystem (OSS).

2.2.1.2 Radio Subsystem

The RSS comprises the radio specific devices such as the MS and the whole base station subsystem (BSS) which consists of the base station controller (BSC) and the base station transceiver (BTS). The BTS [3] is in simple words the "antenna" on the network side. It comprises the radio equipment. The BSC controls one or several BTSs. It reserves radio frequencies and performs handovers from one BTS to another. The MS is the user equipment that has the necessary hardware and software to communicate with a GSM network. A MS consists of the subscriber identity module (SIM). Figure 3 show how a BTS can look like.

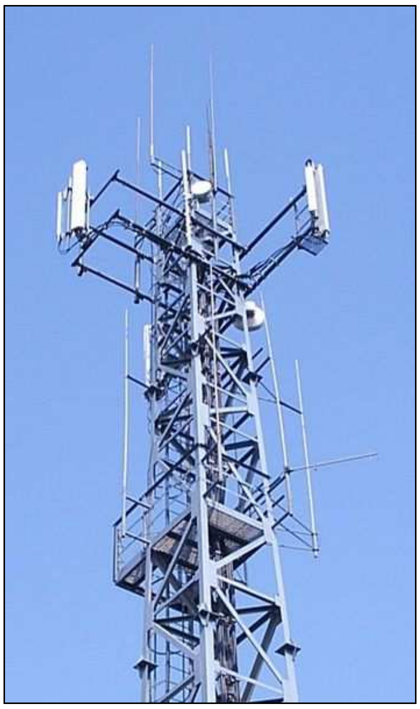


Figure 3 Base station transceiver [1]

2.2.1.3 Network and switching subsystem

The NSS connects the wireless network with the wired network, i.e. ordinary phone lines. Handovers between different BSSs is managed in the NSS. The mobile services switching center (MSC) is an advanced digital ISDN switch. It forms the backbone of the GSM system and can manage one or several BSCs. A gateway MSC has connections to other fixed networks, such as ISDN or analog networks. The home location register (HLR) contains subscriber information. Subscriber information is e.g. the mobile subscriber ISDN number (MSISDN), and the international mobile subscriber identity (IMSI). The location of the current location area (LA) is also stored here and is dynamically updated. The visitor location register (VLR) is associated to a MSC and stores all information for the MS currently in the LA associated to that particular MSC. Such information is typically IMSI, MSISDN and information about the HLR of that MS [4], [5].

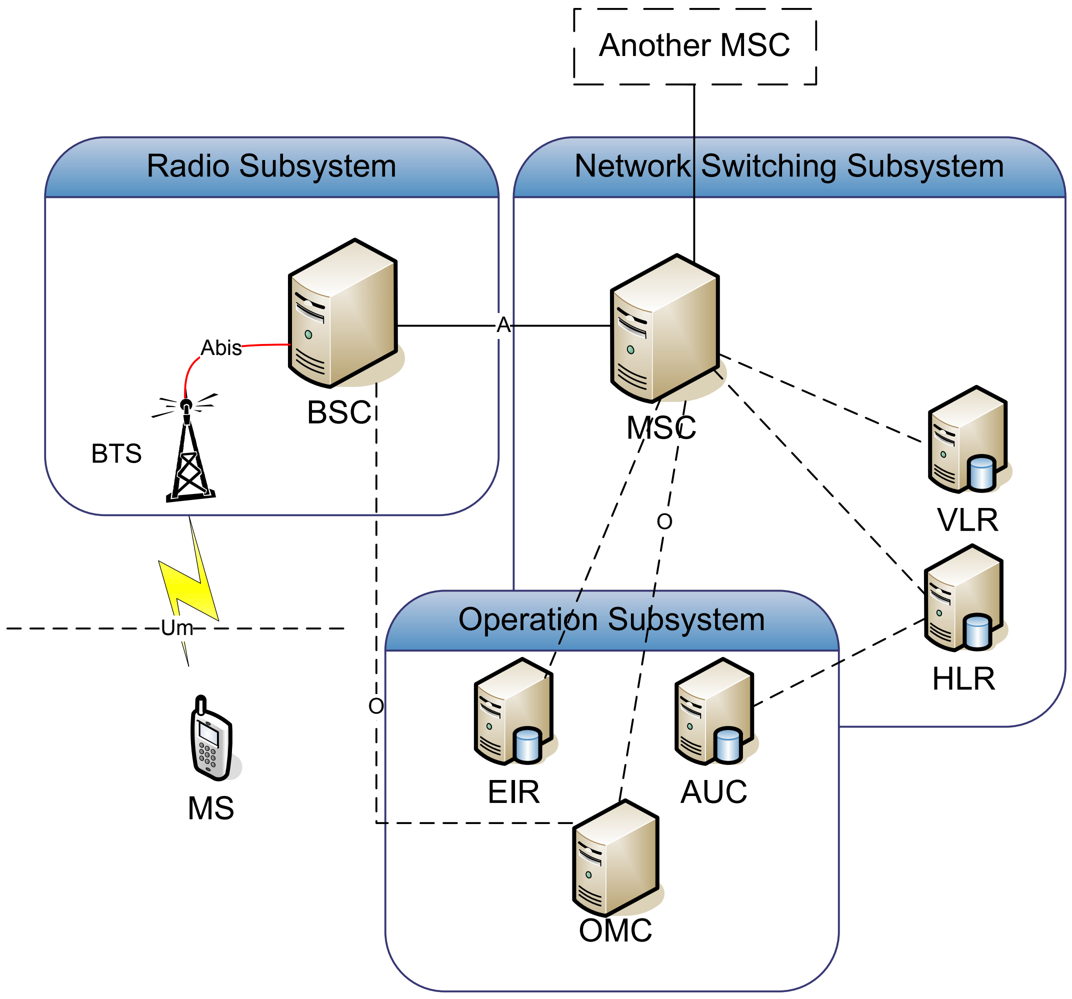


Figure 4 GSM architecture

2.2.1.4 Operation subsystem

This system contains the functions for network operation and maintenance. Signaling system number 7 (SS7) is used to access other network elements. The OMC monitors and controls the network, and provides traffic monitoring and status reports, accounting and billing. The authentication centre (AuC) contains the algorithms for authentication and each subscriber's authentication keys. It generates the values needed for user authentication. The equipment identity register (EIR) contains a register over all devices registered for its network. Typically and ideally a reported stolen MS should be blacklisted here and not allowed used in the network. [2]

2.2.1.5 Tele services

The goal with GSM was to provide a wireless mobile phone system that could provide services compatible to ISDN, and give the users ability to roam throughout Europe. High quality digital voice calls was the primary target for GSM, and voice call is still the most used service. The short message service, SMS is a service that has experienced an enormous popularity in Europe since the mid nineties. It allows text messages with up to 160 characters to be sent. These SMS messages do not use traditionally data channels, but are transferred over the signaling channels. This means that sending and receiving SMS is possible during voice and data communication. SMS is also the only way to reach the MS from the network, so it is used for updating MS software and push services. GSM also provides another tele service, group 3 fax, which transmits fax data as digital data over the analog network [2].

2.2.2 UMTS

Universal Mobile Telecommunications Systems, UMTS [5], [6] or 3G is the newest system mobile network operators use today, and it is the common system used in Europe. The air interface is based on the Wideband Code Division Multiple Access (W-CDMA) standard. W-CDMA utilizes code division multiple access (CDMA). This is a new radio interface compared to GSM / 2G. While GSM uses time division as access technique, W-CDMA uses different code sequences that are spread to increase the bit rate before transmitting the data, and the end user can expect speeds up to 384 kbps, although theoretical limit is 1920 kbps. Upgrades are being developed and higher and higher bit rates will be introduced. According to [7] if the network is being overloaded, congestion control can be done in four steps. First step is to reduce bit rate of non-realtime applications. Then some subscribers will be moved to less loaded frequencies. Furthermore moving some subscribers to GSM and the fourth method of resolving overload is to disconnect some subscribers to ensure the quality for the remaining ones. Figure 5 illustrates the UMTS network.

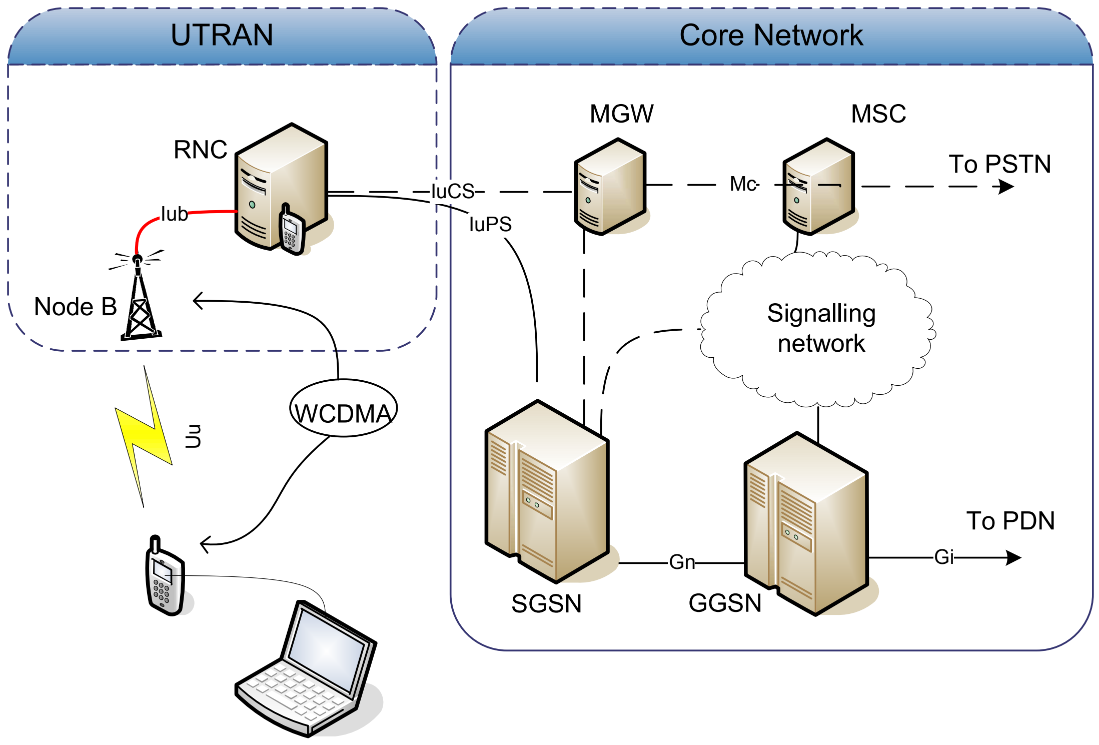


Figure 5 UMTS Network

2.3 GPRS

General Packet Radio Service (GPRS) [8] is packet switched (PS) communication enhancement for GSM. GPRS is often recognized as ”2.5G”, since it is halfway a step towards UMTS or 3G, but this isn’t an official term. The PS domain optimizes the use of network and radio resources. GPRS was originally developed for GSM, but the network subsystem and radio interface was kept separated so that re-use of these with other radio technologies is possible. A common core network is used with both GPRS and UMTS. This enables the ability to have realtime sound/video transfer together with non-realtime data transfer. GPRS data is usually billed per kilobyte of downloaded and uploaded data [9]. Such data volume based charging is based on SGSN and GGSN Call Data Records (CDRs). The trend is however towards “contents based” charging where e.g. the GGSN actively looks into the data contents and generates CDRs based on e.g. which site/link information.

2.3.1 Technical overview

GPRS and the packet domain transfers packet in an efficient way and optimized the use of network and radio resources. Unlike circuit switched communication, resources are not held when they are not used. To allow the network subsystem to be reused with other radio access technologies, a strict separation between the radio subsystem and the network subsystem is maintained. This allows a common packet domain core network for both GSM and UMTS.

This packet switched core network is designed to provide various Quality of Service (QoS) levels for various types of traffic, including both non-realtime traffic like mail and web and realtime applications like voice and video streaming.

2.3.1.1 Nodes introduced

Some new nodes are introduced with the packet switched domain. The Serving GPRS Support Node (SGSN) maintains the overview of where the MS is located and performs access control and security functions. The SGSN is connected either to the GSM base station system through the Gb interface, or to the UMTS Radio Access Network through the Iu interface. [8]

GGSN Gateway GPRS Support Node is a router witch de-tunnels user data from GPRS tunneling protocol and transmits it as ordinary IP-packets. This node contains routing information for GPRS. Address conversion is performed in this node, and it is connected to external networks via the Gi interface. Packets are transferred via an IP-based GPRS backbone to the SGSN via the Gn interface. [8]

SGSN Serving GPRS Support Node does many similar tasks like the Local Agent in Mobile IP, but is even more complex since it also manages the connection to the radio interface. The SGSN feature’s is therefore not equal in GSM and UMTS. It is connected to the MS via the Gb interface. It requests user addresses from the GPRS register, counts bytes for billing, performs various security functions and access control and keeps track of each individual MS’s location. [8]

MSC Mobile services Switching Centre, advanced digital switch. It is only used for signaling in GPRS [2].

PCU Packet Control Unit does some of BSC’s exercises for packet switched data. The PCU may be implemented in the BTS, BSC or even in the SGSN. The allocation of channels between voice and data is done by the BTS, but when a channel is allocated to the PCU it takes full control of it. The implementation of RLC/MAC procedures on the network side is here [3].

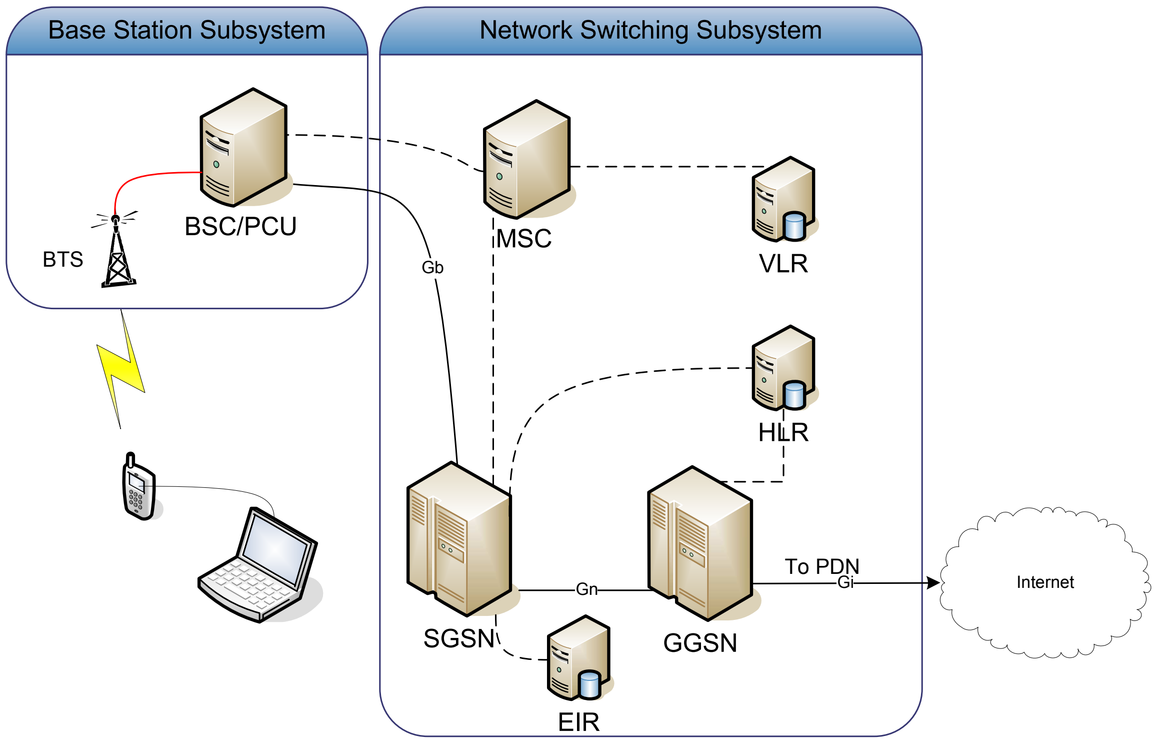


Figure 6 GPRS with GSM radio access.

The radio link protocol (RLC) provides a reliable link. To provide a reliable link, retransmission techniques are used. The nature of this delivered data is affecting upper layer protocols, in particular TCP.

To register the MS to the GPRS network, a GPRS attach is performed. However, no data can be sent or received until a PDP context is activated. The PDP context [8], or Packet Data Protocol context, must be defined before any data can be transmitted, and it defines the context for a data call, including address and QoS. Several PDP contexts can exist for one MS at the same time. To transfer the data, in GSM, a Temporary Block Flow (TBF) must be established. It is as the name says a temporal connection between the MS and the PCU and it transmits data in a specific direction, i.e. there are independent TBFs for uplink and downlink. The TBF establishment introduces a significant delay, and originally the TBFs were released at once when all data transfers were finished at the LLC level. This delay has a negative effect on service performance, especially with bursty data. To minimize this effect, techniques to keep the TBF alive for some time after data transmission is finished has been introduced.

2.3.1.2 Session establishment

As mentioned earlier, to be able to transfer any data, the user needs to set up a packet data protocol (PDP) context. The PDP context defines and describes the connection to the external packet data network. QoS profile for the context is among the parameters. To set up a PDP context, these steps are involved:

- The terminal requests a PDP context activation, included is e.g. request QoS profile for the context.

- When the SGSN receives the activate request it checks it against the subscriber information received from the HLR. If the requested QoS violates with the subscription, it may be rejected, or simply ignored and the subscriber given another profile.
- The access point name (APN) is sent by the SGSN to a DNS server to find the IP address of the GGSN that is connected to the required network.
- SGSN tries to set up radio access bearers (RAB). QoS re-negotiating may be required.
- SGSN sends a PDP create context message to the GGSN which must accept or decline.
- An IP tunnel is created between the used GGSN and the SGSN and assigned a tunnel id.
- The MS is assigned a PDP address, and the PDP context is stored in the mobile, SGSN, GGSN and HLR. The PDP address is normally an IP-address.

If the user requests services that need two different QoS profiles, e.g. web surfing and streaming or real-time VoIP, a secondary PDP context must be activated and two IP addresses are used. In UMTS Release 99, however, the secondary PDP context can be used by several application flows using the same address, APN and PDP type, but with different QoS profiles. The flows are differentiated by a network layer service access point identifier (NSAPI) number. [10]

2.3.2 End to End behavior in GPRS

It should be noticed that for an end-to-end application, the GPRS/GSM infrastructure is completely invincible. The application uses ordinary internet transport protocols like TCP and UDP on top of IP. To prevent unwanted attacks on the mobile terminal from the Internet, the GGSN usually uses NAT [11] and firewall [12] features to assign local IP-addresses to each terminal. Local IP-addresses are not routed through the internet, so it is not possible to reach a MS from the internet. Since a subscriber has to pay for traffic, even though it originates from an attack, this is usually also in the subscribers interest. However, NAT and local addresses suffers from the same issues as in the wired internet, and it can make peer-to-peer applications and multimedia streaming with UDP in particularly difficult. The network operators therefore also often provide APNs which assigns public IP-addresses. A reason for NAT could also be the limited set of IPv4 addresses [13]. It is a question if all MS will be assigned public addresses when IPv6 [14] is introduced, and if, how the MS then will be protected from attacks.

2.3.3 Enhanced Data rates for GSM Evolution / EDGE

Enhanced Data rates for GSM Evolution (EGPRS) [15], or simply EDGE, is an enhancement to GPRS. It is often referred to unofficially as 2,75G, although some, especially network operates, even call it 3G. 2,75G implies that it is an enhancement of GPRS, but not a true 3G system. It only introduces changes to GPRS on the base station system [16]. It can be implemented in all GSM systems capable of GPRS with the necessary upgrades to the carrier, i.e. hardware upgrade in the radio interface, including base station and mobile terminal. The main difference is another modulation scheme, 8 Phase Shift Keying modulation (8PSK), which encodes three bit for each symbol, compared to GPRS which uses GMSK (Gaussian minimum-shift keying) and produces one bit per symbol. This means that the bit rate can be increased with a factor of three. It increases throughput for the system overall, not only for the individual user. [1], [15]

There are four coding schemes for GPRS and nine for EDGE, and the system chooses coding schemes depending on the error rate of the radio link. The lower schemes provide good error correction capabilities, and the highest none or very limited. As we can see in figure two, maximum user data throughput with coding scheme MCS9 is 59,2kb/s per timeslot, compared to GPRS with 20kb/s (CS4). All coding schemes in GPRS utilize GMSK modulation, while MCS1-MCS4 in EDGE uses GMSK and MCS5 and upwards uses 8PSK. In very bad radio conditions, GMSK is more efficient than 8PSK, that is why it still is used in EDGE. The reason for difference in throughput between CS1-CS4 and MCS1-MCS4 which all uses GMSK is differences in EDGE headers and GPRS headers and payload. EDGE supports a technique called resegmenting, i.e. retransmitting with another coding scheme. If for instance, a packet is lost with a higher coding scheme, it can be retransmitted with a lower coding scheme, and thus with more error correction. This is not possible in ordinary GPRS and is the reason for the difference in payload size between the first four coding schemes in GPRS and EDGE. [15]

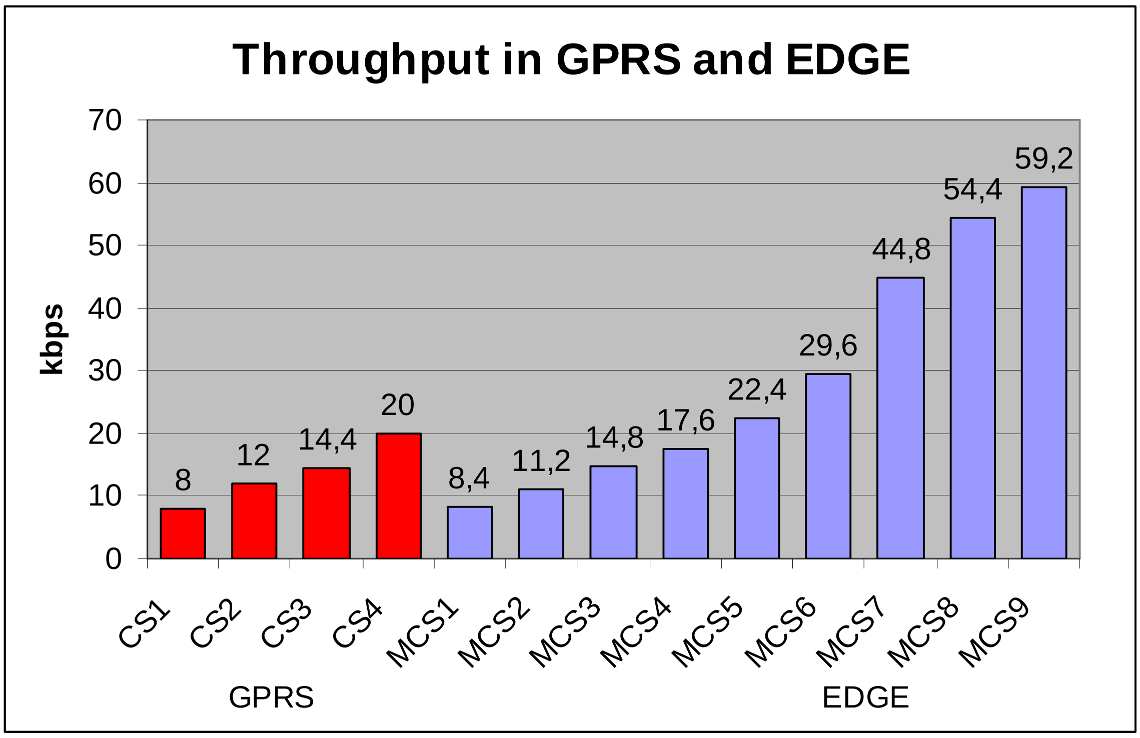


Figure 7 Throughputs in GPRS (red) and EDGE (blue) per timeslot [4].

2.4 Modem AT commands

Mobile phones use radio modems to communicate with base stations over the air interface, modulating and demodulating ones and zeroes to analogue signals. AT commands are used for controlling and assigning commands to a modem. Hayes Smartmodems were among the first who started using a command-set for their modems, called "Hayes Command Set" [17]

With a terminal application on a computer, like the built in HyperTerminal in Windows XP, and an interface for communicating with a mobile phone, we can manually control the mobile phone's modem function using AT commands.

2.4.1 Examples of AT-command use

AT commands are used as a prefix when giving commands to a modem.

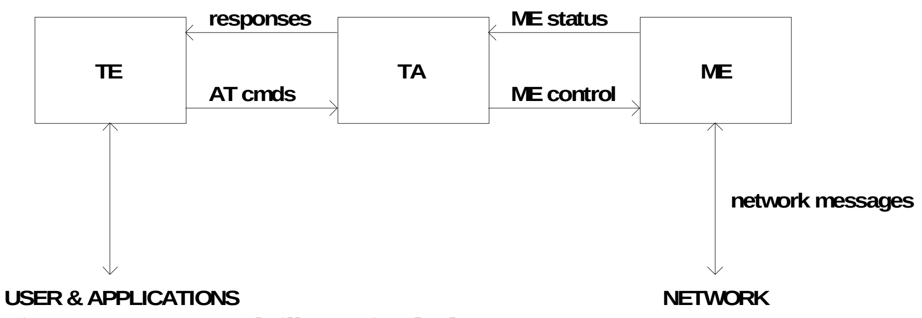


Figure 8 AT commands illustration [18]

Figure 8 shows the user with his Terminal Equipment, or TE (i.e. laptop computer), sending AT commands through Terminal Adaptor, or TA (i.e. a Bluetooth interface that enables a computer to connect to the mobile phone) to control the Mobile Equipment (ME), or the phone itself. Then the user has the ability to obtain information on the state of the cellular network [18], [19].

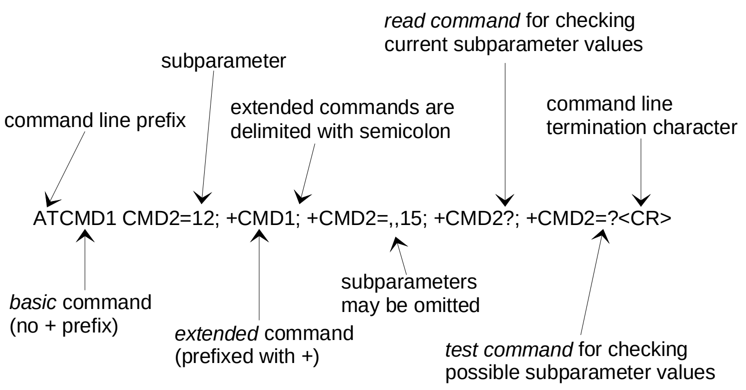

read command for checking current subparameter values
subparameter
extended commands are delimited with semicolon
command line prefix
command line termination character
ATCMD1 CMD2=12; +CMD1; +CMD2=,,15; +CMD2?; +CMD2=?<CR>
basic command (no + prefix)
subparameters may be omitted
extended command (prefixed with +)
test command for checking possible subparameter values
Figure 9 Example of an AT-command syntax [18]

Figure 9 above shows the syntax for how AT commands are used in a terminal application. Below is an example of a simple command and it's response.

Example:
Requesting ME revision
```
at+cgmr
```

Returns manufacturer revision
```
V 05wk07v31
27-10-05
RM-84
(c) Nokia.
```

2.5 *Internet protocols*

2.5.1 Transport layer protocols

TCP

The Transport Control Protocol is a connection oriented reliable transport-layer protocol, which means that two entities in a network first establish a connection between themselves, before transmitting the actual data. The main functionality is to deliver a fault-tolerant data transmission method over IP, by retransmitting data that has gotten lost or corrupt on the way from sender to receiver. TCP has flow control, congestion control and guarantees reliable error free in-order delivery of data. Because of this feature, TCP does not work well with realtime-applications such as streaming video, as it will require extra high amount of bandwidth required due to retransmission [20].

UDP

User Datagram Protocol is a simple connectionless best-effort transport-layer protocol. There are no mechanisms for error or flow control like in TCP, and therefore no guarantees that every packet is received, or that packets are received in the correct order. UDPs primary field of use is mainly realtime applications such as multiplayer-gaming, voice over IP and video streaming. Applications where low delays are highly appreciated, like DNS, are also utilizing UDP [21], [22].

2.5.2 Streaming media

Streaming media is media (audio or video) that is being presented to the user while being delivered. This technology is widely used in the internet as a means for viewing movie clips or listening to an internet radio. Some TV-stations are also streaming some of their live-broadcasts out on the internet, and with the introduction of 3G, mobile phones can be used as a means of obtaining such a stream from the internet. Protocols used while streaming are RTSP and RTP over UDP [10].

2.5.3 Application layer protocols

FTP

File Transfer Protocol is a simple application-layer protocol designed for transferring files from a server to a client or in the opposite direction. FTP is run over TCP-protocol and communication between the two data-exchanging parts is done by using one port for control and another for the data-transfer itself. [23]

HTTP

Hyper Text Transfer Protocol is a protocol developed for the World Wide Web. It is a stateless request/response protocol used primary for web-browsing. It runs over TCP, and typically uses the common port 80 for request and data transfer. [24]

2.6 Measurement applications

2.6.1 TWSE2E

TWSE2E is an application developed by Teleca Wireless Solutions written in the java programming language for running end-to-end test in the cellular network. It has no fancy graphical user interface, but runs from the command line. Configuring the program, for instance selecting which tests to run, is done in a configurations file. The application can perform several tests, these include: signaling (GPRS-attach/detach and activate/deactivate) and measure the time these actions need. After creating a dialup connection with the phone, one can perform these tests: html-download, ftp-down and upload and ping for round trip time calculation with various packet sizes. The program gathers information from these tests, like bandwidth and delay, and writes it to a file called statistics.log. A Microsoft Excel spreadsheet is made that imports data from the statistics.log file and presents it nicely with graphs and computes data like average throughput in kbps on file transfers. Since one use the phone as a dialup modem, one has to be careful with other applications like mail checking and instant messaging which might use network resources and affect the result of the TWSE2E test. Even small details like internet clock updating in Microsoft Windows will affect the result.

2.6.2 Windows Performance Counter

The Performance Counter is located in Control Panel, Administrative tool under the name "Performance". It is part of the Microsoft Management Console and is shipped with Microsoft XP Professional. It is a tool used to measure all sorts of information, in our case, all sorts of data traffic, like bytes received at interface per second, and TCP segments transmitted and received per second. One can log the information to comma delimited text files which then can be imported into Microsoft Excel for plotting etc. The total bytes received and transmitted on the network interface (modem) can be interesting to use to determine which coding scheme is used in GPRS and EDGE, as there is no way to request this information from the phone itself.

2.6.3 Ethereal

Ethereal is an open source network protocol analyzer, aka packet sniffer. It is licensed under the GNU General Public License, and is available from http://www.ethereal.com. It is available for both MS Windows and GNU/Linux. It is very useful for analyzing strange measurements and to verify that no other applications than the one that were supposed to access the internet is doing it.

2.6.4 Iperf

Iperf is a tool for measuring TCP and UDP bandwidth performance. In UDP mode it should be an effective application for simulating streaming based traffic. Quoting from the documentation for version 1.7.0, its features are:

- TCP
    - Measure bandwidth
    - Report MSS/MTU size and observed read sizes.
    - Support for TCP window size via socket buffers.
    - Multithreaded if pthreads or Win32 threads are available. Client and server can have multiple simultaneous connections.

- UDP
    - Client can create UDP streams of specified bandwidth.
    - Measure packet loss.[1]
    - Measure delay jitter.
    - Multicast capable.
    - Multithreaded if pthreads are available. Client and server can have multiple simultaneous connections (This does not work in Windows).

- Where appropriate, options can be specified with K (kilo-) and M (mega-) suffices. So 128K instead of 131072 bytes.
- Can run for specified time, rather than a set amount of data to transfer.
- Picks the best units for the size of data being reported.
- Server handles multiple connections, rather than quitting after a single test.
- Print periodic, intermediate bandwidth, jitter, and loss reports at specified intervals.
- Run the server as a daemon.
- Use the servers as a Windows NT Service.
- Use representative streams to test out how link layer compression affects your achievable bandwidth.
- A library of useful functions and C++ classes.

More specifically, for TCP and UDP, the following parameters are possible to change:

- TCP window size
- TCP buffer length
- TCP max segment size
- UDP buffer size
- UDP and packet size
- UDP bandwidth

[1] More correctly it is datagram loss. By setting the datagram size to fit into a single packet, packet loss is measured.

Iperf is freeware and open source, which means that it can be modified and redistributed. The full license it is released under can be found here http://dast.nlanr.net/Projects/Iperf/ . Se Appendix A for installation details.

2.6.4.1 Usage

Iperf is a console application working under most major operating systems, including Microsoft Windows, Linux, Solaris and BSD variants. A console application has one great advantage in that it can be run from a shell terminal. To perform a test with Iperf you need two hosts. One acts as a server, and another as a client. To set Iperf in server mode, apply the –s switch (Iperf –s). To set it up in client mode, run Iperf –c serverhost. By default, Iperf works in TCP mode. To work in UDP mode, apply –u. This has to be done to both server and client. The client is the one who sends data. In TCP mode, the goal is to achieve maximal bit-rate. In UDP mode, one can set the client to send with a constant bit rate. Below is an output from an UDP client test. Iperf is here set to push UDP datagrams at the rate of 10 megabits per second. This is a stream that can simulate voice communication. The datagram size can be changed with the –l switch. In the output below we see that 11,9 megabytes was transmitted in ten seconds, that equals a rate of 10,0 megabits per second. The jitter was 0,002 ms and zero of 8504 datagrams was lost, although one was received out-of-order.

```
user@host:~$ iperf -c 128.39.203.23 -u -b 10M -i 1
------------------------------------------------------------
Client connecting to 128.39.203.23, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:   107 KByte (default)
------------------------------------------------------------
[  3] local 128.39.202.96 port 59483 connected with 128.39.203.23 port 5001
[  3]  0.0- 1.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  1.0- 2.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  2.0- 3.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  4.0- 5.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  5.0- 6.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  6.0- 7.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  7.0- 8.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  8.0- 9.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  9.0-10.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec
[  3] Sent 8505 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec  0.002 ms    0/ 8504 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

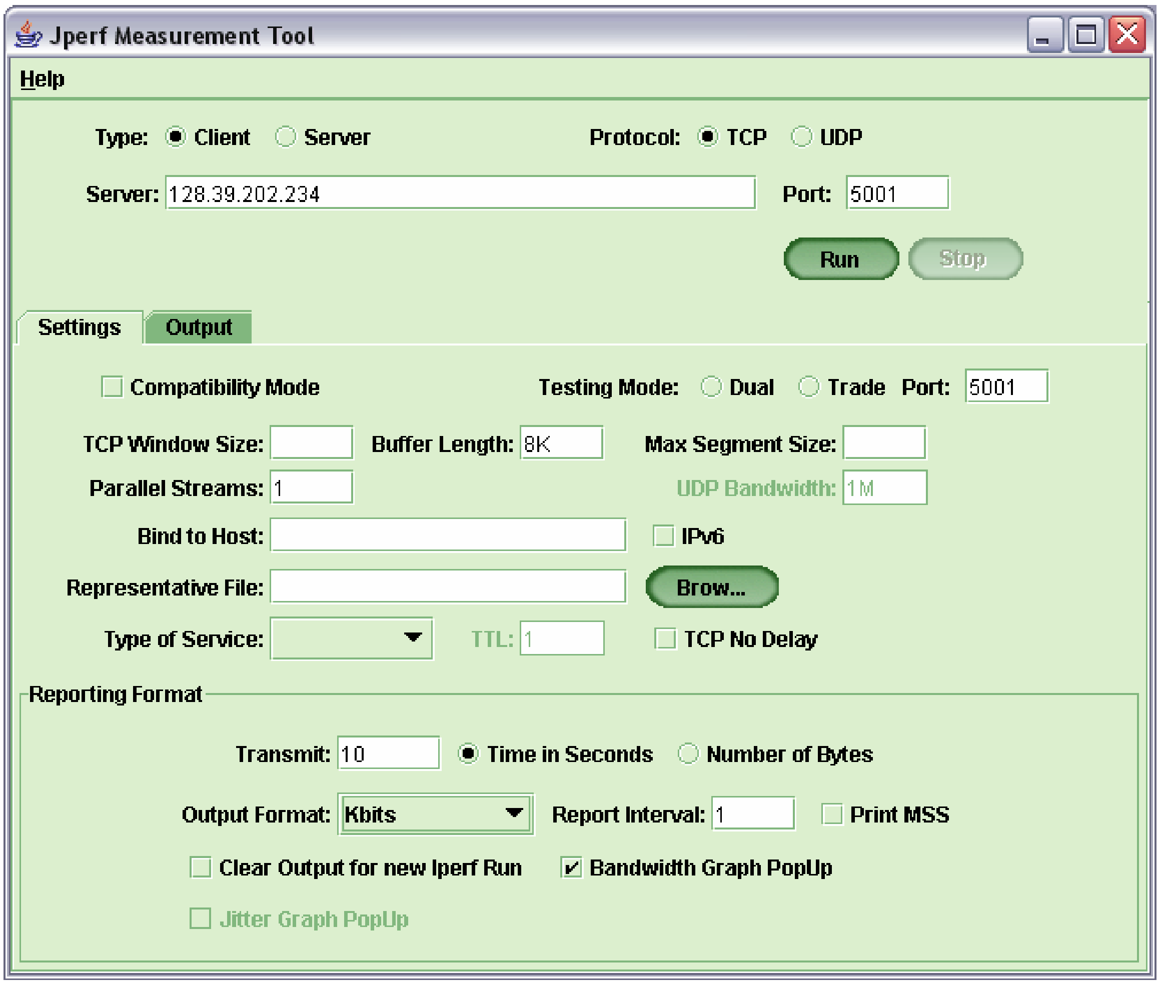
In the sample below we can see in the last line the average bit rate was 4,04 megabits per second over ten seconds. Especially changing the TCP window size and see how this affects the throughput is interesting.

```
user@host:~$ iperf -c 128.39.202.234  -i 2
------------------------------------------------------------
Client connecting to 128.39.202.234, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.11 port 1190 connected with 128.39.202.234 port 5001
[  3]  0.0- 2.0 sec    984 KBytes  4.03 Mbits/sec
[  3]  2.0- 4.0 sec  1.03 MBytes  4.33 Mbits/sec
[  3]  4.0- 6.0 sec    896 KBytes  3.67 Mbits/sec
[  3]  6.0- 8.0 sec    992 KBytes  4.06 Mbits/sec
[  3]  8.0-10.0 sec  1008 KBytes  4.13 Mbits/sec
[  3]  0.0-10.0 sec  4.83 MBytes  4.04 Mbits/sec
```

2.6.4.2 Jperf

Although Iperf has a pretty easy understandable user interface, a graphical user interface (GUI) written in Java, called Jperf, is also available at http://dast.nlanr.net/projects/jperf/ . Written in java, it is platform independent, and should work on all platforms that are supported by the SUN Java Runtime Environment.

2.6.4.3 User Interface


Jperf Measurement Tool

Help

Type: [✓ Client] Server   Protocol: [✓ TCP] UDP

Server: 128.39.202.234   Port: 5001

Run   Stop

Settings   Output

Compatibility Mode   Testing Mode: Dual Trade   Port: 5001

TCP Window Size: Buffer Length: 8K   Max Segment Size:

Parallel Streams: 1   UDP Bandwidth: 1M

Bind to Host: IPv6

Representative File: Brow...

Type of Service: TTL: 1   TCP No Delay

Reporting Format

Transmit: 10   [✓ Time in Seconds]   Number of Bytes

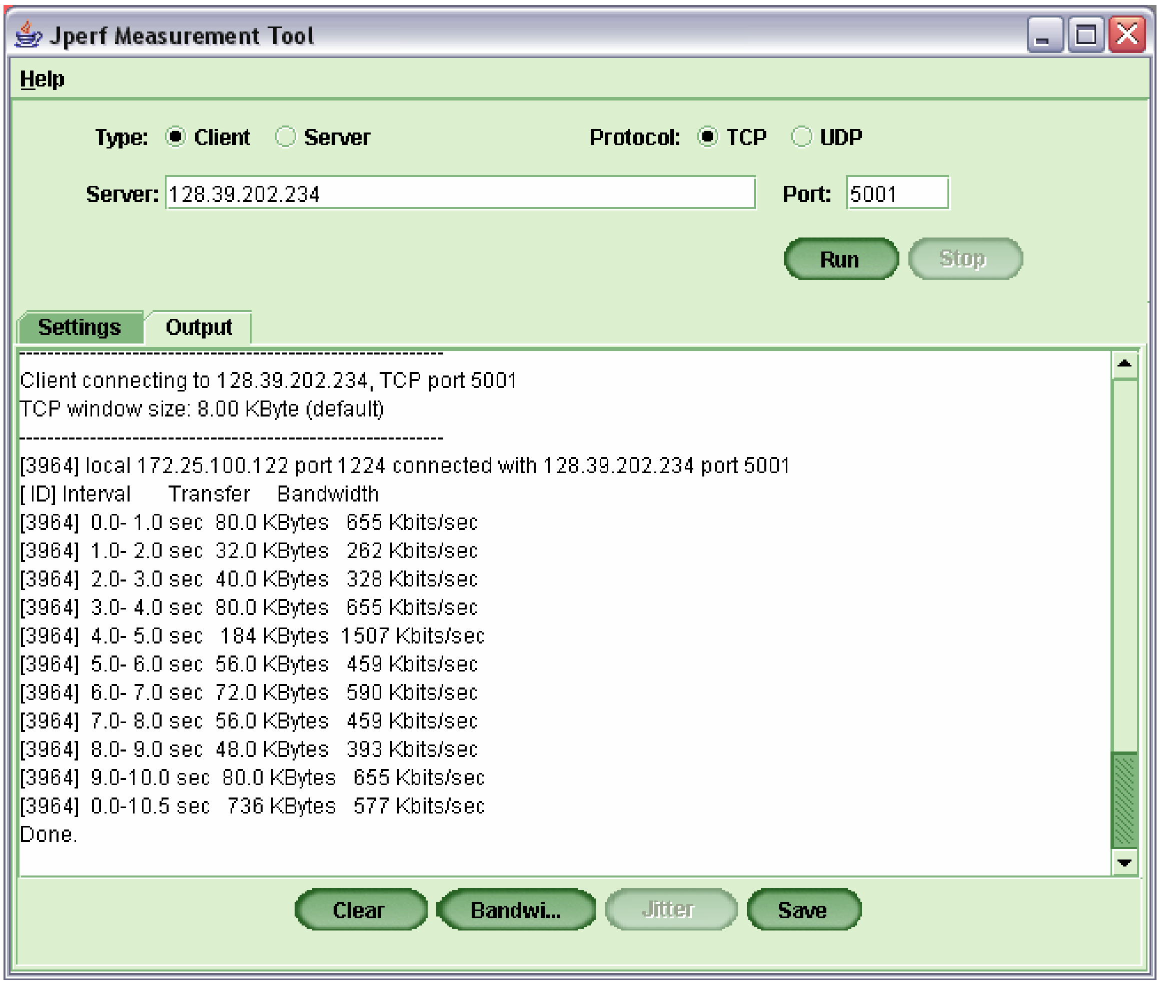
Output Format: Kbits   Report Interval: 1   Print MSS

Clear Output for new Iperf Run   [✓ Bandwidth Graph PopUp]

Jitter Graph PopUp

Figure 10 Jperf settings with easy configuration.

Figure 10 show how Jperf looks like. In addition to a nice looking GUI it provides easy configuration of all Iperf options, and it draws a bandwidth graph (Figure 11 and Figure 12). The server side plots a graph of Jitter too. Jperf is only a GUI, so it needs Iperf too on the system. As one can see in Figure 11, the actual output from Iperf is also visible in Jperf.



```
------------------------------------------------------------
Client connecting to 128.39.202.234, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[3964] local 172.25.100.122 port 1224 connected with 128.39.202.234 port 5001
[ ID] Interval       Transfer     Bandwidth
[3964]  0.0- 1.0 sec  80.0 KBytes   655 Kbits/sec
[3964]  1.0- 2.0 sec  32.0 KBytes   262 Kbits/sec
[3964]  2.0- 3.0 sec  40.0 KBytes   328 Kbits/sec
[3964]  3.0- 4.0 sec  80.0 KBytes   655 Kbits/sec
[3964]  4.0- 5.0 sec   184 KBytes  1507 Kbits/sec
[3964]  5.0- 6.0 sec  56.0 KBytes   459 Kbits/sec
[3964]  6.0- 7.0 sec  72.0 KBytes   590 Kbits/sec
[3964]  7.0- 8.0 sec  56.0 KBytes   459 Kbits/sec
[3964]  8.0- 9.0 sec  48.0 KBytes   393 Kbits/sec
[3964]  9.0-10.0 sec  80.0 KBytes   655 Kbits/sec
[3964]  0.0-10.5 sec   736 KBytes   577 Kbits/sec
Done.
```

Figure 11 Output from Iperf integrated in Jperf.

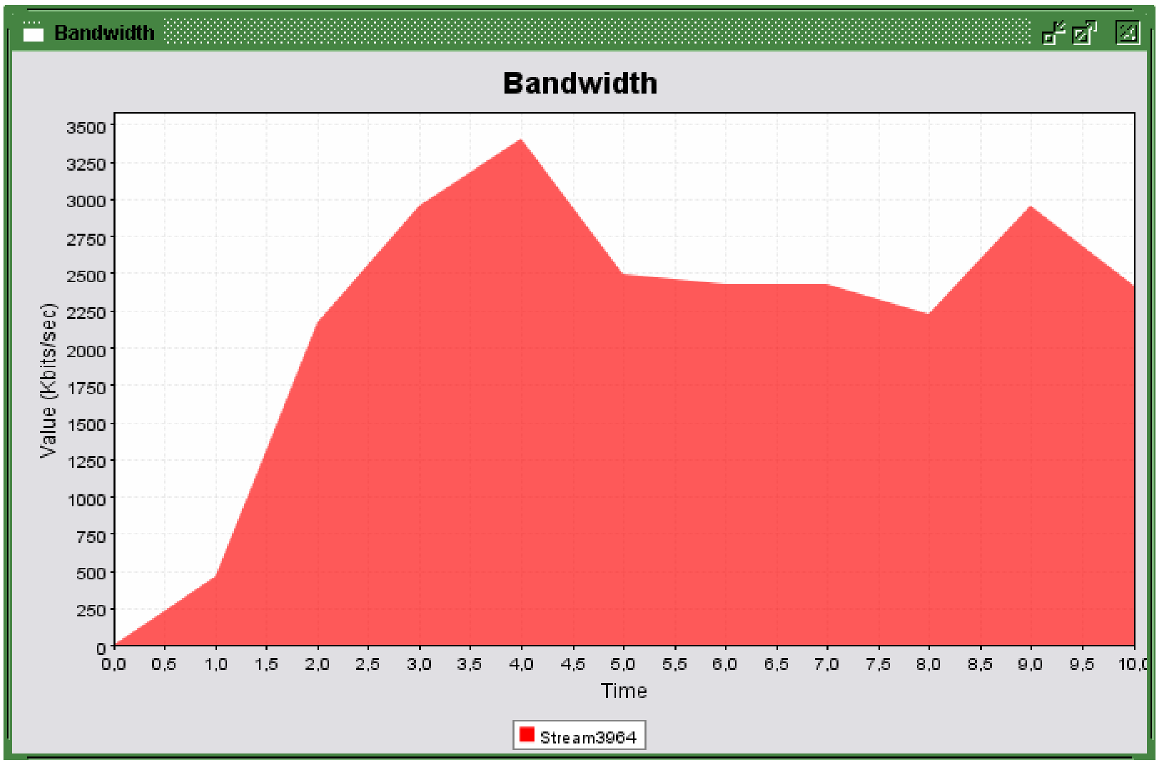


Figure 12 Bandwidth graph plotted in Jperf.

2.7 Related research

Research has been done on the performance of particularly TCP over wireless links such as GPRS. Michael Meyer states in [25] that due to the fact that GPRS offers a reliable link implemented with retransmissions, TCP performs what some would consider surprisingly well over GPRS. He concludes that the reliable link layer mostly provides fast enough retransmissions, so that TCP observes only packet delays, not packet losses. In the case that TCP times out, it may believe it has lost a packet, when it only is delayed. Such a timeout will enforce an unnecessary retransmission and TCP slow start, and will greatly degrade the performance.

In [26] Chakravorty accepts the fact that TCP performs fairly well over GPRS, but instead asks questions like

- Why do web users experience poor performance of HTTP in wireless wide area networks ?
- Even though TCP is relatively well tuned to perform efficiently in these environments, why is the performance of HTTP applications significantly worse?

Chakravorty concludes that there are a severe mismatch between TCP and HTTP performance. The achieved throughput in HTTP is sometimes as low as 70 % lower than the ideal downlink rate. This is greatly related to the “go-and-wait” behavior of default HTTP 1.0, where objects are requested one by one. E.g.

```
GET image 1
← Image 1 received.
GET image 2
← Image 2 received.
```

The high RTT of wireless links, and the fact that most websites are built of many small objects, is the main reason for this degradation. Measurements have shown that actual throughput with large files is good. Chakravorty shows that pipelining, an experimental feature of HTTP 1.1 will increase the performance greatly. Performance gain between 35 % – 56 % was achieved by using pipelining. With pipelining enabled, the client asks for several objects, before they are being received, without having to wait between, e.g.:

```
GET image 1
GET image 2
← Image 1 received.
← Image 2 received.
```

Unfortunately, most browsers do not support pipelining, neither do the web servers.

Due to the same reason that HTTP performs badly over GPRS, i.e. many small files, Chakravorty shows that compression of the payload gives no significant performance increase. In web pages with few large objects, compression will give better performance.

A session layer optimization they suggest is to open several TCP connections instead of the default two in HTTP 1.1. The number they found as optimal is six. This technique is in reality almost the same optimization as pipelining. Chakravorty also suggests DNS/URL-rewriting proxies that reduces the DNS lookup time by rewriting the clients GET requests such that the client need to perform at most one DNS lookup.

They show that the same issues are relevant in 3rd generation wireless networks.

Peter Benko, Gabor Malisco and Andreas Veres concludes in [27] regarding TCP tuning parameters that the TCP timestamp option only gives a minimal performance increase, while the selective acknowledge (SACK) option may increase the throughput by ten percent. They also conclude that a MSS of 1500 bytes is the optimal; this can be explained by less overhead compared to using smaller MSS, and faster slow start phase and recovery from losses.

2.7.1 Opera Mini

An application that tries to overcome the issues with HTTP over GPRS is Opera Mini from Opera Software. It can be found at http://www.opera.com/products/mobile/operamini/. The application consists of two parts, a proxy server part and a client part. The client is “the browser”. According to Wikipedia [28] it works as the following: when requesting a web page, the proxy server downloads the web page, reformats it for small mobile screens, compresses it and converts it to the Opera Binary Markup Language and sends it to the client on the phone. The size of the web page has then been reduced to about 70 % - 90 % of original size. The issues with the “go-and-wait” behavior of HTTP are also eliminated. This will give a better user experience for the user as the page is not only formatted for small screens, but also loads faster than with a normal wap/web browser.

Such an solution has many benefits, but also some issues, privacy and security is one. Opera Software, or the company running the proxy client, knows every step you do on the internet, and can even alter the information on the pages you visit.

2.8 Test classifications

Gómez et al claims in [29] that the trials or tests should be classified after their purpose. They define several classes. From an end-to-end performance view, the most important is the following:

- Basic service performance benchmarking. This test concentrates on basic services and KPIs like throughput, latency and RTT. Attach/detach signaling in GPRS should also be monitored here.
- Single service verification. These tests are done to measure performance on specific services. These could be WEB or FTP, but are often done on new services that need a certain performance to fulfill user experience. Such services are e.g. VoIP and streaming.
- Application performance audit. Here is the focus not on specific services, but to evaluate the overall performance of different services in the network, and the main objective is to understand the performance from a user’s point of view. Radio information is ignored, and one only looks at the user’s view of the performance. It does not need deep knowledge of the underlying technology and is well suited for benchmarking between different networks, or different technologies. In case problems are discovered, troubleshooting specific services should be considered.
- Troubleshooting. When performing a troubleshooting test, one should consider application KPIs and radio signal quality together with network KPIs and network elements.

An end-to-end test tool will fit straight into the second classification (Application performance audit) and in the first and to some extent the third. Troubleshooting however is not straightforward to do from an end-to-end tool since it involves having knowledge of information from the radio and core network that cannot be extracted from such an application.

2.8.1 Number of repetitions

The number of repetitions of each test is an important factor. One should never do a file download once, and be satisfied with the answer. To obtain any statistical correctness, Gómez et al [29] suggests ten repetitions as rule of thumb, and one should never perform less than five. One also has to decide if the testing should be performed when the network is busy, i.e. mid-day, or at night time when the load presumably is low. The latter can give the best view of max performance, and the first can give a good understanding on degradation in performance when the user interacts with other user's traffic. When comparing test results from two or more tests, it is very important to keep this in mind.

2.8.2 Mistakes when performing performance analysis

When performing performance analysis, there are several mistakes that can be done. Raj Jain has outlined some of them in [30]. He claims that most of them are not intentional, but they occur due to incompetence/lack of knowledge, oversights and misconception. We will take a look of some we believe are very important when developing, and using an end-to-end test tool:

No Goals
Before starting the measurement a goal of what is to be achieved is important. Is it to benchmark two systems? To discover implementation problems in the system? It may seem obvious to state one or more goals, but too often the opposite is the reality. Goals relevant for an end-to-end tool may be "Benchmark two network operators FTP throughput and RTT".

Biased Goals
Setting no goals is bad; setting biased goals is just as bad. If the goal is to show that "our system is better than theirs", the problem becomes finding those parameters, KPIs, that favors our system, rather than those KPIs that can be used for comparing and benchmarking the two systems. It is a must to be unbiased. An example of a biased goal is "Why is Netcom better than Telenor (or vice versa)"

Unsystematic Approach
Arbitrarily selections of parameters, factors and metrics may lead to inaccurate conclusions. A systematic approach to identify a complete set of goals and parameters is needed.

Incorrect Performance Indicators
It is tempting to select those indicators that easily can be measured. The important is however to select those parameters that has relevance for measuring the performance of the measured object. Which parameter to measure depends on the service that is measured. For FTP it may be throughput or data connection establishment failure rate.

Overlooking Important Parameters

One should make a complete list of system and workload characteristics that affect the performance of the system. These are called parameters, and seen from an end-to-end test tool view, they may be

- The phone used and its capabilities (Multislot class etc)
- Cellular network used, coding scheme
- Number of active users in cell.
- TCP parameters

Not all may be documented, due to limitations of an end-to-end tool.

Ignoring Significant Factors
Parameters that are varied and affect the performance when they are varied are called factors. It is important to select the right factors and not ignoring significant ones. Factors that may be varied by the end user should be preferred.

No Analysis
The job is not finished when the data is collected. The measured data needs to be analyzed by personnel with analysis expertise.

Improper Presentation of Results
Help in decision taking is the aim of all performance analysis. Any analysis that do not produce any results is a failure. It is also a failure if the decision makers do not understand the results. Words, pictures and graphs are important to make sure the results are clear for the decision makers.

2.9 Key Performance Indicators

G. Gómez and R. Sánchez have collected several articles about data service performance and optimizations in 2G/3G in [29]. The articles in this book provide very insightful information and theories about issues in cellular networks affecting the user performance. The following sub-chapters will present ideas from this book.

2.9.1 Introduction

Key performance indicators, or simply KPI, are indicators which are particularly important for a services performance. First one has to define the KPIs associated to a service, and then one has to determine how to measure the KPIs. For instance, in VoIP, delay and jitter is two KPIs. These two factors play an important role in VoIP performance. In the telecommunication world, the three big KPIs are [31]:

- Accessibility
- Retainability
- Quality

Accessibility has to do with the users being able to set up a call and access radio resources. Retainability covers the ability to keep up a call, and quality deals with how good the connection is. Quality can be measured with speech frame error rate. They can all be measured at the radio level and easily be transformed into a measurement of service quality.

2.9.2 Key performance indicators in packet switched networks

In packet switched networks the connection between network metrics and user performance is often not as easily seen as in circuit switched networks. The difficulty to see this can often be attributed to the several layers of packet switched communication. A problem at the low link layer, e.g. unreliable link, may only bee seen as high delays in the upper layers. Another fact is that all applications may not suffer from the same degradations. While web browsing suffers when the delay becomes too high, a MMS can suffer a delay of ten seconds without degrading the user experience and satisfaction. There are some indicators that affect end-to-end performance that need some extra attention. The end user's experience and satisfaction of the various services he uses will depend heavily on these parameters. Some parameters are important in some services, while they are not as important in others.

2.9.2.1 Delay

There are different types of delay; in tele and data communication especially three types of delay are commonly described, these are Round Trip Time, access delay and jitter. Following is a description of each of these.

Round Trip Time

Round Trip Time (RTT) or Round Trip Latency, or simply Response Time is the time from the sending of a packet to it is received again. The RTT is depending on the distance between the sites and the delay in each hop. A limitation with the RTT is that asynchronous links can make the packet travel fast in one direction and slow in the other. One cannot detect this with the RTT, as the RTT is the time from an echo request is sent, to an echo response is received. This is illustrated in Figure 13, the RTT is the sum of T1 + T2. Processing delays in the local and remote machine is not taken into consideration in this figure.

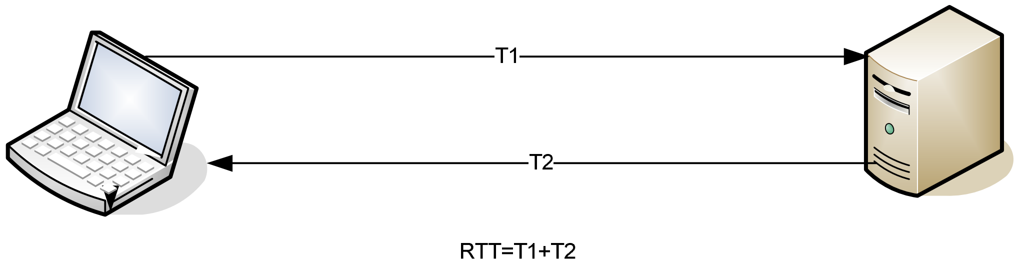


Figure 13 Illustration of the round trip time

The most common way to measure the RTT between two sites, is with the "Ping" application which sends Internet Control Message Protocol (ICMP) "echo request" packets to the target host and listens for "echo response" packets. Some version of the Ping application exists in all major operating systems, like Microsoft Windows, Linux or UNIX variants. In addition to RTT computation, Ping will also estimate the packet loss rate. The most common use of ping however, is perhaps to decide if the host computer has network access, and if the target host has network access. Following is a sample output from a GNU/Linux version of ping done on a computer connected to the internet with cable modem.

```
user@host:~$ ping -c 10 www.vg.no
PING www.vg.no (193.69.165.21) 56(84) bytes of data.
64 bytes from 193.69.165.21: icmp_seq=1 ttl=248 time=28.5 ms
64 bytes from 193.69.165.21: icmp_seq=2 ttl=248 time=20.6 ms
64 bytes from 193.69.165.21: icmp_seq=3 ttl=248 time=21.1 ms
64 bytes from 193.69.165.21: icmp_seq=4 ttl=248 time=31.6 ms
64 bytes from 193.69.165.21: icmp_seq=5 ttl=248 time=22.8 ms
64 bytes from 193.69.165.21: icmp_seq=6 ttl=248 time=21.8 ms
64 bytes from 193.69.165.21: icmp_seq=7 ttl=248 time=22.5 ms
64 bytes from 193.69.165.21: icmp_seq=8 ttl=248 time=51.4 ms
64 bytes from 193.69.165.21: icmp_seq=9 ttl=248 time=28.1 ms

--- www.vg.no ping statistics ---
10 packets transmitted, 9 received, 10% packet loss, time
14231ms
rtt min/avg/max/mdev = 20.696/27.662/51.466/9.176 ms
```

Time is the RTT. We can se the smallest RTT was 20,696, average was 27,662 and maximum was 51,466 ms. One packet out of ten was lost on the way, resulting in 10 % packet loss.

Generally we can say that it is a good thing with the RTT as low as possible, however different services has different requirements for the RTT to give a good user experience. A RTT of 28,5 ms must be considered as relatively low, a RTT of 2,85 seconds on the other side, would highly degrade the performance. Transmission protocols like TCP which relies on acknowledging packets before the next one can be transmitted, is heavily affected by the end-to-end latency. This can make TCP troublesome in high latency wireless networks.

One-way latency or delay is the time it takes for a packet to get from one host to another. This is more difficult to measure than two-way latency since it need synchronized clocks. However, the one-way latency can provide important information as it deals with the problems of asynchronous links, i.e. the delay in both directions is not equal.

Delay is caused by propagation, simply the time for a signal to travel from a point to another in i.e. a wire. Transmission delay is caused by the medium, i.e. a large packet will take longer time than a short. Delay is also caused by routers which are examining the packet headers and changing TTL-fields etch.

Access delay
Access delay is the delay which appears when communication is established. When turning on your mobile phone and opening the web-browser for the first time, one is experiencing some access delay when the phone is attaching to the gprs network. This delay is normally in the range of a few seconds. An access delay of a few minutes would make it cumbersome.

Jitter
In IP-networking particularly, jitter is the variance in delay of packets. All IP-networks has some jitter. In all cases (at least near to), a small jitter is preferred.

Buffering streaming media for instance, is an approach used to minimize the unwanted effect of jittering. The ideal buffer size is so that the most delayed packet can be played at once. This buffer size is of course impossible to implement in real life, since one cannot know the future and how much delayed that packet will be, and thus not how large the buffer has to be.

2.9.2.2 Bandwidth and Throughput

The rate which the network is able to send or receive data is the throughput. Throughput is the bit rate and is limited by the capacity of the network channel. Throughput is mostly measured in bits per second with one kilobit equal 1000 bits, in contrast to the size of a file which mostly is measured in bytes, and with one kilobyte equal to 1024 bytes. The potential, or theoretical, throughput of a network is called the bandwidth, while the throughput is the actual number. The reason bandwidth is more often quoted than throughput is probably because it is easier to calculate. Throughput is difficult to calculate since it is dependent on many different variables, i.e. packet loss [32] and transport protocol. This means that user data throughput is higher with the use of UDP over IP than TCP over IP, which is the reason services that needs high throughput, like video streaming, use UDP. Bandwidth and throughput are also often quoted as the same.

There is often a difference between what can be measured as the average throughput, which is what you can expect to get when downloading a file, and the max peak bit-rate which is the rate one can achieve for shorter periods of time.

Below is an example of and UDP throughput test with iperf. In this example, iperf is running as server on host 128.39.203.23. The client sends UDP datagrams at the rate of 10 megabits per second. As we can see, the server only received one datagram out of 8505 out-of-order, and no one was lost. The network between the two hosts was fully capable of this bit-rate.

```
user@host:~$ iperf -c 128.39.203.23 -u -b 10M -i 1
------------------------------------------------------------
Client connecting to 128.39.203.23, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:   107 KByte (default)
------------------------------------------------------------
[  3] local 128.39.202.96 port 59483 connected with 128.39.203.23 port 5001
[  3]  0.0- 1.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  1.0- 2.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  2.0- 3.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  4.0- 5.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  5.0- 6.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  6.0- 7.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  7.0- 8.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  8.0- 9.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  9.0-10.0 sec  1.19 MBytes  10.0 Mbits/sec
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec
[  3] Sent 8505 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec  0.002 ms    0/ 8504 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

2.9.2.3 Reliability

What happens if packets and data are lost when they are transmitted? The error rate is the probability that packets are lost or received with errors. The reliability of the link layer and network layer will affect throughput. Packet losses in the network layer when TCP are used will cause retransmission and make TCP slow down its transmission, with low throughput as its final result. Packet loss with video streaming and UDP can lower the quality of the picture and cause glitches in the video. GPRS provides, unlike many other wireless links, a reliable link in the radio link control (RLC) with its own Automatic Repeat Request (ARQ) [25]. Radio blocks on this link are much smaller than usual maximum segment sizes for TCP, and allow several retransmissions before TCP timeout. This means that TCP see packet delays, rather than packet loss. Reliability on the lower layers will often appear as low throughput or latency from the user’s point of view. Cell reselection is another action in the radio layer that may make the IP-based communication halt for some seconds. Although the radio network is able to buffer the user data until radio communication is re-established, it may result in TCP timeout and retransmission. Hence, no user data is actually lost, only so delayed that TCP thinks it is lost and retransmits it [31]. TCP optimizations to quickly adopt new time out values are highly interesting in such wireless environments.

In IP communication the two most used transport protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP provides a reliable connection between two hosts. If a segment is lost on its way, the TCP protocol will retransmit the segment. TCP’s congestion and flow control will also make TCP slow down its transmission rate. UDP however, will continue to send at the specified bit rate whether the receiver and network can handle it or not. Sometime one has to compromise between performance and reliability. 100 % correct data is important in email and web browsing, but not in video or voice calls. In the two last cases, performance and low delay are the most important factor. Whether one packet is lost or not, is not that important.

2.9.2.4 Availability

The probability that a service is available when the end user requires it, is called availability or accessibility. Traditionally, availability will be affected by the network uptime and system and application stability at the service provider. In a wireless environment, factors like signal coverage will be important [31].

2.9.3 Classifying the KPIs

From the way KPIs are measured one can divide them into

- Passive KPIs, which are measured directly in the networks management system without any active involvement. Usually measured in intervals of e.g. one hour.
- Active KPIs, which are measured by a human operator with various test and monitoring tools. They give a high level of detail, but lacks statistical information, and the measuring must commonly be repeated several times.

Further, one can divide the KPIs into the following, depending on what they focus on:

- Network KPIs. Most of these KPIs are passive, as they deal with how the mobile communication system is performing from the systems point of view. This includes radio resource sharing and assigning, mobility managing etc. These KPIs are usually monitored in order to detect network bottlenecks.
- Service-based KPIs. These KPIs provide understanding of how the specific service is performing from a user's point of view. They address the performance the end-user is experiencing, something the network KPIs do not do. For instance, a service based KPI for HTTP/Web browsing defines and characterizes the KPIs and how they affect that service in particular. Service based KPIs are mostly active. E.g. a benchmark tool to measure file downloads can be used to measure service based KPIs.

Most interesting from an end-to-end test tool's point of view is the service based KPIs. The network KPIs is hard to measure from such an application. There are however some active KPIs that have been labeled as network KPIs in [31], that can be monitored, such as GPRS attach and activate delay, and failure rates.

2.9.3.1 Service based KPIs

Those factors, or indicators, that specifically affect how a specific service is performing, are service based KPIs. This means that there are different KPIs for HTTP web browsing or MMS or ftp file transfer. They are in almost all cases active, and easily measured and monitored with the appropriate tool [31]. In the following we will address some of the KPIs for major services in packet switched networks..

FTP

For the end user to be satisfied with the file transfer protocol, some criteria may have to be fulfilled. First of all, he must be able to connect to the server within a reasonable time. When he has connected and set up the control connection, he must be able to establish a data connection in either passive or active mode. Firewall rules may prohibit this. When both control and data connection is established, the reliability of the data connection is important. A file download which fails half wards, is annoying for both the user, and significantly increases the time it takes to get the file. This leads us to the throughput. All delays and connection failures will degrade the throughput. Also, of course, will the actual bit rate the link is able to provide when download is in progress affect the throughput. The KPIs are:

- FTP start-up failure rate: The percentage of cases the ftp client is not able to connect to the server or establish the ftp data connection. This can be the result of several problems at the TCP/IP level or firewall configurations.
- FTP abort rate: This is the number of times in which the FTP transfer is started but fails to finish. Could be related to several issues, client or server application error, IP-transport error, congestion etc.
- FTP throughput: The average throughput for a file transfer. This measure how "fast" the download (or upload) was. [31]

HTTP / Web browsing

Some of the same KPIs that apply to FTP also apply here. In bottom, both these services are about transmitting files. However, from an end-users view, there are certain things that in most cases will give a better experience. For instance, some data takes longer time to load than other. If you were to wait until all data (text + pictures + applets) were loaded before rendering the page, this will in most cases take some time, especially on low bit rate connections. If you get the text up and readable early, this will in most cases give a better user experience, than if you have to wait for a longer time, and only see the page when it is fully loaded and rendered. A summarized list of KPIs is:

- Access failure rate: When the HTTP client is not able to connect to the server, either because of application errors or transport level errors like congestion.
- Abort rate. Number of times when a HTTP transmission is started but aborted before the page is fully loaded. Can be due to many as the same errors as failure rate, application error or congestion.
- Access time. The time it takes from the user requests the page, i.e. clicks on the link, to the user see the first contents of the page. Text is usually first loaded.
- Access time to text. The time it takes from the user requests the page, i.e. clicks on the link, to the user see the full text (the .html file) but not the images etc.
- Throughput/delay. The time, or delay, from the user requests the page to it is fully loaded [31].

Multimedia Messaging Service (MMS)

The MMS is a service somewhat similar to SMS but with a lot of enhancements when it comes to features and content. With MMS one is able to send pictures, video clips and sound. It is transferred packet switched, unlike SMS which mostly is transferred via CS signaling. The most important thing from a user’s point of view is that it is delivered. Some delivery delay can in most cases be tolerated, although there is always a positive thing to get it delivered as fast as possible. The KPIs are:

- Send/retrieve failure rate. The percentage of failed delivery of MMS.
- Send/receive throughput. The send/send receive throughput/data rate in bits per second.
- Send/receive delay: The delay from the when the WAP connect event takes place to the moment when WAP disconnect takes place for the sender/receiver.
- End-to-end delay. The time from WAP connect at the sender to WAP disconnect at the receiver.
- Notification delay. The time from WAP disconnect at the sender to WAP connect at the receiver. [31].

Ping

Has only one KPI, the round trip time, or shortly RTT. It is the time it takes for a packet of different sizes to go from one host to another and back again. It measures the latency of the network. It sends “echo request” ICMP packets to the host, and waits for “echo response” packages.

Common parameters

A parameter that applies to all services is the delay from the request is started to it is finished. One wants things to go as fast as possible. Throughput at transport layer, TCP, is a key word in FTP, but good TCP throughput do not necessary mean good throughput in HTTP. FTP is often about transferring a small amount of large files, while HTTP is about transferring many small files.

2.10 Factors affecting the end-to-end performance

In [33] G. Gómez et al states that end-user performance is affected by every protocol layer and network element from one end point to another endpoint. This means that every layer starting from the bottom will degrade the performance of the link. The throughput of the physical layer is used as the starting point, and the degradation of performance on each layer above in the protocol stack is estimated. [33] Describes further that the factors that produce a degradation of the link level throughput can be divided into two groups; data link level and upper layer effects. In this chapter we will look into these effects.

2.10.1 Data Link Effects

Those factors that degrade the performance depending on radio coverage, interference and resource sharing are called data link effects. The performance after these degradations is called data link throughput, and is the final throughput offered by the Radio Access Network to the upper layers. Data link throughput and latency can be calculated based only on the network itself.

2.10.1.1 Data Link Effects in GPRS and EGPRS/EDGE

GPRS is affected by the interference levels in the frequency planning and delays occurring with transmission times between BTS and BSC, Radio Resource Management (RRM) functions and radio protocol functions [33]. Seen from the perspective of data link effects, one can define the following performance indicators:

Peak throughput

The throughput delivered to the LLC layer without RLC/MAC headers depends on the used modulation and coding scheme (CS/MCS). The peak throughput given for each coding scheme in GPRS can be seen in table 1.

Coding scheme	*1 slots*	*2 slots*	*3 slots*	*4 slots*	*5 slots*	*6 slots*	*7 slots*	*8 slots*
CS-1	9,05	18,2	27,15	36,2	45,25	54,3	63,35	72,4
CS-2	13,4	26,8	40,2	53,6	67	80,4	93,8	107,2
CS-3	15,6	31,2	46,8	62,4	78	93,6	109,2	124,8
CS-4	21,4	42,8	64,2	85,6	107	128,4	149,8	171,2

Table 1 GPRS throughput at LLC layer [2].

Timeslot capacity

The timeslot capacity is the available throughput in a timeslot (TSL) after including the effects of interference and RLC retransmissions if RLC acknowledged mode is used. There are several factors that affect TSL capacity, such as radio link quality, network planning (frequency reuse) and configuration, the layer where GPRS is allocated (BCCH hopping, non-hopping) and Effective Frequency Load (EFL).

Reduction Factor

TSLs are shared between several connections. The reduction factor (RF) includes this fact, that it is a shared medium. Network load and dimensioning conditions affects the RF and it depends on several factors:

- GPRS allocation size: How many TSL's that are reserved for GPRS and how many that is shared between voice and data is important to prevent high GPRS blocking, and thus high RF.
- CS load and pre-emption criteria: The priority given to CS and PS traffic is important in preventing RF. If CS traffic is given higher priority than PS, high CS load will degrade PS traffic too (increase RF for PS).
- Terminal capability, Multislot class: Terminals which supports several TSLs are capable of getting higher bit rates from the system. A high-end phone, like Nokia N70, is typically capable of using 4 timeslots for downlink and 2 for uplink, i.e. switching between 4+1 and 3+2 since the maximum number of timeslots that can be used simultaneously usually is set at 5 from the network operators.
- RRM scheme: The job of the RRM is to take care of minimizing the TSL sharing when doing channel allocations. It ensures that the terminal is connected to the best cell, and may support QoS mechanisms that can prioritize certain flows. This means that other flows may be queued. I.e. high priority flows will utilize the radio resources for a longer period of time.

RLC signaling

Whenever data needs to be sent through the radio interface, a Temporary Block Flow (TBF) has to be established. The TBF may cause some delay when it is being established, typically in the area 300 – 600 ms, and thus TBFs being released and established continuously may cause performance degrading. The throughput will also be affected given the fact that the RLC control blocks used for signaling shares the same radio resources as that of the data RLC blocks. In RLC acknowledged mode, ACKs for uplink data is sent. Bursty uplink data, such as TCP ACKs, that caused frequent establishing and releasing of TBFs, made way for the introducing of an enhancement that tries to eliminate this delay. The idea is that TBFs are not immediately released when there is no more data to send, but a timer, typically set to 1-5 seconds is started, and when this timer is finished, the TBF is released.

One wireless event that may affect upper layer behavior is one originating from the mobility issue. Cell reselection will cause some delay, in the level of seconds. Various new enhancements have been introduced to lower this delay, three such enhancements are Network Controlled Cell Reselection (NCCR) and Network Assisted Cell Change (NACC) and to use the Packet Common Control Channels (PCCCH) for signaling. Using these together may minimize the cell change delay too around 500 ms. [33]

RLC retransmissions will cause higher delay and jitter; this can affect upper layer protocols like TCP.

2.10.1.2 Data Link Effects in WCDMA

For transferring of user data, the physical layer of WCDMA provides various transport channels. These are Common Channels, Dedicated Channels (DCH) and Downlink Shared Channels (DSCH). Common channels use an explicit addressing of the mobile phone involved in the data transfer. These channels are FACH, RACH and CPCH.

- Forward Access Channel (FACH), downlink channel intended to carry control information to a UE known to a location in the cell. All UE’s must be able to decode its information. It can carry packet data information for any user in the cell.
- Random Access Channel (RACH), uplink channel for signaling, but also capable of transmitting user data. There is a collision risk in the RACH, and it is received in the entire cell.
- Common Packet Channel (CPCH), uplink channel intended for packet data transmission. It should be viewed as an extension of the RACH. The CPCH is associated with a DCH on the downlink.

The only type of dedicated channel in WCDMA is the DCH. Since it is dedicated, no explicit addressing of the target is necessary, and it can carry all user data and higher layer signaling intended for a given user in both uplink and downlink.

DSCH is a channel shared by several UEs. It is a downlink channel, and time multiplexing is used to share all the resources between the users sharing the channel.

To indicate which UE will receive data in each time interval, one DSCH is assigned to one or several downlink DCHs.

The bit rates in the common channels are pretty low, typically 8-32 kbps, and for this reason they are not used for transmission of large amount of data. They do not use fast power control (one exception is CPCH) and soft handover. The DCH uses both fast power control and soft handover and is much more efficient from a power transmission point of view, and supports mostly a much wider range of bit rates, typically in the range 8-512 kbps. A drawback with the DCH however, is that its establishments introduces some delay, and thus transmission of small packets could be faster carried out over a common channel. The DSCH typically provides a high bit rate, 64 – 1512 kbps, but is shared between all the users sharing the channel. I t also requires an allocation of large amount of power and code resources. Its power transmission is not as efficient as DCH’s and its usage is only convenient for code shortage situations.

When a UE is turned on, it has to select a Public Land Mobile Network (PLMN) to connect to. When this is done, it is in idle mode. The mobile goes from idle to connected mode when a Radio Resource Control (RRC) connection is established between the UTRAN and the mobile. In connected mode, the UE can be in four different states, CELL_DCH, which is the only state allowed when a DCH is established, in CELL_FACH, in which the UE continuously monitors the FACH in the downlink. The third state is the CELL_PCH in which uplink activity is not possible, and the UE uses discontinuous reception to monitor the paging channel. The last state is the URA_PCH which is very similar to CELL_PCH. In URA_PCH the location of the UE is known on UTRAN registration area (URA) level instead of cell niveau.

CELL_DCH and CELL_FACH are the only two states that allow user data transmission. When packet data transmission is to be started, it is UTRAN’s responsibility to decide which channel should be allocated. To select the channel type, either dedicated or common, traffic volume reporting is used. The MAC layer receives information from the RLC layer about its Buffer Occupancy (BO). If the BO exceeds a specific threshold, UTRAN selects DCH or DSCH as the transport channel. To do this, a capacity request is sent to the Packet Scheduling functionality. A DCH is established between the UE and the UTRAN if the scheduler grants the capacity request. If, however, the BO does not exceed the threshold, a common channel is used for transmission. The establishment of a DCH introduces a delay, and not unlike the idea of not releasing TBF’s in GPRS immediately, the DCH is not released immediately after the RLC buffer runs empty. Just like with what can be done with TBFs in GPRS, a release timer is started, and if the timer expires, the DCH is released and the UE returns to CELL_FACH state.

This switching between states in the RRC connection will affect the upper layers. I.e. in an ftp session, the packets transmitted in the initial TCP connection establishment are typically transferred over common channels, while when the data is to be transferred, a DCH is established. The capacity request processing and the DCH establishment introduce a delay which has two immediate effects [33]:

1. Delays the overall file transfer.
2. Raises the Round-Trip-Time (RTT).

The RTT increase indirectly affects TCP performance, the DCH release timer will directly influence other consecutive downloads, such as HTTP/WEB browsing. The following download will avoid the DCH establishment delay if the release timer is high enough.

Sources of throughput degradation for DCHs in WCDMA

Factors that determine the end-user throughput in WCDMA is the following:

Peak throughput

The maximum bit rate that the physical layer can provide under ideal signal quality conditions is the peak throughput. The following determines the peak throughput in WCDMA:

- Chip rate
- Spreading factor (SF) of the spreading code used
- Coding rate

Resource multiplexing among users

All the users utilizing the same frequency in the same cell have to share the resources. These resources are the air interface capacity and the physical resources, e.g. the spreading codes of the code tree. The Packet Scheduler and Resource Manager manage these resources. There are three typical scheduling methods for distributing the available loads among the users:

- Fair resource, which means all users gets equal resource. The user location and signal strength will determine the effective bit rate.
- Fair throughput, which means all users get the same throughput. The users with bad signal strength at the edge of the cell will consume a large amount of resources.
- C/I scheduling, which means users with good radio signal strength will be prioritized at the expense of users with bad signal. Users with poor radio conditions will be offered a lower amount of resources.

Block error rate (BLER)

A restrictive BLER will need more power, and since power is a limited resource, will lower the systems overall throughput. If BLER is too high, many retransmissions are needed, which also consume resources and degrade end-user performance. Since the link level throughput that can be achieved by an individual is a function of the allocated resources, interference and BLER, adjusting the right BLER is important. It can be viewed as a trade-off between quality and system capacity.

RLC signaling

To ensure error free delivery of data, RLC signaling in the form of control information is transmitted in acknowledged mode. Delays from RLC retransmission may affect upper layer performance, i.e. TCP transport. [34]

2.10.2 The upper layer effects

Those degradation factors that depend on the transport and application protocols are called the upper layer effects. These factors degrades the performance independently of the network that is used, however, there are some factors on the link layer that more contribute to upper layer effects than other, most important data link layer throughput and latency. I.e. in most cases, a high latency in the data link layer will give more upper layer effects degradation than in those with low latency. In this section we will look at the two most common used transport protocols, starting with the by far most used, the Transmission Control Protocol.

2.10.2.1 TCP performance in wireless networks

TCP is by far the most used transport protocol on the Internet. It provides reliable data transport with flow and congestion control. It was originally developed and designed for wired networks. In wired networks the main source of packet loss is network congestion. Wireless networks have another profile. Delays, jitter and packet loss not due to congestion is more frequent. Since TCP is very sensitive to factors such as delay, throughput and packet loss, it will perform somewhat different and performance may degrade in wireless networks compared to in traditionally wired networks. [35]

2.10.2.2 TCP issues in wireless networks

The most important factors that affect TCP performance in wireless networks is the following:

Latency

TCP slow start greatly reduces the performance of TCP when RTT is high, as it often is in wireless and cellular networks, at least compared to wired networks. The result of large delays is large TCP connection establishment times and slow recoveries form TCP slow start mechanisms.

Packet Losses

Wireless networks, unlike in wired networks where packet losses mostly are due to buffer overflows, have errors in the radio link. RLC acknowledged mode provides a reliable link in GPRS, with in-order delivery, but this is not always used. Packets could also be lost due to mobility issues like handover. TCP however, assume packet loss is due to congestion, as for the most is the case in wired networks. TCP reacts to packet loss by limiting its transmission rate by reducing the TCP congestion window and slow start threshold to the half.

Delay Spikes

Even though TCP updates its retransmission timeout value (RTO) based on the acknowledgments it is receiving and the RTT, a sudden delay may cause problems. When this happens, we have a delay spike. Such a delay may be a result of several wireless events, such as:

- RLC retransmissions, due to bad quality in link because of low signal strength/radio coverage. RLC acknowledge mode delivers in-order blocks, thus several TCP segments may be buffered and delivered in a burst when all RLC retransmissions necessary are done [25]

- Pause in transmission during cell changing.
- Blocking by high-priority traffic. Often data traffic is given lower priority than e.g. CS voice calls.

High delays can cause TCP timeout, i.e. TCP believes the packet is lost, and begins retransmission and begins a new slow start phase. The TCP option TimeStamp is recommended in wireless applications and deals with managing sudden delay spikes. In short, the TimeStamp makes it possible to estimate the RTT per segment, and not only per window as without. This makes the RTO adapt to the RTT quicker and make TCP more capable of managing delay spikes without timeout.

Variable data rates

Sudden changes in the bit rate can affect TCP. For instance, the number of users connected to one cell affects the available bandwidth. The distance from the base station can affect the bandwidth due to radio coverage. Even if TCP tries to adapt to the link bit rate, sudden changes in capacity may cause TCP to under use the link capacity or use a too high transmission window. Sudden reduction in link capacity can cause packet losses, and then cause TCP to on go into “slow start”, which is very slow given the high RTT

Asymmetry

GPRS and 3G have asymmetric uplink and downlink, i.e. higher bandwidth downlink than uplink. Battery power consumption is one factor limiting the uplink. However, the difference should be in the range such that a technique like ACK congestion control is unnecessary.

Other

The fact that TCP uses the radio interface in both directions because of ACKs for every segment is very important to take into considerations. The bit rate is not as high in the reverse direction, but to optimize TCP performance, one has to keep this in mind. When ACKs get lost, or is delayed, it will reduce TCP performance. At least the following issues must be recognized:

- ACKs can be delayed or get lost in wireless networks.
- The ACKs arrival rate determines TCP sending rate. ACK traffic is usually bursty, and establishment of radio resources of each ACK will often produce unhealthy delays.
- In GPRS ACKs produce signaling in lower layers that consume resources in the opposite direction.

2.10.2.3 Bandwidth delay product

The ideal state of TCP is when the sender injects its segment into the TCP pipe at the same rate as the receiver removes it from the pipe. The number of ACKs making its return trip is the same as the number of segments in the pipe. The BDP is particularly important in window based transport protocols like TCP. The BDP is the amount of data that has been transmitted but not yet received by the receiver, i.e. it defines the amount of data the sender can transmit before it must wait for ACKs, and the amount of data the sender must store in its buffer in case of retransmissions. The BDP is defined in [36] as

```
BDP (bits) = total_available_bandwidth (bits/sec) x round_trip_time (sec)
                                    or
BDP (bytes) = total_available_bandwidth (KBytes/sec) x round_trip_time (ms)
```

It is suggested in [33] and [37] that the congestion window should be slightly higher than the BDP and that it is really important that the advertised window in the receiver side is larger than BDP to not limit congestion window.

Round Trip Time's influence on the performance

The Round Trip Time, normally measured with the "Ping" application, is one of the most important sources for performance degradation. TCP uses a three way handshake, i.e. a TCP connection establishment will take approximately 1,5 x RTT. The RTT in wireless networks is almost without exceptions higher than in wired networks. The congestion window in slow start can only be raised in a per RTT basis, i.e. the higher the RTT, the slower the congestion window will raise, because the ACKs need longer time to get to the sender. It has been shown in [38] that for EDGE a terminal can suffer up to 7% throughput degradation per additional 100ms in the RTT in the worst case. It is especially for smaller files that the slow start effect of high latencies becomes clearly noticeable. A long establishment time, and a long time before the window reaches its optimum reduces the performance and overall throughput more for a small file than a large file. A connection with higher potential bandwidth will also waste more resources than a connection with low bandwidth. To improve RTT performance, that is, to reduce the delay, is without doubt a factor that will improve the overall performance.

2.10.2.4 UDP performance in wireless networks

The user datagram protocol (UDP) is unlike TCP an unreliable protocol, that is, it guarantees no delivery of data. When a host wants to send UDP datagrams, it simply sends it. No connection establishment is needed and the sender has no idea if the receiver receives the data. Thus it introduces not as many issues as TCP since there are no ACKs, retransmissions, and flow or congestion control. For real time applications, which are what UDP are mostly used in, medium to small size datagrams are commonly used. This implies that only a small datagram is lost in case of datagram transfer error, but it also introduces more overhead (due to datagram headers) than what would be the case with larger datagrams. Header compression is used to deal with this, and finding the correct datagram size is of importance.

2.10.2.5 Application layer

There are also degradations at the application layer. For applications that need session establishment, like VoIP with SIP and streaming with RTSP the network delay plays a significant factor in the time it takes to establish the session. Some applications, like streaming needs several PDP contexts, one for RTSP signaling and one for RTP data. This means higher setup delay, on the other side it makes it possible to give better QoS for the different flows. Application protocol tweaking and tuning is also something that should be focused on, e.g. HTTP 1.1 delivers better performance than HTTP 1.0.

3 Measurements and results

The concept of end-to-end testing is to perform tests and measurements that can tell us about the performance as experienced from an end user. The testing is done by performing tasks that are possible to do without any “inside” information or equipment installed in the network. An end-to-end test is typically performed with a computer connected to GPRS via a mobile phone. What measurements to be done to effectively measure the end-to-end performance are the subject of this thesis, but they should be measurements that indicate the user experience. Throughput and different types of delay are typically such indicators. First in this chapter we will tell the modifications of Telecas E2E Agent that were necessary to get it to work with the N70. Secondly we will describe the various measurements we performed and their results. First measurement is a benchmark between UMTS and EDGE in Telenor’s network to see if there are any significant differences in performance. Secondly we benchmark the dialup delays and RTTs in Netcom and Telenor’s 3G Network to see if there are any great differences. Latency is as we have noticed earlier and important factor especially for TCP and HTTP. In 3.5 we wanted to see the effects of different TCP window sizes in cellular networks. We wanted to see the effects of the establishment of the radio access bearers (RAB) and performed a jitter and packet loss measurement in two different scenarios, one with RAB established and one without RAB established, for both down and upload. Jitter is a KPI that introduces the buffering of streaming media. It became interesting to compare the uplink and downlink measurements with focus on packet loss and buffering.

3.1 TWSE2E modifications

The TWSE2E tool we used to perform the measurements needed some modifications to support the Nokia N70 phone, as the tool had originally been developed for, and certified for, some specific Sony Ericsson phones. Eclipse was used as the development environment, and we will document our changes here. As we quickly found out, the TWSE2E tool did not support the phone out of the box. Our first task was to debug and find out which modifications that were required. We found the problem to be unsupported AT commands on the N70. We then had to replace or remove those AT-commands that the N70 did not support, and that was used in the application. The TWSE2E uses only standardized commands, and no proprietary Sony Ericsson commands, but the standard [19] specifies that many commands are optional. Commands not supported at all were:

AT+cgqreq, to request QoS class. This command is optional to implement according to standard [19]. If the command is not implemented, all the values are considered to be unspecified.
AT+cgpaddr, to show pdpaddress (ip-address). Implementation is optional according to standard.
There is also a command supported, but with an unsupported parameter:
AT+cgdcont. Enabling data compression is unsupported.

Removing these commands and working around the problems was the changes that needed to be done. A class for the N70 was created that identified and verified the phone, and the application was ready for action.

```
Starting 'E2E Agent' - property of Teleca Wireless Solutions
Using COM5
NN70 interface starting up
12:48:04;Extracting Mobile Terminal (MT) information...OK...Nokia
N70 detected.
12:48:21;Test scenario description (Enter to continue)
12:48:21;-->:
12:48:23;Test description:
12:48:23;Defining PDP-context...OK
12:48:23;->Apn = internet, Data-compression = OFF
12:48:23;Setting QoS with reliability class = 2(LLC_ACK)...ERROR
12:48:23;Failed setting QoS. Command probably not supported on
phone. Continuing anyway.
12:48:23;Switching on V.42bis compression...OK
```

Figure 14 TWSE2E console output with the Nokia N70.

That the AT-cgqreq command is unsupported means that we have no option to request LLC acknowledged or unacknowledged mode, and have to be happy with what the network assigns to us. A problem with the AT-cgqreq in general is that although one is requesting a certain QoS, one cannot verify what QoS one is actually given by the network.

3.2 Overview

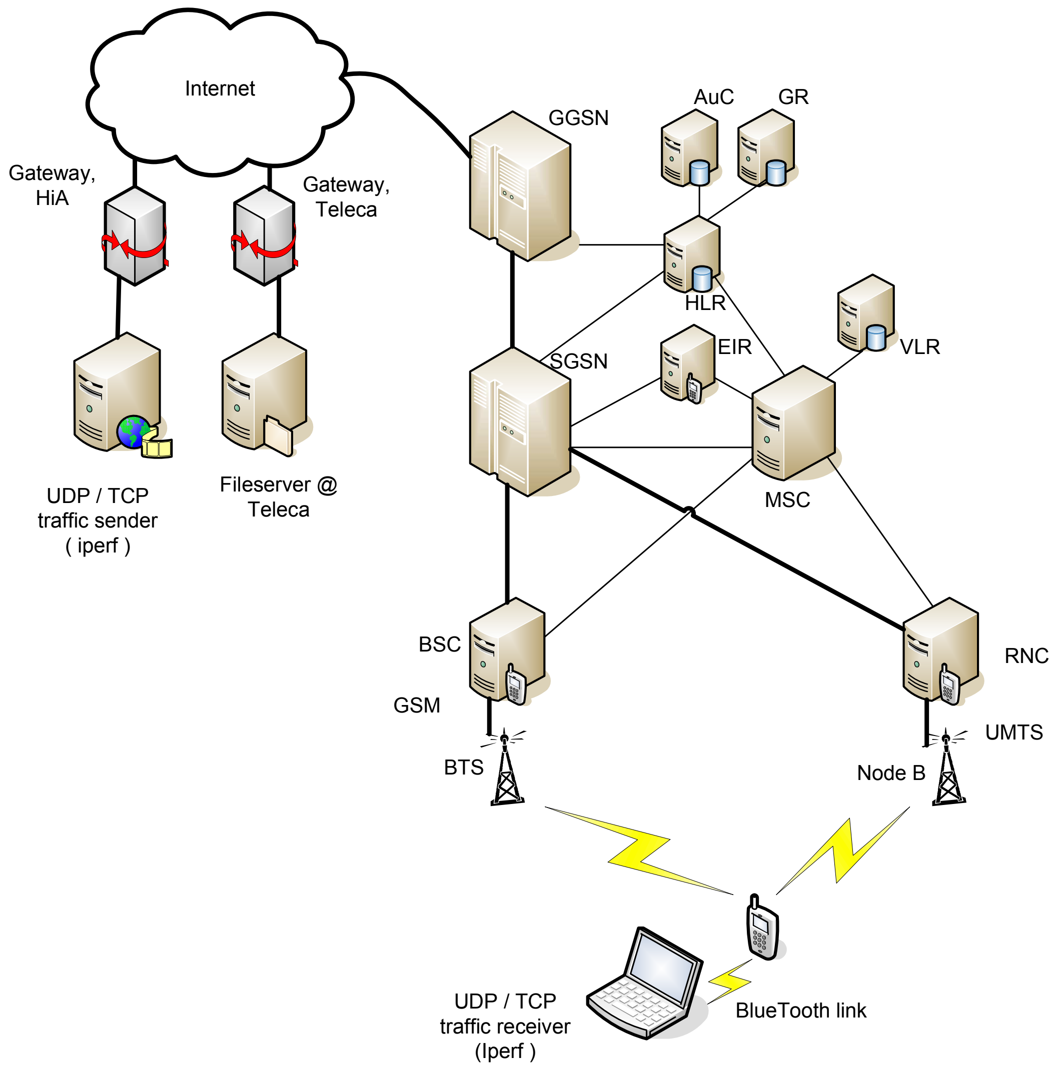


Figure 15 Overview of the test architecture

3.3 Comparison between Telenor EDGE and 3G/UMTS

3.3.1 Introduction

We performed an end-to-end test with Telecas TWSE2E tool to see the application in work. The test was done as a comparison between EDGE and 3G, i.e. UMTS, in Telenor's network. The goal was to obtain relevant data which can be used to analyze the characteristics of those respective technologies and point out differences between them, with weight on performance as seen from the end user. One of the objectives of this measurement was to see if there were any significant differences in the performance, i.e. throughput, in services like FTP and HTTP. It was interesting to see if our measurements are like the results of Chakravorty [37]. Measurements that will be done are throughput and latency for FTP and HTTP.

3.3.2 Test Setup

The test was performed using a Nokia N70 mobile phone as a modem. It was connected to a laptop running Windows XP. The phone is GSM, GPRS, EDGE and 3G capable. The test consists of a loop which performs these actions:

1. Signaling, Attach → Activate → Deactivate → Detach.
2. HTTP downloads of webpage.
3. Wap download[2].
4. FTP download of zip and text file.
5. FTP upload of zip and text file.
6. ICMP, PING tests for Round Trip Time measurements. Performed with ICMP payload of 12, 200, 468 and 1472 bytes.

All up-and downloading and pinging were done against Telecas own test server. The reason for ftp down-and uploading of both a zip and text file is to check if any data compression is used. Compressing an already compressed zip-file will not make any difference, while a plain text file is highly compressible. The loop was executed fifteen times. Fifteen times should be enough to ensure statistical relevance. The Performance application in Windows XP was used to log bytes received and sent per second, total traffic per second and TCP segments retransmitted per second. The Performance application logs the numbers to a file which can be imported into Microsoft Excel, or another spreadsheet. The phone can be forces to operate in GSM mode, but there is no option to force EDGE off or on. Therefore we had to look at the results to conclude if EDGE or plain GPRS was used (One can determine this by i.e. the throughput achieved). When the phone is connected to a 3G network, an icon is displayed on the phone's display.

Measurement information:

- 3G test started: 13:34 06.03.2006
- EDGE test started: 14:17 06.03.2006
- Network used: Telenor GSM and UMTS
- File size ftp-download-zip: 199 KB (203 917 bytes)
- File size ftp-download-txt: 199 KB (203 806 bytes)
- File size ftp-upload-zip: 49,7 KB (50 994 bytes)
- File size ftp-upload-text: 49 KB (50268 bytes)
- HTTP: Download: 15 files, total download of 172,5 KB (176592 bytes) with HTTP 1.1 including persistent connection, but without parallel connections.
- File size WAP: 4,7 KB (4852 bytes)

3.3.3 Test results 3G

The test was started at 13:34 06.03.2006. Received signal strength indicator (RSSI) and channel bit error rate (BER) was extracted from the MS after each loop. RSSI was between 8 and 11, which is not optimal. BER was received as 99, not known or not

[2] This is not really WAP. The file downloaded is somewhat like a WAP-file, but transport protocol is TCP and download method as in the HTTP test. WAP v1.x uses UDP as its transport protocol. WAP v2.x is more like ordinary HTTP with TCP. However, the measurement indicates the performance of downloading a single small file, and therefore has some relevance.

detectable for each loop. LAC (Location Area) was E1C4 and CI (Cell ID) switched between AF63 and B42A.

3.3.3.1 Signaling

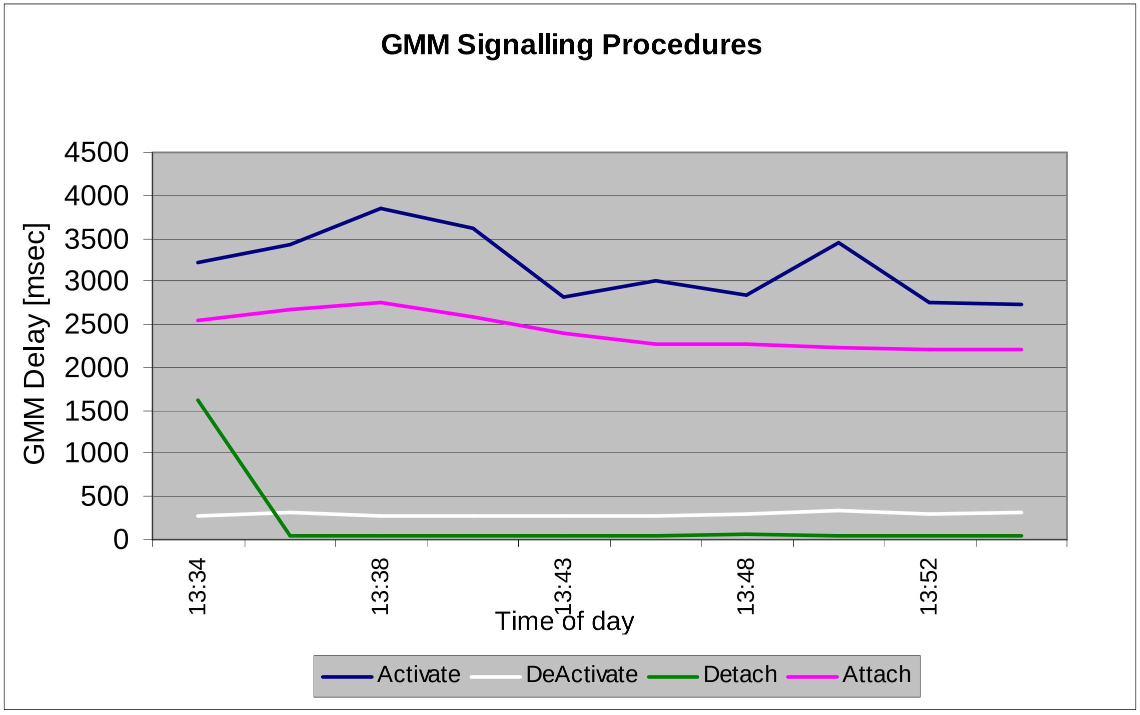


Figure 16 Signaling delay in Telenor 3G. Time of day on x-axis.

The various signaling delays are relatively stable.

GMM delay [msec]	***Average***	***Minimum***
Attach	2492	2163
Detach	150	40
Activate	3104	2674
Deactivate	302	270

Table 2 Average and minimum signaling delay

3.3.3.2 ICMP

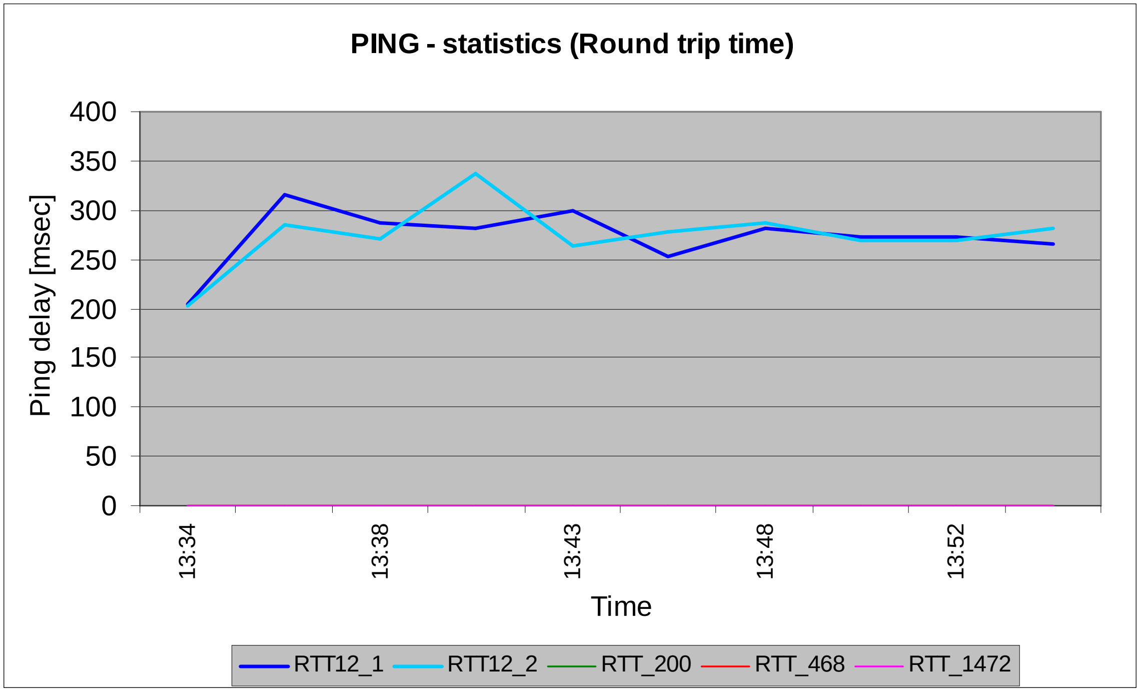


Figure 17 Round Trip Time in Telenor 3G.

Ping with ICMP payload of 12 bytes was successfully echoed over the internet. Packets with ICMP payload of 200, 468 and 1472 bytes was sent, but not echoed back. They were stopped by a firewall. It is interesting to notice that the RTT at the first ping (RTT12_1), when the radio access bearer (RAB) should not have been established is generally not any higher than the second ping (RTT12_2) which was sent when the RAB should have been established. The average delay for RTT12_1 is 280 ms and for RTT12_2 276 ms, which must be considered to be equal.

3.3.3.3 Web/FTP/WAP[3]

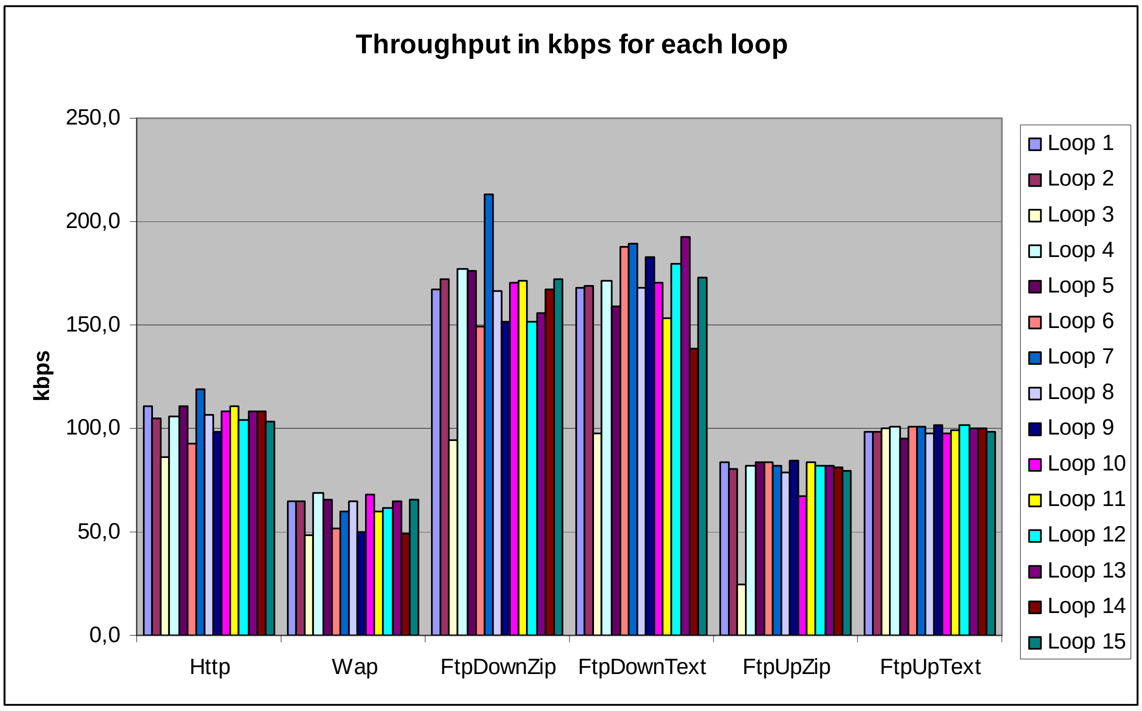


Figure 18 Throughput in kilobit per second for FTP, WAP and HTTP for each loop in Telenor 3G.

Figure 4 shows the throughput for each test in each loop. We can see that the throughput varies slightly, especially the downloads. Max download speed achieved for an entire file was 213,2 kbps. For the downloads there is nothing that indicates that any compression is used (the text file is highly compressible, while the zip file is not). The upload, however, goes a little faster with the text file. This could be only a coincidence, but as it is higher in each of the 15 loops, and the average throughput is almost 20 kbps higher, it could be that there is a technical reason for the difference. Especially the upload-speed is very stable, and varies just slightly. Something has happened in loop three that affected all, but the FTP upload text test. Unfortunately Ethereal was not run during this test, so it is hard to analyze what happened, but the Performance counter reports some retransmitted TCP segments during this loop. Applications we are not aware of, like Windows Update could also have accessed the internet at this time. This shows us the importance of running Ethereal or another packet analyzer while performing the tests.

Throughput [Kbps]	***Average***	***Limit ****
HTTP_download	104,5	n/a
WAP_download	59,6	n/a
FtpDownload.zip	159,3	213,3
FtpDownload.txt	162,2	192,5
FtpUpload.zip	70,1	84,2
FtpUpload.txt	99,3	101,5

Table 3 Average throughput in kbps

[3] This is not really WAP. Se footnote 1.

Figure 5 displays the average throughput in kbps. In the Limit columns, the first three seconds used to establish a FTP session is removed. The number in these columns is more like the number an ftp client would report as the download speed. An interesting point is that according to Telenor [39], they do not offer higher uplink bit rate than 64 kbps in UMTS. Our numbers exceeds this.

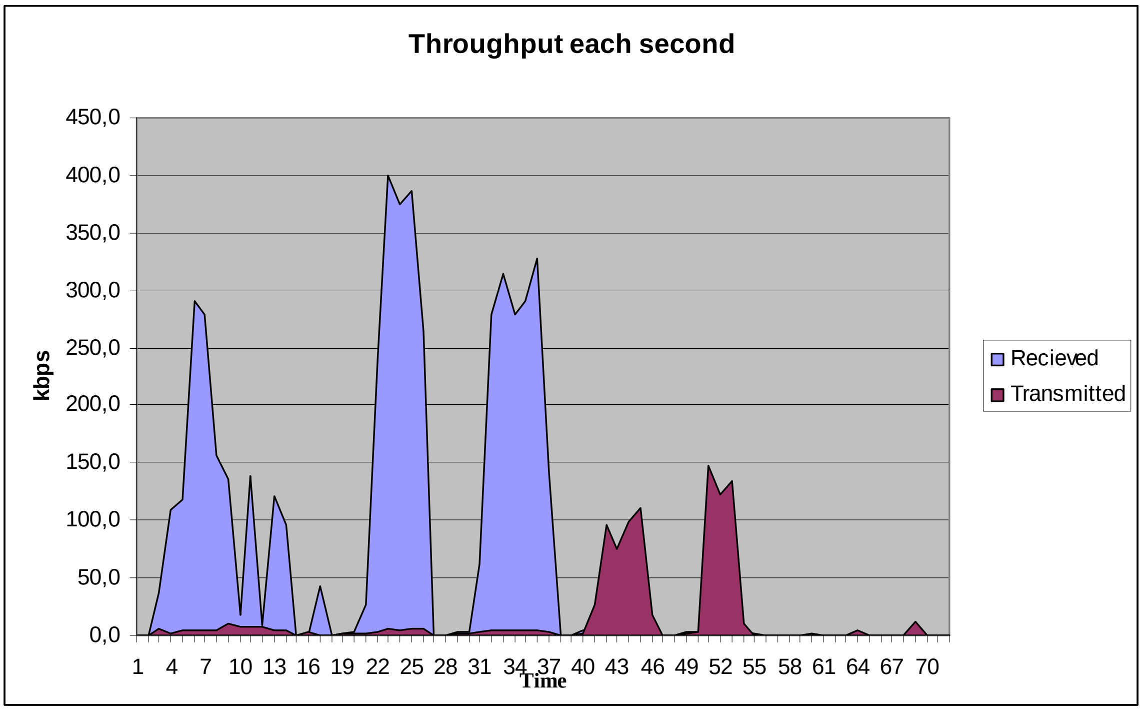


Figure 19 Brutto throughput each second in repetition 7 in Telenor 3G.

Figure 19 is a graph created with Microsoft Excel. It uses data collected with the Performance Counter in Windows. According to documentation the received values are "*Bytes Received/sec is the rate at which bytes are received over each network adapter, including framing characters*", and for transmitted:" *Bytes Sent/sec is the rate at which bytes are sent over each network adapter, including framing characters.*" The values are converted to bits before plotting. It plots the new value every second, and it illustrates the data traffic as it was in loop 7. The blue area is received data, and the purple is transmitted data. We can see the received HTTP and WAP data up to about 17 seconds. The next received data is the ftp-zipfile, and then at around 31 seconds the ftp-text file. The two purple areas are first the transmitted zip file, then the transmitted text file. We can see the peak for download is at 400 kbps, so the much anticipated bit rate of 384 kbps should be in range at least for the peaks. The reason the speed here is even higher than what should be maximum, is most probably inaccurate measurements, e.g. timing error in the sample interval.

3.3.3.4 Notes

Either during or after loop 7, the phone switched Cell ID from AF63 to B42A. During or after loop 13 it switched back to AF63. During or after loop 14 it switched once again to B42A. No noticeable differences in the results have been identified as a result of the Cell ID switching.

3.3.4 Test results GPRS/EDGE

The test was started at 14:17 06.03.2006. There is no option to force EDGE off and on, but the test results clearly show that EDGE is used, as the achieved bitrates supersedes what ordinary GPRS can manage. Received Signal Strength Indicator (RSSI) and channel bit error rate (BER) was extracted from the phone after each loop. The RSSI was 17, which is not optimal. BER was received as 99, not detectable or not known. LAC was BE34 and CI 23D5.

3.3.4.1 Signaling

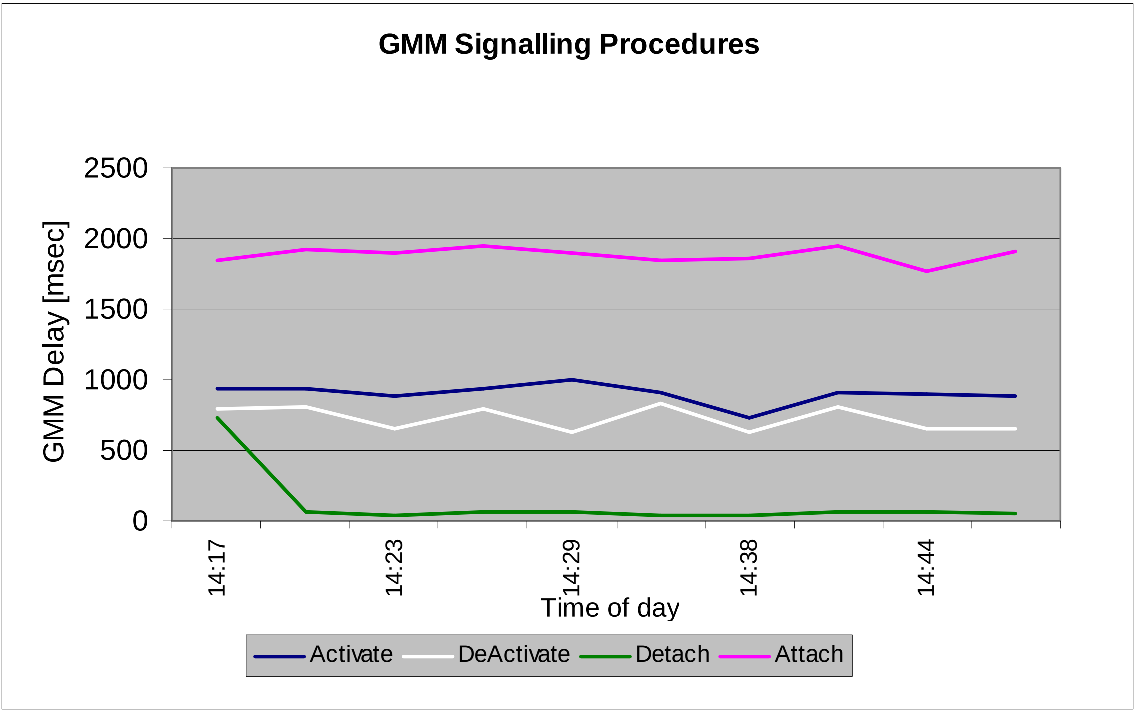


Figure 20 Signaling delay in Telenor GPRS/EDGE.

The graph shows us that the delays are pretty stable each time the various actions are performed.

GMM delay [msec]	***Average***	***Minimum***
Attach	1884	1712
Detach	97	40
Activate	949	731
Deactivate	716	630

Table 4 Average signaling delays.

3.3.4.2 ICMP

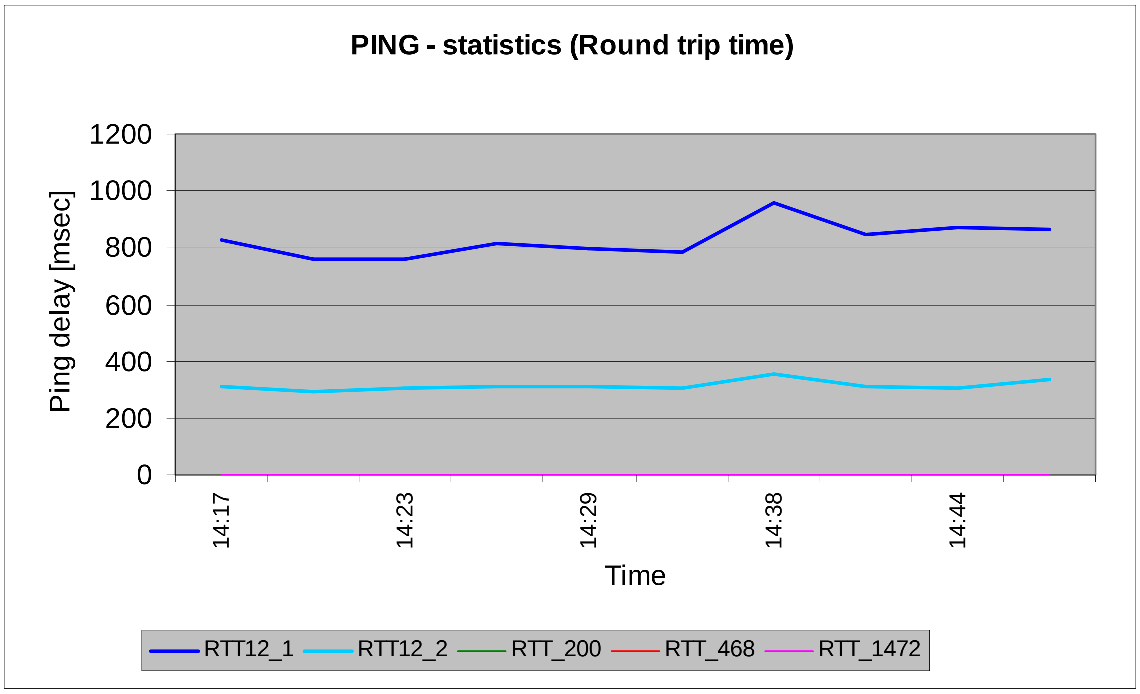


Figure 21 Round trip time in Telenor EDGE.

Ping with ICMP payload of 12 bytes was successfully echoed over the internet. As in 3G, packets with ICMP payload of 200, 468 and 1472 bytes were not possible to perform. Here we can clearly see that the response time of the first ping with 12 bytes of ICMP payload (RTT12_1) is clearly higher than the second one. This has to do with the radio interface and the need to set up a TBF (Temporary block flow) for the first packet. When the second ping is executed (RTT12_2) the already established TBF is reused and the RTT is therefore significantly lower.

3.3.4.3 Web/FTP/WAP[4]

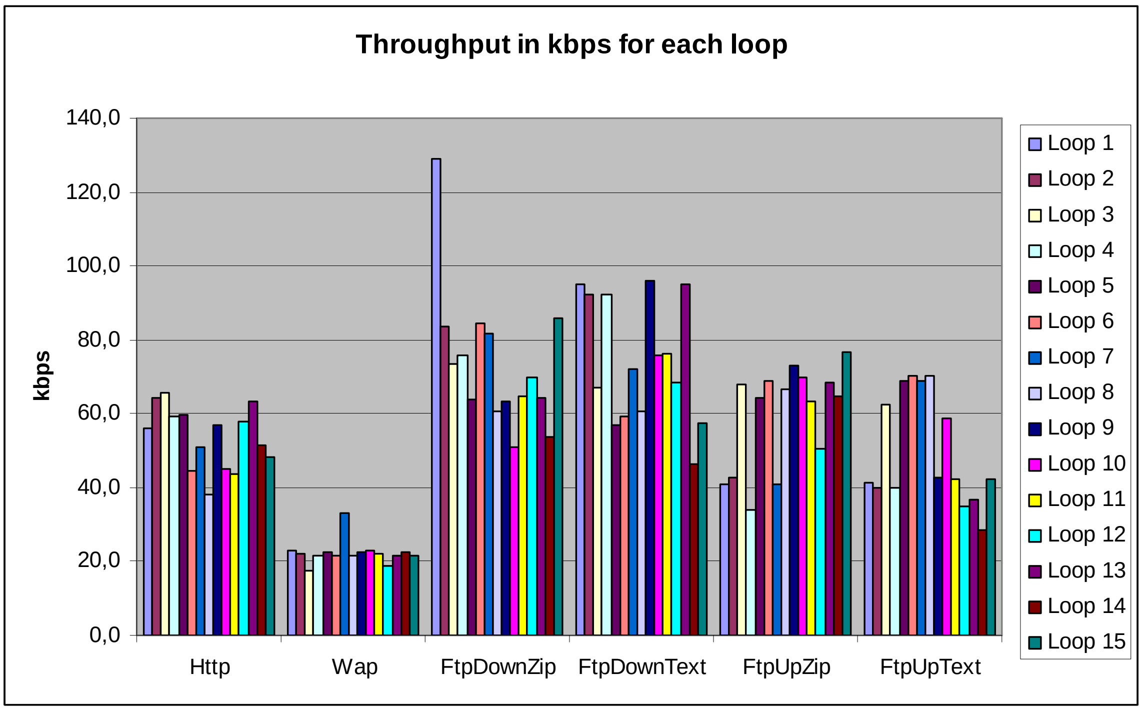


Figure 22 FTP, WAP and HTTP throughput in Telenor EDGE.

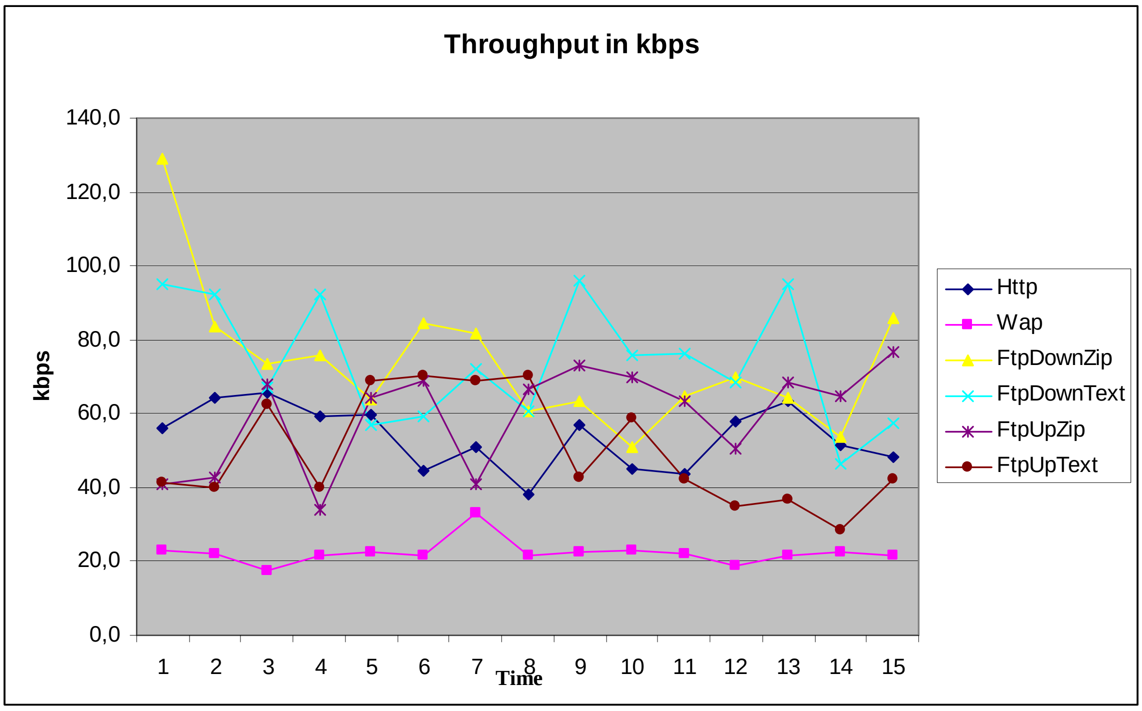


Figure 23 FTP, WAP and HTTP throughput in Telenor EDGE.

As with the 3G test, there is no indication that the network performs any compression, as it seems to be a coincident which file is transferred fastest. The average throughput

[4] As in 3G, this is not really WAP. See footnote 1.

table in Figure 12 shows us that average throughput for the FTP downloads is essentially the same. For the uploads, however, the ZIP file has an advantage for unknown reasons. According to Telenor [39], downlink bit rate should be in the area 100 – 200 kbps and uplink bit rates 50 – 75 kbps. It is only, as seen in Figure 13, in the peaks for a shorter period of time bit rates of 100 – 200 kbps are reached.

Throughput [Kbps]	***Average***	***Limit ****
HTTP_download	52,3	n/a
WAP_download	21,9	n/a
FtpDownload.zip	70,2	129,1
FtpDownload.txt	70,4	96,0
FtpUpload.zip	56,0	76,9
FtpUpload.txt	45,9	70,5

Table 5 Average throughputs.

In the Limit columns, the first three seconds used to establish a FTP session is removed. The number in this column is more like the number an ftp client would report as the download speed.

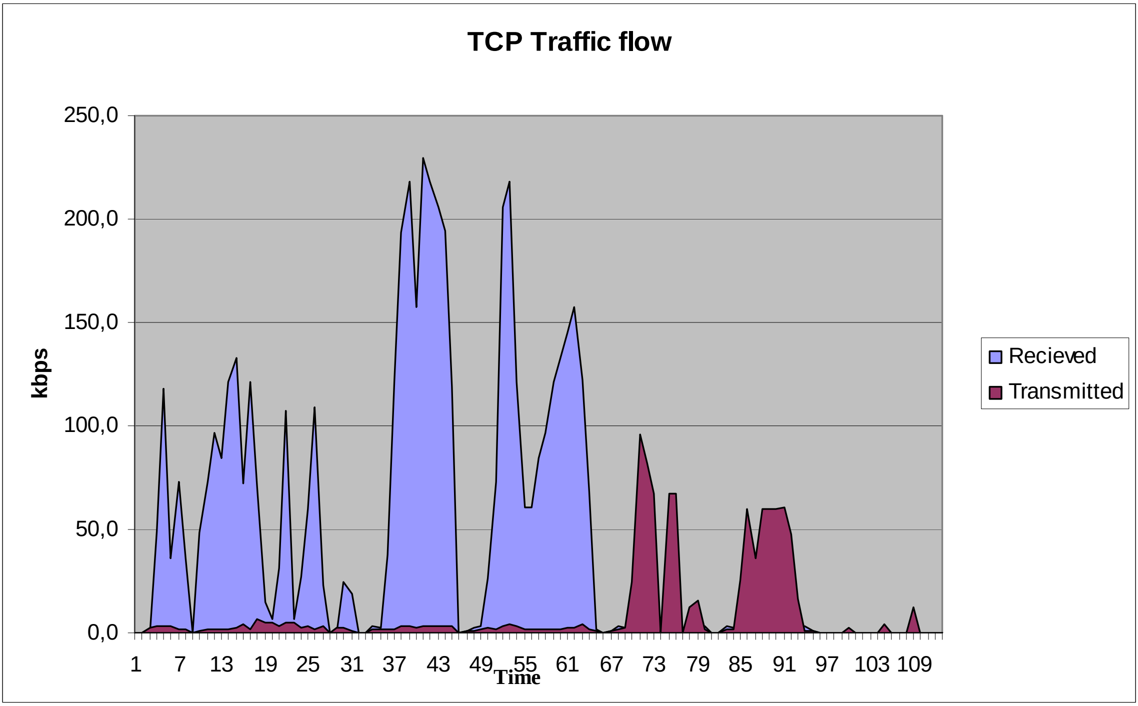


Figure 24 Brutto throughput each second in Telenor EDGE measured with Windows performance counter.

Figure 13 plots every second so we can se the actual throughput at each second and find the max peak values. Blue areas are received data, and purple areas are transmitted data. The highest peak in the first loop is the ftp zip file download. This peak is at 229 kbps. Max throughput when four timeslots are used should be 236,8 kbps in EDGE. Max upload peak occurs when the zipfile is transmitted and is 96 kbps. To achieve these speeds, one can assume that 4 slots are used for down and 2 for uplink (Switching between 4+1 and 3+2) with coding scheme MSC9.

3.3.5 V.42bis compression

The two tests were performed 07.03.2006 at 13:44 and 14:37.

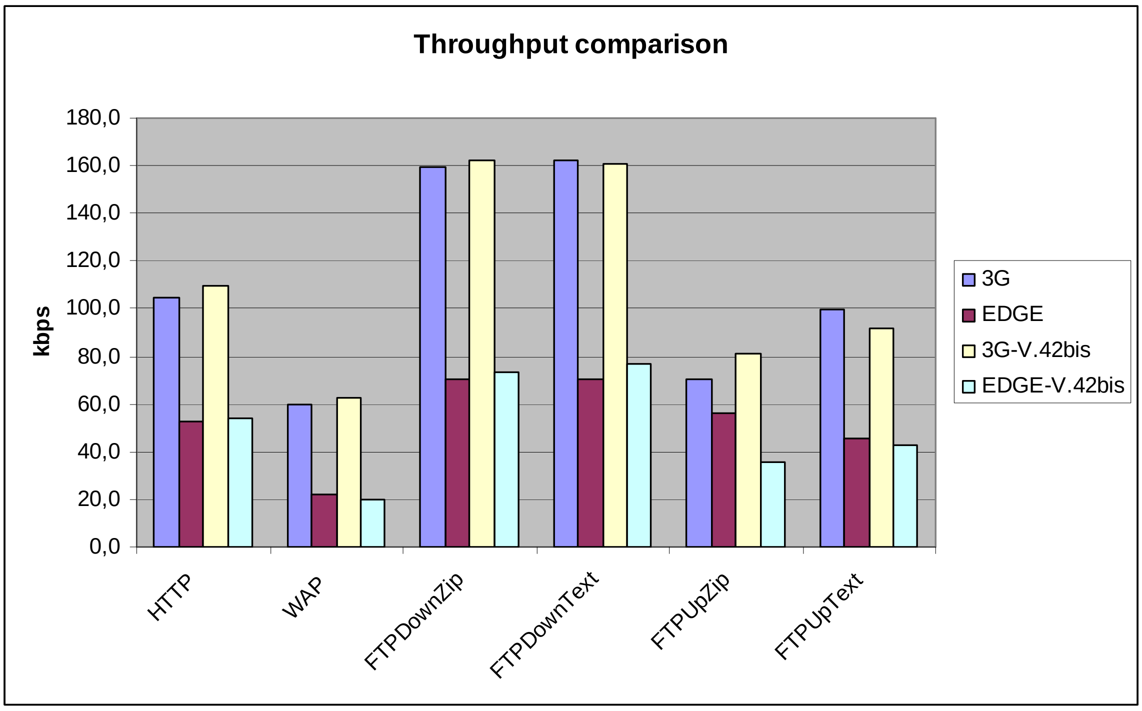


Figure 17 Throughput comparison with V.42bis compression enabled and not.

Enabling V.42bis compression had no effect, either for 3G or EDGE. No further comments on the two tests with compression enabled, as they did not show any significant differences from the tests with compression disabled.

3.4 Dialup and Round Trip Time

This measurement was a benchmark test of the Round Trip Delay and dialup delay in Netcom and Telenor using the 3G network. The tests were performed with the Nokia N70 3G capable mobile phone connected to a laptop. The application used for the measurements was Teleca's TWSE2E End Agent. The application is capable of measuring throughput and GMM signaling, but only dialup and Round Trip Time (RTT) was performed in this test. As mentioned earlier, the RTT is a parameter that greatly affects the upper layers, therefore we saw it as interesting to se if there were any differences in the performance of the two operators network regarding RTT and access delay. A total of 14 repetitions were performed, in the order: **Dialup → Ping 1 → Ping 2 → Hang-up.** ICMP payload in the echo request packages in the ping was 12 bytes. The reason for two consecutive ping tests was to see if there is a difference between the RTT when the network has been idle for a period and not. When the network has been idle, the radio access bearers should have been released. The goal of this measurement was to find the establishment delays, of both the dialup connection, and the radio bearers. In our previous test, we discovered that 4,5 seconds of idle time was not enough to take down the radio bearers in Telenor 3G. We wanted to see if we could find how long this idle time should be, to ensure RAB is taken down. The establishment time of the radio bearers are identified with a single ping after the network has been idle for a period of time. The host that was pinged was www.teleca.no

3.4.1 Telenor

Date of test: *03.21.2006*
Time of day: *13:15*
Location: *Teleca Grimstad.*
Repetitions*: 14*
Pause before first ping: *11,5 secs.*

13:15 should be a time the network is fairly busy. The reason for using 14 repetitions was to get a high enough number for some statistical relevance. 11,5 seconds of idle time was used as we discovered with earlier measurements that this was enough to ensure that the radio access bearers was completely taken down.

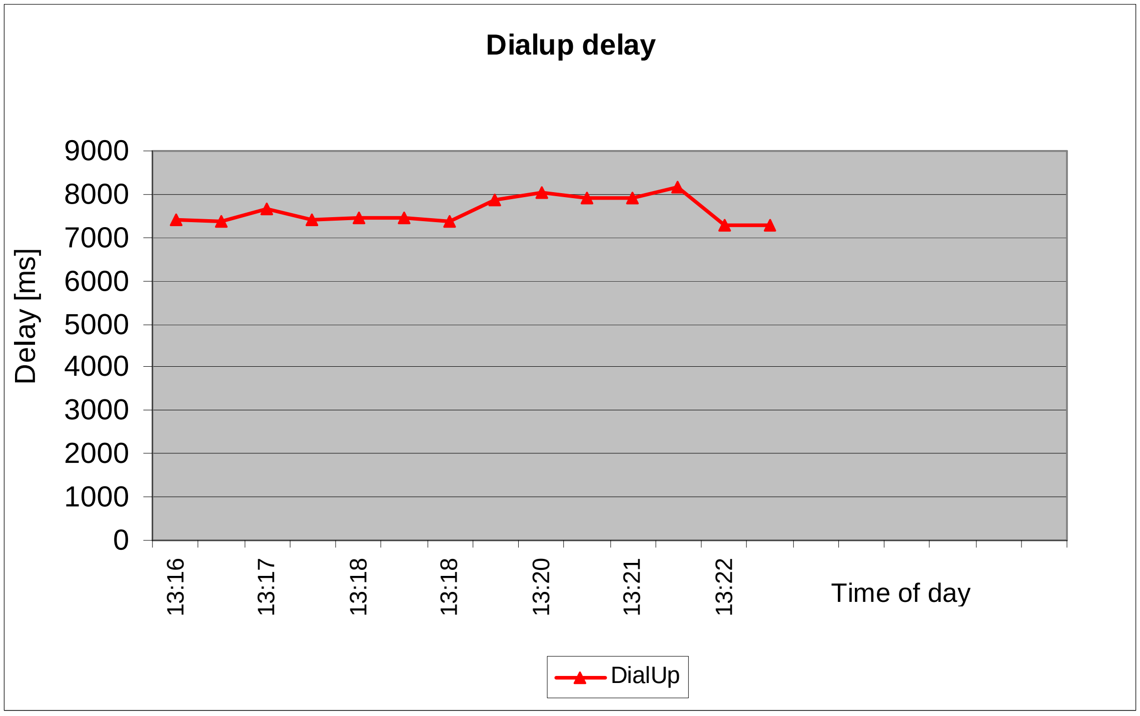


Figure 25 Telenor dialup access time.

From the graph, we can see the dialup access time is stable around 7 or 8 seconds.

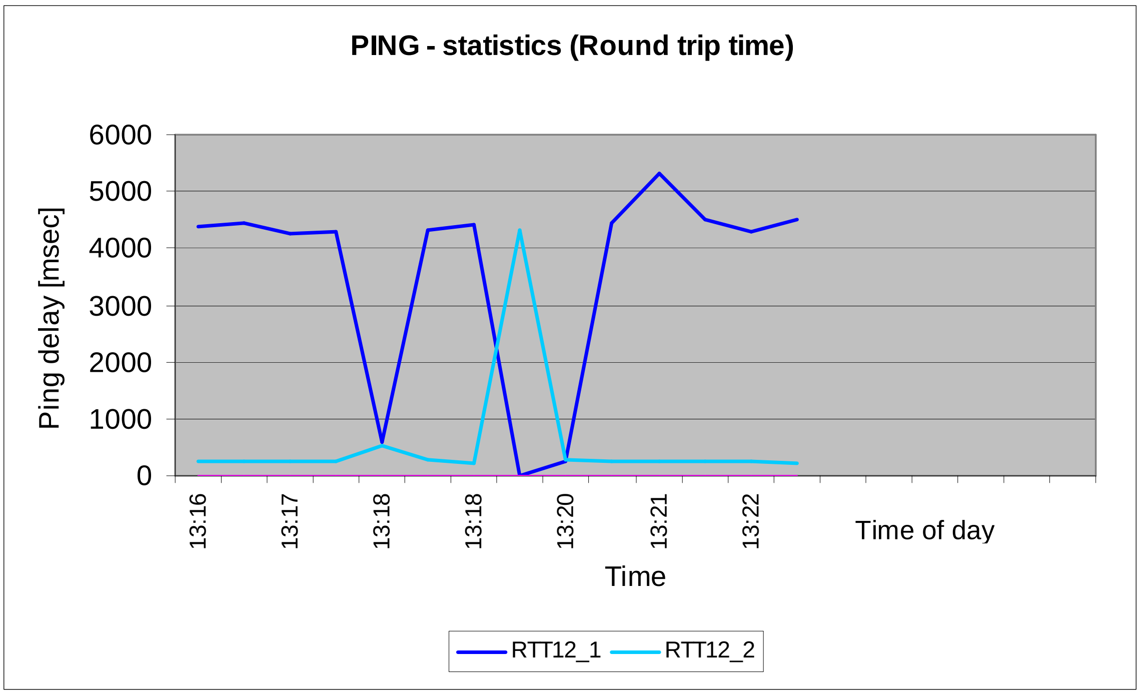


Figure 26 Telenor round Trip Time in 3G

As we can se from the graph, the first RTT is significantly higher than the second. The RTT of the first ping is about 4-5 secs, with three exceptions, two of them with low RTT and one time out. After the timeout after about 13:19, the RTT of the consecutive ping is about 4,5 secs. Before the first ping (RTT12_1) the network is idle of 11,5 seconds. The high RTT of the first ping indicates that radio resources are taken down and has to be set up again and that this takes what must be considered pretty long time. The reason for the two RTTs with low delay, may be another application accessing the internet, or incoming trash-traffic from the internet. Unfortunately, we have no Ethereal log for any post-processing from this test.

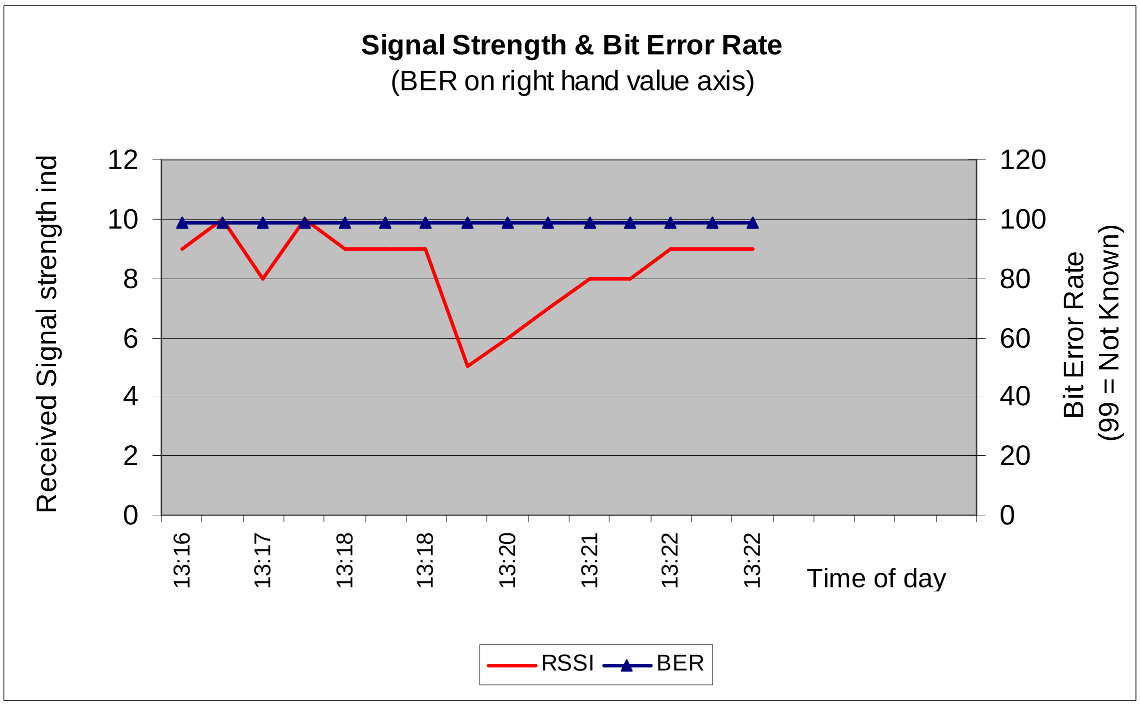


Figure 27 Telenor signal strength in office at Teleca Grimstad.

Signal strength is not optimal, bit error rate is unknown. The signal strength could perhaps affect the long delay for ping test, i.e. that setting up the radio resources takes longer time with bad signal than with optimal signal.

3.4.2 Netcom

Date of test: *03.28.2006*
Time of day: *15:22*
Location: *HiA Grimstad.*
Repetitions: *14*
Pause before first ping: *14,5 secs.*

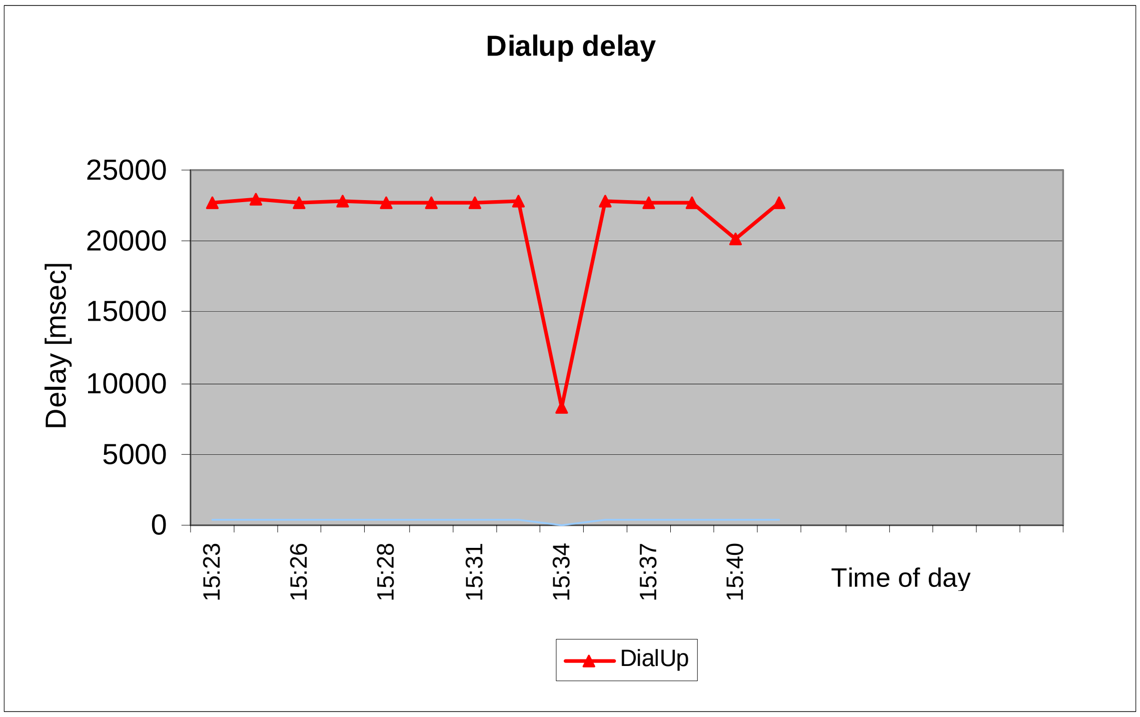


Figure 28 Netcom dialup delay in 3G.

Dialup delay is as high as about 22 seconds with one exception.

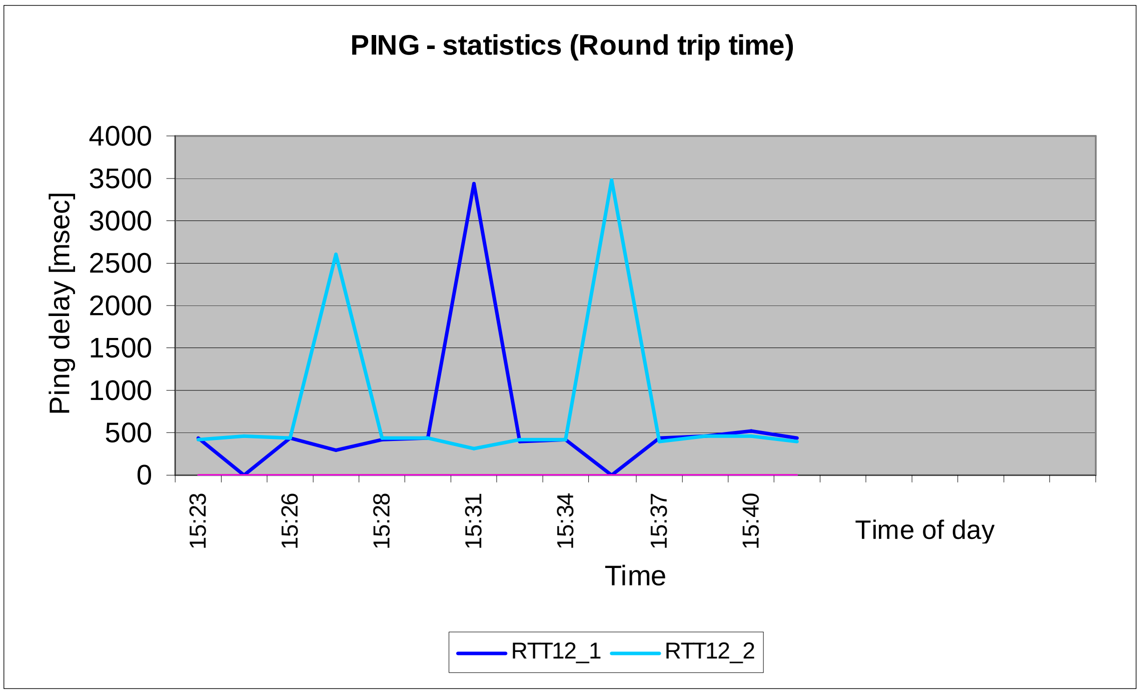


Figure 29 Netcom round trip time in 3G.

Even though there are some high RTT times, there is no indication that the first ping takes longer time. In these fourteen tests, surprisingly enough, ping no. 2 are most delayed in two cases, and ping no. 1 in only one, when there are significantly differences between the two. The idle network traffic before RTT12_1 was 14,5 seconds.

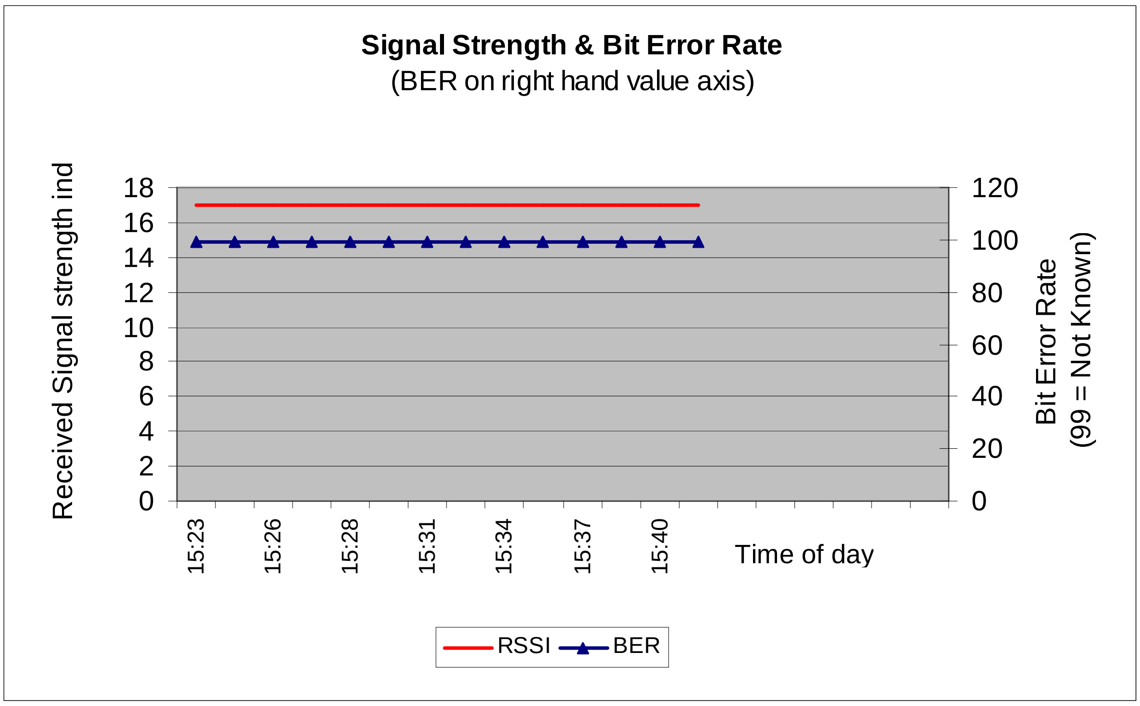


Figure 30 Netcom signal strength in at HiA Grimstad.

Signal strength is 17, bit error rate unknown.

3.5 TCP throughput downlink with different TCP window sizes

Test setup

Time of day: 12:00.

Network used: Telenor UMTS with public APN for the mobile phone, Uninett 100 mbit/s connection for the server end.

This small test using Iperf is to experiment with different TCP window sizes to look at eventual differences. TCP window size is the definition of how much data is in transit in the network at any time. For example if the value is too low, the data-exchanging parts will be idle for the time period while the data is in transit, causing overall performance to decrease noticeably.

```
Ping statistics for 128.39.202.234:
    Packets: Sent = 10, Received = 9, Lost = 1 (10% loss),
Approximate round trip times in milli-seconds:
    Minimum = 237ms, Maximum = 264ms, Average = 249ms
```

Round Trip Time from mobile to server is an average of 249ms. Bottleneck bandwidth in UMTS is 384 kilobits per second. From this we compute the bandwidth delay product:

```
BDP = 384000 bit/s * 0,249s = 95616 bits = 11952 bytes /1024 = 11,67 KByte/s
```

This gives us a starting point for the window size as 12KB. Default in Iperf is 8KB and we run the test with window sizes 8, 12, 25, 50 and 130KB.

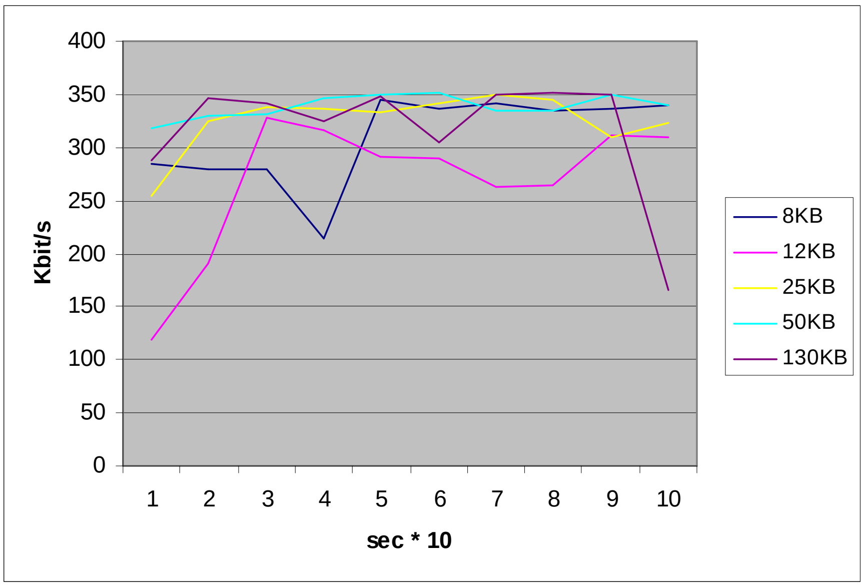


Figure 31 TCP throughput with various window sizes

Figure 31 is a graphical presentation of the results. We ran 10 tests with the different window size for 10 seconds each.. As the figure shows, there are slight differences between them. The middle values are usually around 300 – 350 kilobits per second, while there are sudden drops in speed with the 130KB and 12KB window size.

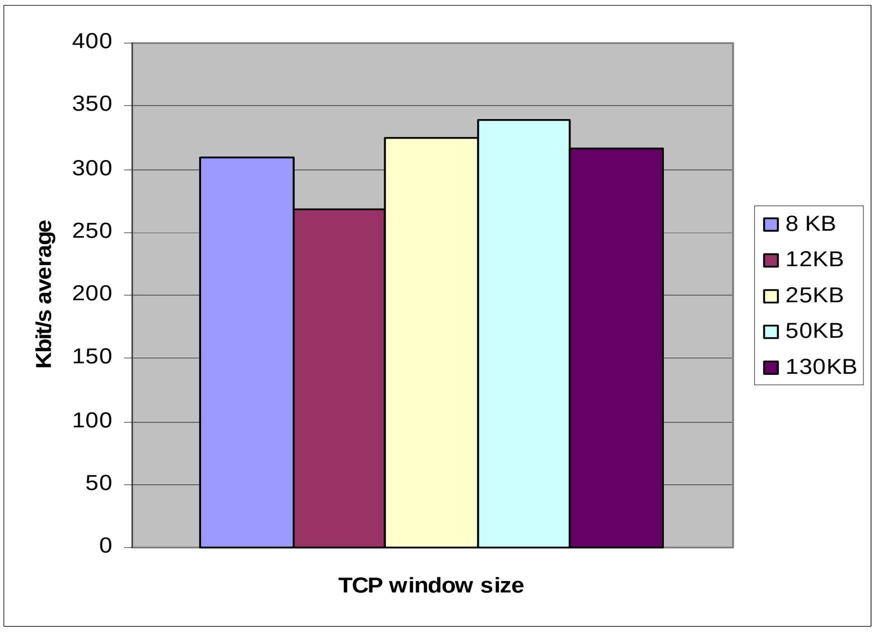


Figure 32 Average throughput with various window sizes

Figure 32 shows the average bitrate of the tests. Throughputs are around 300 kbps, which is the practical speed an end user could expect over time. There are some reasons why the test with 12KB is lower than the others: The tests were not performed in a lab- environment, rather a field-experiment, so the sudden drop in bitrate could be caused by higher network load at that specific moment due to congestion control reducing bitrate.

3.6 *Iperf testing - download*

We performed a measurement with Iperf,, the goal of the measurement was the following:

- To see how Iperf worked.
- To see what bit rates one could experience with UDP.
- How the radio access bearer establishment affects the performance.

3.6.1 Test setup

A laptop connected to the N70 using it as a modem. Iperf ran in server mode on a laptop and in client mode on a computer located at HiA. Iperf ran with the following parameters in server mode:

```
iperf.exe -s -u -i 1
```

And in client mode:

```
Iperf.exe –c <iptoserver> -u –b 360k
```

We were streaming UDP datagrams at the rate of 360 kilobits per second from the client to the server. To se how the radio bearer establishment in 3G is affecting the performance, we ran two trials. First, one where the network has been idle and RAB are released. Second, one where we first pinged the server from the client, and then immediately started streaming UDP datagrams to the server. This should ensure that the RAB was established. The network used was Telenor UMTS, which we have seen in the previous test had quite a large delay with the RAB establishment.

3.6.2 Scenario 1 – RAB not established

```
C:\Documents and Settings\Erling\Desktop\iperf>iperf.exe -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1932] local 212.17.139.28 port 5001 connected with 128.39.202.234 port 3149
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[1932]  0.0- 1.0 sec  14.4 KBytes   118 Kbits/sec  29.114 ms 1917869114/   10 (1.9e+010%)
[1932]  1.0- 2.0 sec  15.8 KBytes   129 Kbits/sec  271.243 ms  118/  129 (91%)
[1932]  2.0- 3.0 sec  15.8 KBytes   129 Kbits/sec  153.750 ms   22/   33 (67%)
[1932]  3.0- 4.0 sec  18.7 KBytes   153 Kbits/sec  120.059 ms   24/   37 (65%)
[1932]  4.0- 5.0 sec  53.1 KBytes   435 Kbits/sec  29.682 ms   32/   69 (46%)
[1932]  5.0- 6.0 sec  48.8 KBytes   400 Kbits/sec  18.400 ms    0/   34 (0%)
[1932]  6.0- 7.0 sec  43.1 KBytes   353 Kbits/sec  13.418 ms    0/   30 (0%)
[1932]  7.0- 8.0 sec  48.8 KBytes   400 Kbits/sec  13.156 ms    0/   34 (0%)
[1932]  8.0- 9.0 sec  38.8 KBytes   318 Kbits/sec  18.366 ms    0/   27 (0%)
[1932]  9.0-10.0 sec  50.2 KBytes   412 Kbits/sec  13.345 ms    0/   35 (0%)
[1932] 10.0-11.0 sec  38.8 KBytes   318 Kbits/sec  18.474 ms    0/   27 (0%)
[1932] 11.0-12.0 sec  47.4 KBytes   388 Kbits/sec  10.719 ms    0/   33 (0%)
[1932] 12.0-13.0 sec  45.9 KBytes   376 Kbits/sec  17.391 ms    0/   32 (0%)
[1932] 13.0-14.0 sec  34.5 KBytes   282 Kbits/sec  23.014 ms    0/   24 (0%)
[1932] 14.0-15.0 sec  45.9 KBytes   376 Kbits/sec  16.413 ms    0/   32 (0%)
[1932] 15.0-16.0 sec  40.2 KBytes   329 Kbits/sec  16.567 ms    0/   28 (0%)
[1932] 16.0-17.0 sec  53.1 KBytes   435 Kbits/sec  14.736 ms    0/   37 (0%)
[1932] 17.0-18.0 sec  44.5 KBytes   365 Kbits/sec  14.985 ms    0/   31 (0%)
[1932] 18.0-19.0 sec  44.5 KBytes   365 Kbits/sec  13.974 ms    0/   31 (0%)
[1932] 19.0-20.0 sec  44.5 KBytes   365 Kbits/sec  13.659 ms    0/   31 (0%)
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[1932] 20.0-21.0 sec  37.3 KBytes   306 Kbits/sec  23.858 ms    0/   26 (0%)
[1932] 21.0-22.0 sec  48.8 KBytes   400 Kbits/sec  14.847 ms    0/   34 (0%)
[1932] 22.0-23.0 sec  44.5 KBytes   365 Kbits/sec  14.724 ms    0/   31 (0%)
[1932] 23.0-24.0 sec  44.5 KBytes   365 Kbits/sec  15.157 ms    0/   31 (0%)
[1932] 24.0-25.0 sec  45.9 KBytes   376 Kbits/sec  12.297 ms    0/   32 (0%)
[1932]  0.0-25.7 sec  1.01 MBytes   331 Kbits/sec  15.319 ms  196/  920 (21%)
read failed: Connection reset by peer
read failed: Connection reset by peer
recvfrom failed: Connection reset by peer
```

As we can see, datagrams of 1470 bytes were used, this makes one datagram fit into one IP-packet, and datagram loss is the same as packet loss. The client streamed UDP datagrams at the rate of 360 kilobits per second for 30 seconds. The first we notice is that when the server first started receiving, many packets were lost. A lot of packets got lost probably due to the RAB establishment, and we can clearly see that it took some time because the server only received datagrams for about 25,7 seconds. To directly calculate the RAB establishment time to be 30-25,7 = 4,3 seconds would probably not be hundred percent correct, as the network may have buffered some packets and make such a calculation faulty.

3.6.3 Scenario 2 – RAB established

```
C:\Documents and Settings\Erling\Desktop\iperf>iperf.exe -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1932] local 212.17.139.28 port 5001 connected with 128.39.202.234 port 3150
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[1932]  0.0- 1.0 sec  14.4 KBytes   118 Kbits/sec  30.660 ms 1917869114/   10 (1.9e+010%)
[1932]  1.0- 2.0 sec  30.1 KBytes   247 Kbits/sec  35.025 ms    9/   30 (30%)
[1932]  2.0- 3.0 sec  47.4 KBytes   388 Kbits/sec  24.307 ms   18/   51 (35%)
[1932]  3.0- 4.0 sec  45.9 KBytes   376 Kbits/sec  13.691 ms    0/   32 (0%)
[1932]  4.0- 5.0 sec  41.6 KBytes   341 Kbits/sec  18.060 ms    0/   29 (0%)
[1932]  5.0- 6.0 sec  43.1 KBytes   353 Kbits/sec  18.647 ms    0/   30 (0%)
[1932]  6.0- 7.0 sec  47.4 KBytes   388 Kbits/sec  18.169 ms    0/   33 (0%)
[1932]  7.0- 8.0 sec  40.2 KBytes   329 Kbits/sec  21.280 ms    0/   28 (0%)
[1932]  8.0- 9.0 sec  40.2 KBytes   329 Kbits/sec  13.265 ms    0/   28 (0%)
[1932]  9.0-10.0 sec  40.2 KBytes   329 Kbits/sec  17.420 ms    0/   28 (0%)
[1932] 10.0-11.0 sec  35.9 KBytes   294 Kbits/sec  31.333 ms    0/   25 (0%)
[1932] 11.0-12.0 sec  50.2 KBytes   412 Kbits/sec  22.137 ms    1/   36 (2.8%)
[1932] 12.0-13.0 sec  45.9 KBytes   376 Kbits/sec  18.768 ms    0/   32 (0%)
[1932] 13.0-14.0 sec  47.4 KBytes   388 Kbits/sec  19.923 ms    0/   33 (0%)
[1932] 14.0-15.0 sec  44.5 KBytes   365 Kbits/sec  18.059 ms    0/   31 (0%)
[1932] 15.0-16.0 sec  50.2 KBytes   412 Kbits/sec  15.919 ms    0/   35 (0%)
[1932] 16.0-17.0 sec  44.5 KBytes   365 Kbits/sec  14.115 ms    0/   31 (0%)
[1932] 17.0-18.0 sec  44.5 KBytes   365 Kbits/sec  12.245 ms    0/   31 (0%)
[1932] 18.0-19.0 sec  44.5 KBytes   365 Kbits/sec  12.132 ms    0/   31 (0%)
[1932] 19.0-20.0 sec  41.6 KBytes   341 Kbits/sec  15.191 ms    0/   29 (0%)
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[1932] 20.0-21.0 sec  44.5 KBytes   365 Kbits/sec  11.005 ms    0/   31 (0%)
[1932] 21.0-22.0 sec  44.5 KBytes   365 Kbits/sec  11.812 ms    0/   31 (0%)
[1932] 22.0-23.0 sec  44.5 KBytes   365 Kbits/sec  11.742 ms    0/   31 (0%)
[1932] 23.0-24.0 sec  43.1 KBytes   353 Kbits/sec  11.559 ms    0/   30 (0%)
[1932] 24.0-25.0 sec  44.5 KBytes   365 Kbits/sec  13.127 ms    0/   31 (0%)
[1932] 25.0-26.0 sec  44.5 KBytes   365 Kbits/sec  12.643 ms    0/   31 (0%)
[1932] 26.0-27.0 sec  44.5 KBytes   365 Kbits/sec  15.578 ms    0/   31 (0%)
[1932] 27.0-28.0 sec  41.6 KBytes   341 Kbits/sec  14.896 ms    0/   29 (0%)
[1932] 28.0-29.0 sec  41.6 KBytes   341 Kbits/sec  16.916 ms    1/   30 (3.3%)
[1932] 29.0-30.0 sec  41.6 KBytes   341 Kbits/sec  17.277 ms    0/   29 (0%)
[1932]  0.0-30.1 sec  1.25 MBytes   349 Kbits/sec  16.624 ms   29/  920 (3.2%)
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
[1932] WARNING: ack of last datagram failed after 10 tries.
recvfrom failed: Connection reset by peer
```

We can see here that although some packets are lost in the beginning, not as many as in the first scenario is lost. Total we have 3,2 % packet loss in this scenario compared to 21 % in the first scenario. Even though this measurement gave significantly better results, we still have a noticeable large amount of packet loss in the beginning of the receiving.

3.7 Iperf testing 2 – download 2

In the previous test, there was one very interesting measurement in both scenarios, although most notably in scenario 1. From the iperf output there is some strange numbers regarding packet loss from 0 – 1 second, and it seems like the majority of the packet loss was not in this interval, but from 1-2 seconds. To analyze this further, we performed a similar test but with Ethereal running for post-processing of the packet data. Our first thought is that there is some kind of packet buffering, but that this buffer may only hold a certain amount of data before it starts dropping packets.

In this measurement we increased the bit rate to 384 kilobits per second and streamed for 20 seconds. We increased the bit rate to see if we could make the characteristics even more visible. Except from this, the test setup was the same as in the previous test. Iperf was started with this in server mode:

```
iperf.exe -s -u -i 1
```

And the following in client mode:

```
Iperf.exe –c <iptoserver> -u –b 384k
```

3.7.1 Scenario 1 – RAB not established

```
C:\Documents and Settings\Erling\Desktop\iperf>iperf.exe -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1932] local 212.17.136.37 port 5001 connected with 128.39.202.234 port 3220
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[1932]  0.0- 1.0 sec  14.4 KBytes   118 Kbits/sec  28.463 ms 1917869114/   10 (1.9e+010%)
[1932]  1.0- 2.0 sec  14.4 KBytes   118 Kbits/sec  95.703 ms   27/   37 (73%)
[1932]  2.0- 3.0 sec  15.8 KBytes   129 Kbits/sec  170.324 ms  125/  136 (92%)
[1932]  3.0- 4.0 sec  25.8 KBytes   212 Kbits/sec  104.439 ms   38/   56 (68%)
[1932]  4.0- 5.0 sec  50.2 KBytes   412 Kbits/sec  32.425 ms   27/   62 (44%)
[1932]  5.0- 6.0 sec  47.4 KBytes   388 Kbits/sec  13.700 ms    0/   33 (0%)
[1932]  6.0- 7.0 sec  37.3 KBytes   306 Kbits/sec  21.657 ms    0/   26 (0%)
[1932]  7.0- 8.0 sec  53.1 KBytes   435 Kbits/sec  13.680 ms    1/   38 (2.6%)
[1932]  8.0- 9.0 sec  45.9 KBytes   376 Kbits/sec  14.143 ms    0/   32 (0%)
[1932]  9.0-10.0 sec  45.9 KBytes   376 Kbits/sec  13.604 ms    0/   32 (0%)
[1932] 10.0-11.0 sec  45.9 KBytes   376 Kbits/sec  12.868 ms    0/   32 (0%)
[1932] 11.0-12.0 sec  45.9 KBytes   376 Kbits/sec  13.046 ms    0/   32 (0%)
[1932] 12.0-13.0 sec  45.9 KBytes   376 Kbits/sec  11.578 ms    0/   32 (0%)
[1932] 13.0-14.0 sec  34.5 KBytes   282 Kbits/sec  25.606 ms    0/   24 (0%)
[1932] 14.0-15.0 sec  57.4 KBytes   470 Kbits/sec  15.474 ms    5/   45 (11%)
[1932]  0.0-15.8 sec   620 KBytes   321 Kbits/sec  13.955 ms  223/  655 (34%)
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
[1932] WARNING: ack of last datagram failed after 10 tries.
recvfrom failed: Connection reset by peer
```

As we can see, the characteristics of this measurement do not differ significantly from the previous experiment (chapter 3.6.2) regarding packet loss and jitter. The overall packet loss is higher here, but as the bit rate we streamed with here is at the edge of what UMTS can manage, this was expected.

3.7.2 Scenario 2 – RAB established

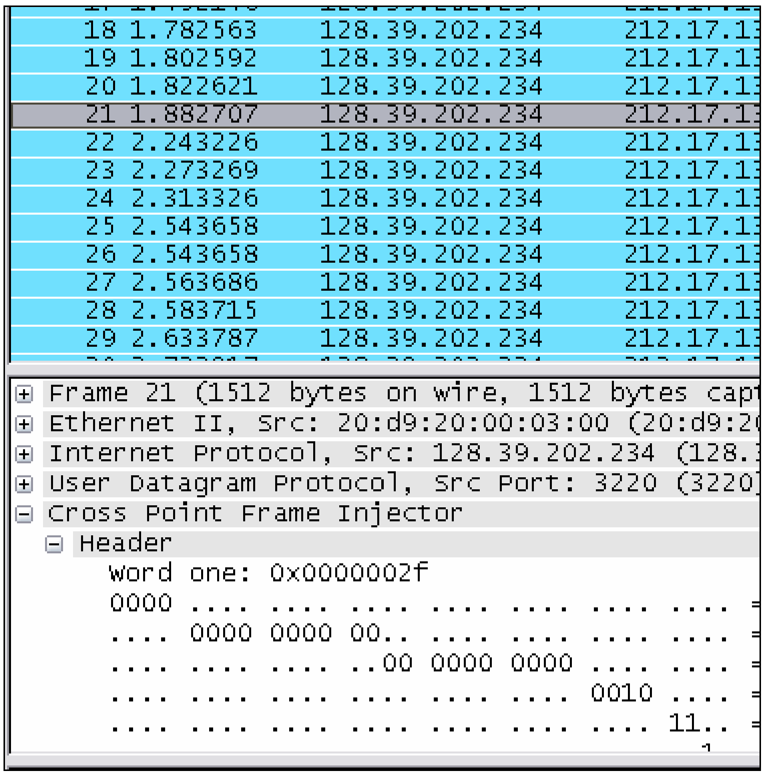
```
C:\Documents and Settings\Erling\Desktop\iperf>iperf.exe -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1932] local 212.17.136.37 port 5001 connected with 128.39.202.234 port 3221
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[1932]  0.0- 1.0 sec  14.4 KBytes   118 Kbits/sec  27.624 ms 1917869114/   10 (1.9e+010%)
[1932]  1.0- 2.0 sec  33.0 KBytes   270 Kbits/sec  43.727 ms   17/   40 (43%)
[1932]  2.0- 3.0 sec  43.1 KBytes   353 Kbits/sec  32.217 ms   14/   44 (32%)
[1932]  3.0- 4.0 sec  48.8 KBytes   400 Kbits/sec  12.666 ms    0/   34 (0%)
[1932]  4.0- 5.0 sec  41.6 KBytes   341 Kbits/sec  19.398 ms    1/   30 (3.3%)
[1932]  5.0- 6.0 sec  45.9 KBytes   376 Kbits/sec  18.614 ms    1/   33 (3%)
[1932]  6.0- 7.0 sec  50.2 KBytes   412 Kbits/sec  13.389 ms    0/   35 (0%)
[1932]  7.0- 8.0 sec  45.9 KBytes   376 Kbits/sec  14.091 ms    0/   32 (0%)
[1932]  8.0- 9.0 sec  44.5 KBytes   365 Kbits/sec  12.171 ms    0/   31 (0%)
[1932]  9.0-10.0 sec  45.9 KBytes   376 Kbits/sec  12.548 ms    0/   32 (0%)
[1932] 10.0-11.0 sec  47.4 KBytes   388 Kbits/sec  12.248 ms    0/   33 (0%)
[1932] 11.0-12.0 sec  37.3 KBytes   306 Kbits/sec  23.368 ms    2/   28 (7.1%)
[1932] 12.0-13.0 sec  50.2 KBytes   412 Kbits/sec  17.433 ms    1/   36 (2.8%)
[1932] 13.0-14.0 sec  40.2 KBytes   329 Kbits/sec  22.473 ms    0/   28 (0%)
[1932] 14.0-15.0 sec  54.6 KBytes   447 Kbits/sec  13.810 ms    1/   39 (2.6%)
[1932] 15.0-16.0 sec  45.9 KBytes   376 Kbits/sec  11.612 ms    0/   32 (0%)
[1932] 16.0-17.0 sec  37.3 KBytes   306 Kbits/sec  23.253 ms    0/   26 (0%)
[1932] 17.0-18.0 sec  45.9 KBytes   376 Kbits/sec  18.872 ms    3/   35 (8.6%)
[1932] 18.0-19.0 sec  43.1 KBytes   353 Kbits/sec  11.414 ms    2/   32 (6.3%)
[1932] 19.0-20.0 sec  56.0 KBytes   459 Kbits/sec  12.989 ms    1/   40 (2.5%)
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[1932]  0.0-20.4 sec   879 KBytes   354 Kbits/sec  24.320 ms   43/  655 (6.6%)
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
[1932] WARNING: ack of last datagram failed after 10 tries.
recvfrom failed: Connection reset by peer
```

Overall packet loss is 6,6 %, and much lower than in the case with RAB not established, were it was 34 %. Compared to the measurements in chapter 3,6 the characteristics of this measurement is similar to that in chapter 3.6.3 with RAB established.

3.7.3 Ethereal packet tracing

By analyzing the ethereal log of the scenario with the RAB not established we see that the first second, approximately twelve packets are received and none are lost. This is according to the Iperf output. The next second, we receive 7 datagrams up to 1,83 secs. After this we loose 26 datagrams before receiving the next datagram, numbered 2F at 1,88 seconds. The next datagram received is numbered 97 and is received at 2,24 seconds. This seems to coincide fairly well with the iperf output. This indicates that there is some kind of buffering in the network, since the first 19 packets are received. 19 packets multiplied with 1500 bytes are 28500 bytes.

In scenario 2 the first packet loss is after 24 packets. This and the performance Iperf reports regarding bit rate (118 kbps, or approximately 10 packets counted in Ethereal) and jitter in addition to packet loss strongly indicates that the first packets are received due to buffering. When the buffer is full, it starts to drop packets.



```
18 1.782563   128.39.202.234   212.17.1
19 1.802592   128.39.202.234   212.17.1
20 1.822621   128.39.202.234   212.17.1
21 1.882707   128.39.202.234   212.17.1
22 2.243226   128.39.202.234   212.17.1
23 2.273269   128.39.202.234   212.17.1
24 2.313326   128.39.202.234   212.17.1
25 2.543658   128.39.202.234   212.17.1
26 2.543658   128.39.202.234   212.17.1
27 2.563686   128.39.202.234   212.17.1
28 2.583715   128.39.202.234   212.17.1
29 2.633787   128.39.202.234   212.17.1

⊞ Frame 21 (1512 bytes on wire, 1512 bytes capt
⊞ Ethernet II, Src: 20:d9:20:00:03:00 (20:d9:2
⊞ Internet Protocol, Src: 128.39.202.234 (128.
⊞ User Datagram Protocol, Src Port: 3220 (3220
⊟ Cross Point Frame Injector
  ⊟ Header
      Word one: 0x0000002f
      0000 .... .... .... .... .... .... .... =
      .... 0000 0000 00.. .... .... .... .... =
      .... .... .... ..00 0000 0000 .... .... =
      .... .... .... .... .... .... 0010 .... =
      .... .... .... .... .... .... .... 11.. =
```

Figure 33 The "Sequence number" of the packet in the “Word one” field

3.8 *Iperf testing 3 – upload*

We wanted to see if we could find similarities to the behavior of the UDP download stream when performing an upload stream, i.e. sending UDP datagrams from the computer connected to internet via the phone. Another goal of the test was to see how buffering of datagrams was done when uploading. Iperf ran in server mode on the remote computer located at HiA with:

```
iperf.exe -s -u -i 1
```

And in client mode:

```
Iperf.exe –c <iptoserver> -u –b 128k
```

It should be noted that 128 kilobits per second is at the limit of what the phone can manage, and the operator Telenor states on their website [39] that max upload bit rate is 64 kilobits per second in UMTS. We performed two scenarios, one where the network had been idle for 1 minute (scenario 1), and scenario 2 where we pinged the remote host on HiA immediately before we started to stream UDP datagrams. This should ensure that the Radio Access Bearers are established.

3.8.1 Scenario 1 – RAB not established

Sender output:

```
C:\Documents and Settings\Erling\Desktop>iperf -c 128.39.202.234 -u -l 1470 -b 1
28k -i 1 -t 20
------------------------------------------------------------
Client connecting to 128.39.202.234, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1916] local 212.17.138.169 port 2443 connected with 128.39.202.234 port 5001
[ ID] Interval       Transfer     Bandwidth
[1916]  0.0- 1.0 sec  7.18 KBytes  58.8 Kbits/sec
[1916]  1.0- 2.0 sec  0.00 Bytes  0.00 bits/sec
[1916]  2.0- 3.0 sec  4.31 KBytes  35.3 Kbits/sec
[1916]  3.0- 4.0 sec  0.00 Bytes  0.00 bits/sec
[1916]  4.0- 5.0 sec  8.61 KBytes  70.6 Kbits/sec
[1916]  5.0- 6.0 sec  8.61 KBytes  70.6 Kbits/sec
[1916]  6.0- 7.0 sec  12.9 KBytes   106 Kbits/sec
[1916]  7.0- 8.0 sec  17.2 KBytes   141 Kbits/sec
[1916]  8.0- 9.0 sec  12.9 KBytes   106 Kbits/sec
[1916]  9.0-10.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 10.0-11.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 11.0-12.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 12.0-13.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 13.0-14.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 14.0-15.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 15.0-16.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 16.0-17.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 17.0-18.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 18.0-19.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 19.0-20.0 sec  15.8 KBytes   129 Kbits/sec
[ ID] Interval       Transfer     Bandwidth
[1916]  0.0-20.1 sec   240 KBytes  97.8 Kbits/sec
[1916] Server Report:
[1916]  0.0-16.3 sec   240 KBytes   120 Kbits/sec  93.135 ms    0/  167 (0%)
[1916] Sent 167 datagrams
```

The network had been idle for one minute, and all RAB should have been released. We can see some strange behavior of the sending of datagrams. From 0-1 secs 7.18 Kbytes is sent, then a pause. Something, probably related to establishment of the radio access bearers is done here. Paging is not an issue since the traffic is initiated from the mobile. We can see the server/receiver received data for 16,3 seconds, and interestingly, no packets are lost. All 167 packets are received correctly. 240 Kbytes are transmitted, 97.8 Kbits/sec seen from the transmitter and 120Kbits/sec seen from the receiver. The difference is because the receiver only received for 16,3 seconds, while the transmitter transmitted for 20 seconds.

3.8.2 Scenario 2 – RAB established

Sender output:

```
C:\Documents and Settings\Erling\Desktop>ping 128.39.202.234

Pinging 128.39.202.234 with 32 bytes of data:

Request timed out.
Reply from 128.39.202.234: bytes=32 time=210ms TTL=120
Reply from 128.39.202.234: bytes=32 time=248ms TTL=120
Reply from 128.39.202.234: bytes=32 time=235ms TTL=120

Ping statistics for 128.39.202.234:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 210ms, Maximum = 248ms, Average = 231ms

C:\Documents and Settings\Erling\Desktop>ping 128.39.202.234

Pinging 128.39.202.234 with 32 bytes of data:

Reply from 128.39.202.234: bytes=32 time=244ms TTL=120
Reply from 128.39.202.234: bytes=32 time=204ms TTL=120
Reply from 128.39.202.234: bytes=32 time=228ms TTL=120
Reply from 128.39.202.234: bytes=32 time=243ms TTL=120

Ping statistics for 128.39.202.234:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 204ms, Maximum = 244ms, Average = 229ms

C:\Documents and Settings\Erling\Desktop>iperf -c 128.39.202.234 -u -l 1470 -b 1
28k -i 1 -t 20
------------------------------------------------------------
Client connecting to 128.39.202.234, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1916] local 212.17.138.169 port 2444 connected with 128.39.202.234 port 5001
[ ID] Interval       Transfer     Bandwidth
[1916]  0.0- 1.0 sec  15.8 KBytes   129 Kbits/sec
[1916]  1.0- 2.0 sec  7.18 KBytes  58.8 Kbits/sec
[1916]  2.0- 3.0 sec  12.9 KBytes   106 Kbits/sec
[1916]  3.0- 4.0 sec  15.8 KBytes   129 Kbits/sec
[1916]  4.0- 5.0 sec  14.4 KBytes   118 Kbits/sec
[1916]  5.0- 6.0 sec  17.2 KBytes   141 Kbits/sec
[1916]  6.0- 7.0 sec  12.9 KBytes   106 Kbits/sec
[1916]  7.0- 8.0 sec  14.4 KBytes   118 Kbits/sec
[1916]  8.0- 9.0 sec  15.8 KBytes   129 Kbits/sec
[1916]  9.0-10.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 10.0-11.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 11.0-12.0 sec  14.4 KBytes   118 Kbits/sec
[1916] 12.0-13.0 sec  15.8 KBytes   129 Kbits/sec
[1916] 13.0-14.0 sec  14.4 KBytes   118 Kbits/sec
[1916] 14.0-15.0 sec  15.8 KBytes   129 Kbits/sec
[1916] 15.0-16.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 16.0-17.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 17.0-18.0 sec  17.2 KBytes   141 Kbits/sec
[1916] 18.0-19.0 sec  12.9 KBytes   106 Kbits/sec
[1916] 19.0-20.0 sec  15.8 KBytes   129 Kbits/sec
[ ID] Interval       Transfer     Bandwidth
[1916]  0.0-20.2 sec   294 KBytes   119 Kbits/sec
[1916] Server Report:
[1916]  0.0-20.5 sec   294 KBytes   117 Kbits/sec  99.216 ms    0/  205 (0%)
[1916] Sent 205 datagrams
```

From the console output we can see how we pinged the remote host to establish the radio access bearers before starting the streaming. The streaming starts more smoothly here, although with some lower bit rate from 1 – 2 seconds. The fact that the receiver received datagrams for almost exactly the same amount of time that the transmitter transmitted indicates that there has been no significant buffering of data in the network. The packet flow from host one to host two has gone trouble less. Neither in this scenario are any packets lost. In both scenarios we clearly supersedes Telenors numbers for maximal bit rate.

3.9 Tracing the route between the hosts

```
C:\>tracert 212.17.139.28 -w 20000

Tracing route to tmi212017139028.mobil.telenor.no [212.17.139.28]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  grooseveien-gw.hia.no [128.39.202.1]
  2    <1 ms    <1 ms    <1 ms  grimstad-gw.uninett.no [128.39.0.137]
  3     1 ms     1 ms     1 ms  kristiansand-gw.uninett.no [128.39.0.246]
  4     6 ms     6 ms     6 ms  stolav-gw.uninett.no [128.39.47.109]
  5     6 ms     6 ms     7 ms  oslo-gw1.uninett.no [128.39.46.249]
  6     7 ms     8 ms     7 ms  nix-gw.telenormobil.no [193.156.90.13]
  7     7 ms     7 ms     7 ms  t1-vir-internett.mobil.telenor.no [212.17.134.34]
  8     *        *        *     Request timed out.
  9  2049 ms   659 ms   659 ms  tmi212017139028.mobil.telenor.no [212.17.139.28]


Trace complete.
```

As these results show, our host at the other end is connected to the internet on a fast and quick-responsive internet connection, as response times are minimal from the host to the Telenor APN (t1-vir-internett.mobil.telenor.no).

4 Summary and discussion

In this chapter we will make a summary and comment the test results from the previous chapter. Further we will discuss which performance indicators an end-to-end testing application should include, and why. We begin this chapter with a short introduction and some notes on the mobile phone we used.

4.1 Nokia N70

The phone assigned to us was the Nokia N70 mobile phone. The phone features the standard GSM (2G) with GPRS, often called 2,5G and the enhanced version EDGE, and the newer third generation cellular network, UMTS. The phone can operate in GSM-only mode, or in the so-called dual mode, were it preferably uses UMTS but falls back to GSM in areas without UMTS coverage. The phone is GPRS class B, and GPRS and EDGE Multislot class 10 capable (four timeslots down and one up, or three down and two up), which should give maximum EDGE uplink speed of 118,4 kbps and downlink of 236,8 kbps with coding scheme MSC9 used. These numbers are also given on the Nokia website, so we can assume the phone is coding scheme MSC9 capable. According to Nokia’s website, maximum 3G downlink speed is 384 kbps, and uplink 128 kbps. The phone features the Symbian User Interface S60, OS version 8.1.

The phone does not support all the AT-commands in [18] and [19], on the other side, it’s important to mention that all of these commands are “optional” to implement according to the standard. The unsupported commands affecting our work are listed in chapter 3.1.

4.1.1 Connecting the Nokia N70 to the PC

A USB cable comes bundled with the N70 together with the PC Suite software for Microsoft Windows. We installed the software and connected the phone with the cable, but with no luck. We tried on two laptops but they simply refused to recognize the phone. After some searching on the web and reading some forums, we found that this was a well known issue. Various tricks to make the pc recognize the phone was being suggested, but the one that worked for us was like the following: start the PC Suite program, click “Get Connected”, select “Cable Connection” and click next. If the pc cannot connect to the phone, turn the phone off and pull the plug. Turn the phone on again and when the booting is completed, connect the USB cable again. Windows should now report that new hardware is detected and being installed, and PC Suite will recognize the phone and connection is established. It seems like this was only necessary the first time to get the drivers installed. It was detected at once the next time we connected the phone to the PC with the USB-cable.

The N70 can also be connected via Bluetooth and Serial Cable. The Bluetooth connection worked without any problems. It is uncertain if the same problems with the drivers as when we connected with USB cable would occur if we connected with Bluetooth at first. However, it is likely to believe it was a problem with USB-only.

The phone had only one COM-port available, which was the dialup modem. We also tried to connect with both USB-cable and Bluetooth at the same time. The Bluetooth connection reported a dialup modem too, and even if they were assigned different COM-ports, we were only able to connect to one of them at the same time in HyperTerminal. This was as expected, as it is the same dialup modem. In comparison, the Sony Ericsson T610 phone, when connected with Bluetooth, reports one dialup modem, and two serial ports. We were able to establish connections with all three COM-ports at the same time using HyperTerminal on the T610. The idea with several connections is quite interesting. This makes it for instance possible to execute AT-commands while a dialup connection is established. This could be useful in instances where we e.g. want to extract signal quality while we have a dialup connection established. Nevertheless, this is not possible with the N70 since it is no support for several communication ports between the N70 and a computer.

4.2 Comparison between EDGE and 3G/UMTS

4.2.1 Comparison

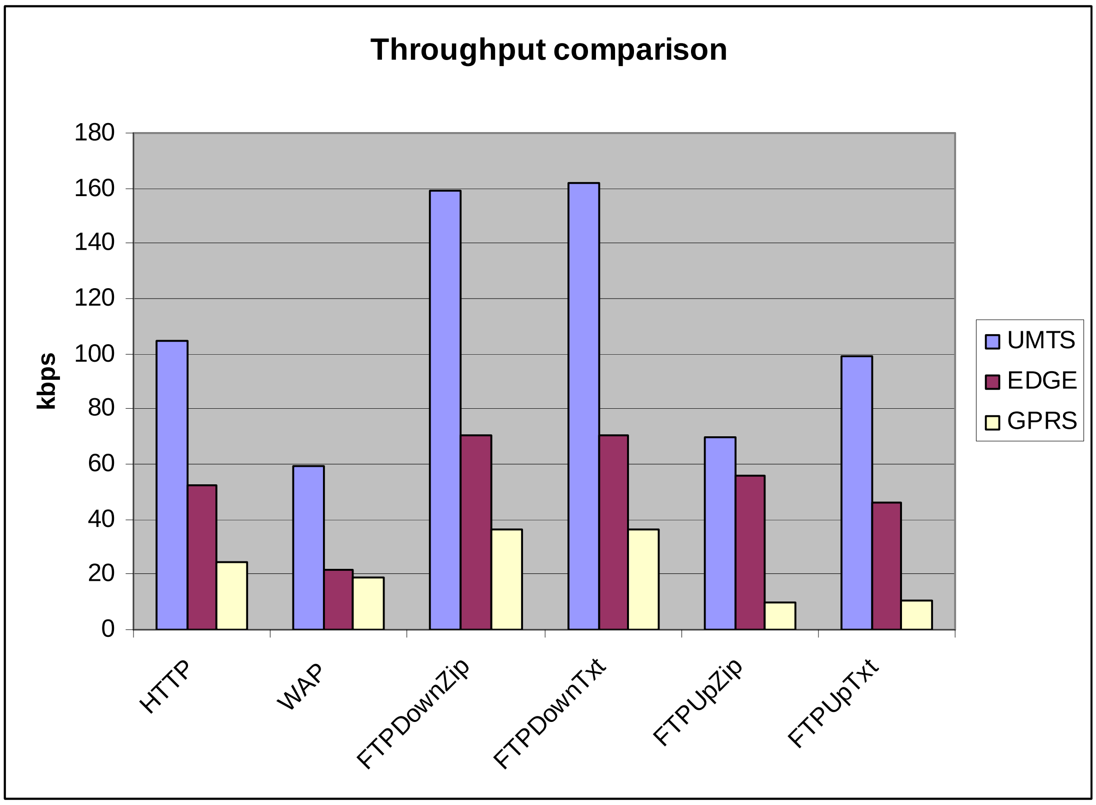


Figure 34 Average throughputs in kbps for UMTS, EDGE and GPRS. We have not tested ordinary GPRS, but included sample data for illustration purposes. UMTS and EDGE results are measured using Telenor's network.

As we can see in Figure 34, 3G gives significantly better throughput than EDGE. Overall, the throughput is at least twice as high. FTP downloads in 3G is around 160 kbps, while in EDGE around 70 kbps. On the other side, EDGE clearly is an improvement over plain GPRS, which typically provides an average throughput in the mid 30s. The only result in our test that is a little surprising is the ftp upload of the zip file where 3G performs rather disappointing and EDGE surprisingly good. It should however be taken into consideration that we received one very bad test result in the 3G test for FTPDownZip, FTPDownText and FTPUpZip that pulled the average down a bit. 3G gave very stable zip upload speeds of around 80 kbps, with exceptions of one at 24 kbps and one at 67 kbps. Overall, at least for the uploads, 3G seems to be able to deliver more stable bit rates. Figure 18 and Figure 22 illustrate this.

Throughput [Kbps]	***EDGE***	***UMTS***
HTTP_download	52,3	104,5
WAP_download	21,9	59,6
FtpDownload.zip	70,2	159,3
FtpDownload.txt	70,4	162,2
FtpUpload.zip	56,0	70,1
FtpUpload.txt	45,9	99,3

Table 6 Average throughput in EDGE vs. UMTS.

Like Chakravorty et al. researched and stated in [26] for GPRS, the HTTP performance is bad compared to ideal TCP throughput. As we see in Table 6, the HTTP_download throughput in both EDGE and UMTS is lower than the Ftp downloads. In EDGE the HTTP download is 74,5 % of the ftp download, and in UMTS 65,5%. (FtpDownload.zip used). The reason the performance is not as bad as in Chakravorty's measurements is most probably due to the fact that our test-website did consist of 15 files, which must be said to be relatively few for a modern webpage. In contrast, the CNN page Chakravorty used consisted of more than 100 embedded objects. The total download of the web page was 172,5 kilobytes and for the ftp downloads 199 kilobytes. Our results do confirm Michael Meyer's [25] writings that TCP performs well over GPRS, much due to its reliable link layer, and Chakravorty's concluding that although TCP performs well, HTTP does not. To reflect a real website more realistic, it should be considered if the number of objects in the HTTP test should be increased.

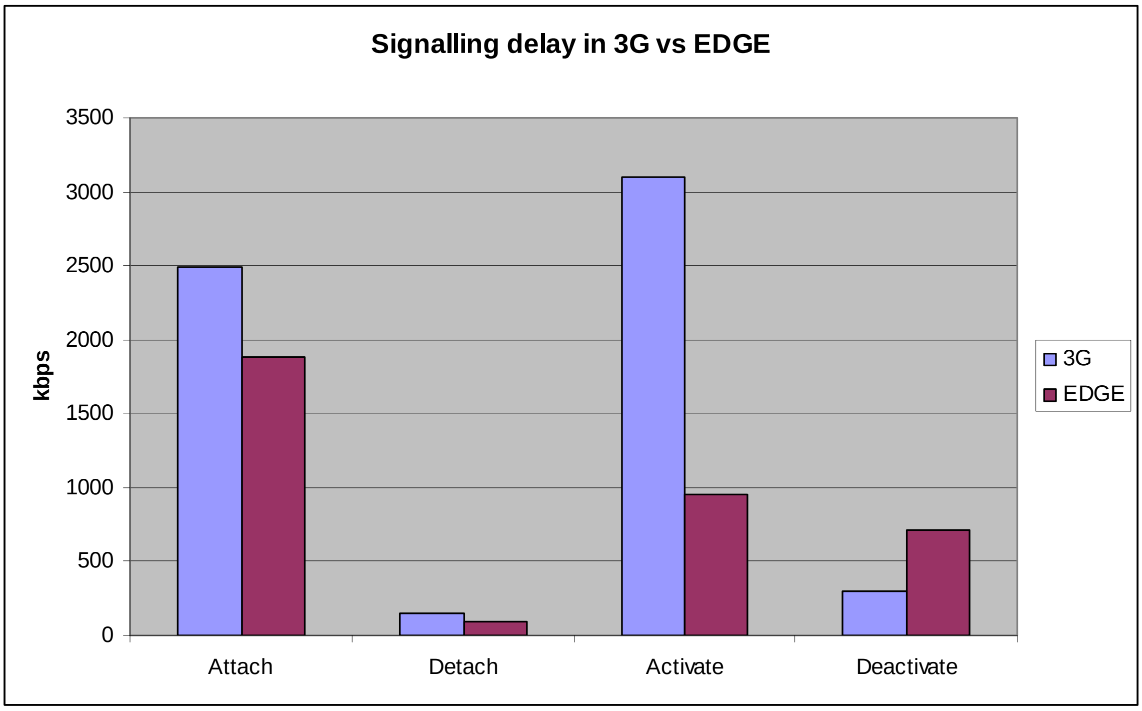


Figure 35 Average delays for signaling in EDGE and 3G.

Regarding signaling in UMTS vs. EDGE, only noticeable difference is that activate takes much longer time in 3G, and to a lesser degree, attach. The detach time is unrealistic low. We have measured Detach to take lower time than RTT, which is not possible. The answer is that the phone does not wait for any response from the network. I.e. we can conclude that these values are not correct. This behavior from the phone can be excused by the argument that no one really cares how long the detach time is.

From our measurements we can conclude that EDGE is, not surprisingly, faster than GPRS, and that 3G is even faster than EDGE. These results were expected. The network operator would have a problem if this were not the case. In EDGE, 236,8 kbps bandwidth is only achieved in max peaks for a short period of time, user data throughput of about 70 – 80 kbps for a file transfer download for a file of the size of 200 Kbytes can be expected, and upload speed of about 50-60 Kbytes per second for a file size of about 50 Kbytes. The files used is maybe a little to small, as the effects of slow start will affect the transfer speed more than it would have done with a larger file. In 3G, transfers of up to 384 kbps are only achieved in max peaks for a short period of time, and user data throughput of about 160-170 kbps for a file transfer download and 70 – 100 kbps for an upload with same file sizes as in the EDGE case can be expected. 3G seems to be a slightly more stable considering bit rates, as the bit rate provided by EDGE varied more from loop to loop.

Regarding the RTT measurements, and radio access bearer delays, our measurements show that 4,5 seconds of idle time before the first ping is not enough to ensure that the radio access bearers are taken down in 3G.

4.3 Dialup and RTT test

Dialup takes much longer time in Netcom, about 22 seconds compared to about 8 in Telenor. Telenor has some issues when it comes to RTT after the network has been idle for some time, with a very large RTT. This is not the case in Netcom; even though we had an even higher network idle time here. We used an idle time of 11,5 seconds in Telenor and 14,5 seconds in Netcom. This can be related to radio interface setup or internal signaling in the core network. Such a high delay is unwanted especially in alarm systems and systems that need low response time.

It is hard, maybe impossible, to determine the exact reason for the high delay from an “end to end” tool, and without any knowledge from the “inside” of the network. But we have shown, and identified low performance and problems with the network. It is then up to the network operator to look into their network and find the exact reason.

An interesting point in Figure 17, is that the radio access bearer (RAB) establishment time seems to not affect the ping in that measurement, as the first ping is no higher than the second. However, in this test, were we had a larger network idle time before the first ping (11,5 secs in Telenor and 14,5 in Netcom), one can clearly see how the RAB establishment affects the RTT in the Telenor case. The RTT is then as high as about 4-5 seconds. This indicates that the default settings of 4,5 seconds of idle time before the first ping is not enough to measure the impact of RAB establishment in UMTS networks. At least it is the case for Telenor.

The long delay could be due to establishment of the dedicated DCH channel, although one would believe that for such small amount of data, to use a common channel, more precisely, the RACH, would be more efficient. A delay of 4-5 seconds could cause problems for applications and services with high demands for quick response e.g. alarm services. It should be in the operator’s interest to lower this delay, as it will affect the user’s experience. Slow response gives a bad impression of the network, and is annoying for the end user. The establishment of the TBF in GSM-GPRS takes significantly lower time, the RTT after the network has been idle for some time is about 800 ms, compared to about 300 ms on the second ping. In other words, the establishment of the TBF takes about half a second.

4.4 Iperf testing

We can see that in scenario 1 in chapter 3.6 we had significantly higher packet loss and latency than in scenario 2, where the radio access bearer was established. The packet loss is 21 % versus 3,2 %, with the major loss in the beginning for both scenarios. The reason we still had a pretty large packet loss in the beginning of scenario 2 could be because the echo request from the PING only established a connection on a shared common channel, while the UDP datagram stream needs to establish a DCH channel. If this is the case, it may be that it is the paging that causes the great packet loss. We can see that compared to scenario one, where we received datagrams for 25,7 seconds, we now received datagrams in 30,1 seconds. These measurements do not contradict the results regarding RAB establishment that we discovered in the RTT and dialup measurement.

A common conclusion from both the scenarios is that once the RAB is established the network provides a reliable connection as we have little packet loss when the streams are "established". This can be contributed to the reliable link layer over the radio interface. It is therefore easy to conclude that the packet loss in the beginning of the transmission is due to buffer overflows due to paging.

This measurement is supported by several other Iperf-tests we have performed, and the RTT measurement in the "Dialup and RTT" also supports it. The RAB establishment time takes a significant amount of time, and should be minimized as much as possible.

In the second Iperf test, chapter 3.7, we found strong indicators that there is some buffering of the incoming packets. The RAB establishment takes some time, but the first packets, approximately 20, is in both scenarios received correctly. From these two measurements we have made it clear that

1. The establishment of the Radio Access bearers and paging takes a significant amount of time.
2. Buffering in the network guarantees delivery of the first packets. Only when the buffer is full, it starts dropping packets.

Perhaps the most interesting difference between the download and upload stream testing is that there is no packet loss on the uplink. One explanation for the heavy packet loss with the download stream without RAB established could be paging, i.e. the network has to search for the phone before it can establish the RAB. Paging is only an issue with network-initiated traffic, thus it is not done in the uplink scenarios. However, we still have an amount of packet loss when the RAB is established downlink. This contradicts this explanation. What makes it difficult is that there could be several effects that are the reason for the packet loss. It may be that paging increases the packet loss. The good performance of the uplink stream indicates that the link layer provides a good bearer, and that the buffers are large enough to wait for the RAB to be established. It is interestingly good that no packets are lost upstream.

4.4.1 Experiences after performing a test with Iperf

The application worked as expected, with one exception. It seems like the bandwidth graph plotted in Jperf only is correct for the first ten seconds. This only affects the graph when performing measurements exceeding ten seconds. An annoying error message also appears as the last output:

```
[1932]  0.0-30.1 sec  1.25 MBytes   349 Kbits/sec  16.624 ms   29/  920
(3.2%)
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
read failed: Connection reset by peer
[1932] WARNING: ack of last datagram failed after 10 tries.
recvfrom failed: Connection reset by peer
```

Even though it reports the summary line, 1932, something goes wrong after that. Firewalls were turned off on both computers running the experiment. Since UDP has no acknowledgements, it is an application layer acknowledgment implemented in Iperf that it is not able to receive. When we ran the application over the fixed internet, this problem did not occur.

4.5 Proposed enhancements of an end-to-end test tool

The last part of this chapter will be dedicated to the discussion of what an end-to-end performance measuring application should implement and what its features should be. There are some main measurements that strongly give an impression of the user performance in any data network. These measurements are strongly related to active KPIs.

4.5.1 Throughput

User throughput measures how fast the user is able to transmit and receive data. It is computed by taking the amount of data transmitted divided with the time it took to transmit it. There is a difference between user data throughput and network link throughput. Higher layers headers will degrade the throughput. Delays will also degrade throughput. This is a reason that HTTP will give lower throughput than for instance FTP. FTP is the de-facto standard for measuring user throughput. One drawback with FTP is the delay with TCP connection and slow-start. The larger the size of the file is, less do these effects affect the throughput. This because the delays will be very small compared to the overall transmission time. How large the file should be, is dependent on the bit rate of the network, but [31] suggests a value for 2.5G systems to be between 500 kilobytes and 1 megabytes. We agree with these numbers as we after our measurements would say 200 kilobytes is a little to small in EDGE and UMTS. To measure HTTP throughput one should use many small files. A modern webpage may consists of over 100 objects, that all are to be downloaded with GET requests. We have shown that HTTP throughput with many small objects are bad, much due to high latency.

4.5.2 Round trip time

The round trip time (RTT) is the time it takes for a packet to go from one host, to another and back again. It is usually measured with the PING application which exists for most operating systems. The RTT strongly affects the upper layers, like TCP. The initial effect of radio access bearer establishment (e.g. TBF establishment in GPRS) should not be included in the result; therefore the ping delay should be averaged over several packets that are transmitted one after the other. This should ensure that the bearers are not released.

4.5.3 Establishment time

The establishment time at the radio level is the time it takes to establish a radio bearer. In GSM-GPRS this means the time it takes to establish a TBF between the terminal and the network. One can also define establishment times at other levels, e.g. establishment time of a TCP connection. Sending an ICMP packet with very low payload after the network has been idle for some time, will in most cases give an appropriate measurement of the radio access bearer establishment time. This is an active way to measure it, and most probably a realistic implementation in an end-to-end test tool.

4.5.4 Cell reselection times

Outage times when reselecting cell will affect upper layers. How much is an interesting question. In [33] it is defined two types of measurements:

1. Radio outage time, which is the time from the radio connection is stopped in the old cell, to the radio connection is established in the new cell.
2. User outage time, which is the time since the user received the last data packet in the old cell, until it received the next packet in the new cell. This outage time can be measured with a TCP protocol analyzer, like ethereal.

The latter is definitively the one that easiest can be measured from an end-to-end test tool. An interesting point is if the end-to-end test tool can detect if the terminal has connected to a new cell, and automatically calculate the outage time. This should in theory be possible by extracting cell id from the MS. Limitations in the phone however, make such measurements impossible to perform. For instance, the Nokia N70 phone we have used in these project, does not allow a secondary communication port with the phone when dialup is activated, i.e. it provides only one COM port. This means that one has to wait until dialup is deactivated before one can establish a new connection with the phone and execute AT-commands for cell identification etc.

If, however, the phone supports several COM ports, as some Sony Ericsson phones do, the question is if the phone can notify when it has changed cell, or if it is possible to do only by executing AT-commands. If the latter is the answer, automatic measurements of outage time will be difficult to implement.

4.5.5 What the application should report

It is important to keep a proper documentation of the test. If all setup parameters are properly documented, comparison between different tests can be done properly, and troubleshooting of strange results is easier. To be able to compare measurements all measurements have to have the some configuration and prerequisites. Some documentation of configuration must probably be done manually, but the best is if most are done automatically. Things that could be done automatically should be done automatically. Things that must be documented manually is easier to forget and makes the use of the application more cumbersome. The application should at least document and log the following:

Application version number
The application should log its version number.

Operating system
The application should log the operating system (OS) and version number used on the computer. One reason for documenting which OS is used is that different OS's may implement the protocol stack differently, and particularly has different TCP implementation, e.g. different window sizes, which could affect the performance. For

instance, Windows 98 as default uses another window size than Windows XP, though this can be changed.

MS used
Which MS is used, and its capabilities should be documented. This includes networks supported, e.g. GPRS, EDGE and UMTS. If GPRS/EDGE is supported, its multislot class should be documented, as this greatly affects the performance. If UMTS is supported, any max bit rates should be documented. More specifically, the following should be documented:

- Model name and revision number
- Operating system and version number
- Network capabilities (GSM? GPRS? EDGE? 3G?)
    - When and how does it change between GSM and 3G, can it be done automatically and/or manually.
- Multislot class (GPRS)
- Max uplink/downlink bit rate in 3G
- Not only network capabilities, but also the network actually used, must be documented (GSM or UMTS?)

Hardware used
One should document the terminal equipment (TE) used, in most cases a personal computer. The connection between the computer and the phone is also of interest. An infrared connection is slow and could be a bottleneck, compared to a high speed USB 2 compatible connection.

Network configuration
Configuration of the dialup connection should be documented, e.g. if any data compression is used. This documentation must probably be done manually.

Date and time of day
The date and the time of day the test was performed should be documented. This is important for comparison with other test results, as a test performed at 3 o'clock in the morning hardly can be compared with a test done at mid-day.

4.5.6 What the application should do

The application is an "Automatic Application Testing and Monitoring" tool according to [31]. The tool automatic enables the wireless connection and executes a scheduled set of tests. The application minimizes the human factor of the tests, and ensures that the test is performed in the same way. The application measures main KPIs related to each service it tests. It should feature some of the things a Drive Test Tool features, or the features that is possible to implement, i.e. information that is possible to extract from the MS.

Information extracted from the MS
This is information that requires the execution of AT-commands and thus needs an available COM port to the MS. If the MS supports only one COM port, this has to be done before (or after) the dialup connection is activated.

Network used
Is it GSM-GPRS, GSM-EDGE or 3G? This could be difficult to extract from the MS itself, and must probably be documented manually. Some phones support the feature of setting it to either GSM or 3G mode. The Nokia N70 supports either GSM or the so-called “dual modus”, where it connects to the available network, preferably 3G. To decide if ordinary GPRS or EDGE is used in GSM, one may have to look up in the operators coverage map.

Cell ID
The Cell ID can be extracted with an AT-command.

Block error rate and signal quality/strength
The block error rate (BER) and received signal strength indicator (RSSI) may be extracted from the MS with AT-commands. In all our tests with the Nokia N70, the reported BER was 99, which is “not known or not detectable”.

Signaling messages
Signaling, i.e. attaching and detaching, and pdp-context activating and deactivating to GPRS should be done and time measured.

Other
Features that most probably is not possible to implement, but that would be highly interesting:

- Number of TSLs used (GSM). The network does not have to provide as many timeslots as the MS supports.
- Coding scheme used in GPRS and EDGE.
- Channel used (WCDMA)
    - It can be figured out when a DCH is established by pinging with ICMP packages with increasing payload. A recurring problem is however that pinging with packets with large ICMP payload often is blocked by the network.

Other things that could be interesting:

- GPS and digital maps to show the exact location of the terminal during mobile tests.

4.5.7 Service specific measurements

In this chapter we describe the following services that should be measured. These services are particularly important for the users experience and will give a good measurement of the user experience. Optimizing the network to give good performance for these services should definitively be in the interest of the network operators. Measurements that should be implemented:

Ping
The Ping service or application itself is not an application that the end user is particularly interested in, but the RTT, which Ping measures, is a network parameter that greatly affects the overall performance of the network. Ping should be used both

to measure RTT and establishment delays. RTT can be measured by sending e.g. 10 pings after an initial ping which is sent to establish the radio bearer. Establishment delays can be measured by sending a single ping after the network has been idle for a period of time (long enough time so we are sure the radio bearers are released). The idle time that is needed, is dependent on the network configuration, and/or if optimizations to keep the TBF's (in GSM) alive after transmission is ended is used. Pinging with packages with different ICMP payload is interesting to see how the size affects the RTT. Large packages will probably be delayed due to bit rate capabilities in GPRS, but will perform better in UMTS. Small packages are ideal for measuring the delay only. A problem is that network operators often do not allow pinging with packets of a certain size and larger. When pinging from a computer utilizing a GPRS connection with two major Norwegian operators with packet sizes of 12 bytes, 200 bytes, 468 bytes and 1472 bytes, only the 12 bytes packets succeeded.

FTP
FTP should be used to measure TCP user data throughput in both directions, i.e. uplink and downlink. The size of the file used should be so big that the effects of TCP establishment and slow start are minimized. A filesize of 500 kilobytes to 1 megabyte is suggested in [31]. Some points should be taken into consideration:

- TCP parameters should be standardized, to ensure that all tests have the same configuration, e.g. on different operating systems. If e.g. changing of TCP window size is not possible, it should be documented.
- Un-optimized TCP parameters may give an incorrect view of the networks actual throughput.
- The test application should if possible report TCP parameters.

The application should also report the FTP start-up failure rate and FTP abort rate. These measurements will say much about the reliability and quality of the bearer, and thus the end users performance.

HTTP
HTTP should be used the measure the performance of traditional web surfing, and the behavior of interactive service and the impact of request-response delays. HTTP 1.1 should be implemented**.** One should use a real webpage with several objects (images, text and applets). A modern web page may contain over 100 objects. To use a public web page like a newspaper on the web is tempting, but not probably a good choice since they often change, and thus the size and number of objects of it changes. To make a snapshot of a webpage (wget –r), for instance www.vg.no may be a good idea. To design a page especially for the test is probably a good choice too. There is probably a good choice to not make the total download to large. Throughput testing is done in the FTP test, other measurements is more interesting here. In addition to throughput/delay before the site is loaded, these measurements should be reported:

- Access failure rate, as this says a lot about the quality and reliability of the connection.
- Abort rate, also says a lot about the reliability
- Access time, as this says something about when the user can start reading the text.

Streaming and UDP throughput
Streaming should be used to measure the quality of real time services. Quality and delay should be checked for different video qualities. Most interesting is RTP/UDP streaming, and not “pseudo streaming” with HTTP and TCP. UDP/Iperf could be used to simulate streaming.

UDP throughput should be used to measure the user-data throughput with a minimal of headers, and to measure packet loss. Iperf can be used. It could also be used to simulate real time streaming. Packet loss with certain bit rates should be measured. Packet/datagram loss and bit rate is important indicators. Streams with bit rate similar to the bit rate of real life streaming media are interesting.

Implementation of a self-developed UDP-test measurement tool is not difficult but requires the use of an active server, i.e. a server that transmits UDP datagrams. Throughput should be pretty trivial to measure, while packet loss needs the implementation of an application protocol with sequence numbers to use. Due to the nature of UDP streams, i.e. connection less, possible problems are firewall and NAT issues since the traffic is initiated from the network.

WAP
WAP performance is an interesting feature, but maybe hard to implement? The Wireless Application Protocol is not a single protocol, but a collection of protocols ranging from transport protocols to application protocols and requires the use of an APN intended for WAP.

4.5.8 Application development platform

Two of the major development platforms are Sun’s Java and Microsoft’s .Net. Java has one great advantage in that it runs on both Linux and Windows. However, communicating with com ports can probably not be done without OS specific API’s or drivers. There are most probably different approaches for communicating with com ports in Linux and Windows. .Net binds you to a Microsoft platform (at least if a GUI is made for the application). Visual Studio and .net provides a very user-friendly IDE.

4.5.9 Application (graphical user) interface

Should the application use a graphical user interface or be commando line based? Both have its pros and cons.

Graphical user interface
A graphical user interface (GUI), if well designed, provides a nice and professional look. To make a GUI that is both good looking and functional is a hard task. However, a GUI based application could be easy configurable, and make unfriendly configuration files obsolete. Configuration of the test and selection of tests to be performed can be selected with graphical widgets.

Commando line based

A commando line based application is not as challenging to design as its GUI counterpart. Configuration of the application could be done with input parameters when it is executed or by a configuration file. The latter is for this type of an application the best alternative since a configuration file documents the configuration of the application when the test was run. When the application is executed, not much of user input is needed until it has finished. A commando line based application is in some cases easier to port to other platforms and operating systems. This is the case if a .net solution is chosen. The ability to run .net applications on Linux is realized with Mono [40] but it lacks the support of GUI based applications.

Presentation of the collected data

Presentation of the collected data is a very important point. The presentation should give a clear and correct view of the data. It can be presented in several ways:

- Tables with actual data (averages), see Table 7.
- Graphs, see example in Figure 36.

Throughput [Kbps]	**Last**	**Average**	**Limit ***
HTTP_download	48,3	52,3	n/a
WAP_download	21,6	21,9	n/a
FtpDownload.zip	86,0	70,2	129,1
FtpDownload.txt	57,5	70,4	96,0
FtpUpload.zip	76,9	56,0	76,9
FtpUpload.txt	42,2	45,9	70,5

Table 7 Example of presentation with average values.

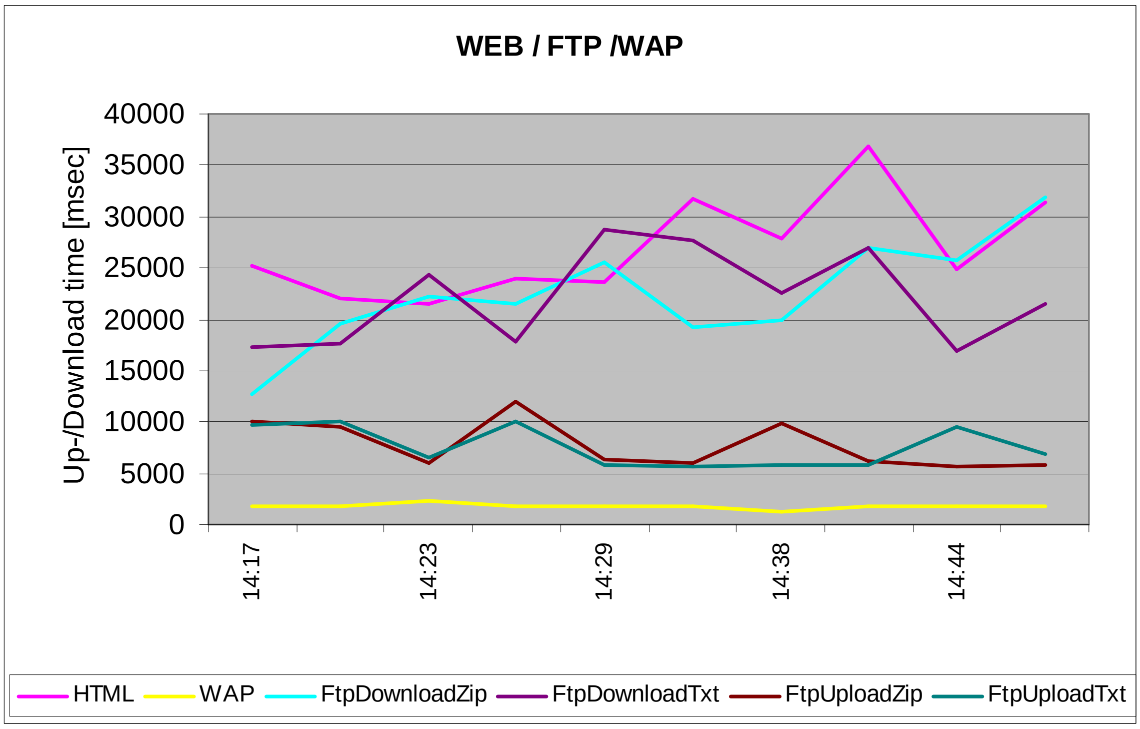


Figure 36 Example of graph presentation of data.

One has to decide if the presentation of the data should be done in the test tool itself, or in an external application. If the latter is chosen, the test tool will only log the measured data to a log file. It is then the task of another application, e.g. a spreadsheet to load the collected data from the log file and present it. This approach makes us reuse already available applications and resources

Jain has in [30] developed a few guidelines for graphic chart presentation:

1. It should require minimum effort from the reader. It should be easy to understand and get the message from the chart. Avoid using to many curves, bars or components on a single chart. Use text instead of symbols.
2. Make the information on the graph as informative as possible. Label the axes informative. It is a must to label the axes. Jain even suggests using direct labeling instead of legend boxes, though it is in our opinion that this will seem unorganized and messy in many cases.
3. One should present as much information as possible with as little ink as possible. One should try to maximize information. Use only grid lines if it really is necessary. Avoid unnecessary information.
4. Use commonly accepted practices. People are used to seeing (0,0) down to the left and seeing the variable plotted along the y-axis. Do not break this practice unless it is necessary.
5. Avoid Ambiguity. Try to make the graph easy to read. This is done by showing coordinate axes, scale divisions and origin. Use different colors on different variable lines. Presenting many y-variables on a single chart is generally not a good idea and one should try to avoid it.

The application should try to extract and sort out and present bad results. An automatic score system is interesting. Development of a score system is not an easy matter, and requires a lot of effort to fine tune the algorithm used to give the score. If the score system introduced is based on giving a score in percent of an ideal system, one has to decide what performance such an ideal system has. What is the RTT and bit rate in an ideal network?

4.6 Other applications that should be used in addition to the test tool

A protocol analyzer like Ethereal should be used to analyze the network interface and the protocols. It is useful for detecting if unwanted programs accessed the internet (mail-checkers, instant messaging, windows update, clock synchronization etc.) during the test phase, and also for detecting effects of e.g. RLC retransmitting and outage times at the TCP layer. Ethereal gives the possibility of filtering out and following a TCP connection from start to end. An application like Ethereal is free software and open source (GNU General Public License/GPL), and could be integrated with the test tool. This would without doubt be a very cumbersome and time-consuming task, but at least possible. This would also do it necessary to release the final test tool under an appropriate GPL compatible license.

The windows performance counter is not a packet analyzer, but it can be configured to measure traffic in and out on the network interfaces. It can be configured to measure many useful things, such as UDP datagrams per second, TCP segments per seconds, or retransmitted TCP segments per second. In our testing we have measured the following:

- Bytes Received/sec
- Bytes Sent/sec
- Bytes Total/sec
- Packets Received Errors
- Packets Received/sec
- TCP Segments Received/sec
- TCP Segments Retransmitted/sec
- TCP Segments Sent/sec
- TCP Segments/sec (Sent or received segments per second)
- UDP Datagrams Received/sec
- UDP Datagrams Sent/sec
- UDP Datagrams/sec (Sent or received datagrams per second)

The Bytes total, and TCP Segments/sec and UDP Datagrams/sec is the total, i.e. sent + received for the respective protocol. Omitting these is perhaps a good choice, if both sent and received are measured too. The interval of measurements can be configured, but new values every second is often appropriate and a good choice, and was what we used in our measurements. The collected data can be exported in several formats, we used a comma delimited text file, which is easy to import into the Microsoft Excel spreadsheet.

4.7 Optimizations possible for network operators

As seen in this project, many of optimizations that can be done, especially with the TCP protocol, are optimizations that are out of the scope for network operators. Some of the optimizations are however up to the network operator to deal with:

- Latency decreasing.
- TBF release time counter, i.e. not released immediately when sending or receiving stops. DCH channel release time in UMTS.
- Time-slot usage in GPRS.
- Using the optimal coding scheme.

To decrease the RTT, or the latency, should be highly prioritized by the network operator. We have also seen that introducing optimizations to keep the radio bearers alive after transmission has ended is an effective way to handle and eliminate long establishment delays. The operator must also decide how many timeslots to allocate for packet switched data traffic, and how many should be reserved for this use.

Optimizations not possible for network operators

When analyzing the measured results, it is important to keep in mind those parameters that are the application vendor’s responsibility, and not the network operator’s.

- TCP window size.

- Increased initial window (TCP)
- TCP segment sizes.
- Selective Acknowledgments (TCP)
- TCP Timestamps to allow RTT computation at each segment.

Optimizations dealing with transport and application layer protocols are out of the scope for network operators. It is the application and application server vendor’s responsibilities to fine tune these parameters. It is however, very important to document and report these parameters, since it affects the test results, and may be different from system to system.

4.8 Discussion summary

We have evaluated an existing internet throughput measurement tool, Iperf, to see how it can be used over wireless links. Our thought is that an UDP streaming test tool is useful as it could be used to simulate multimedia streams of various bit rates. It then gives a view of the available throughput for streaming services that do not use TCP and thus is not affected by TCP’s issues over wireless links. TCP is a much used transport protocol and suffers from some issues or characteristics with wireless networks (chapter 2.10.2). Especially high latency and RTT, which makes its slow start mechanism perform badly. We have shown by measurements that while TCP performs relatively well with large files, e.g. for FTP transmission of one large file, it performs badly for traditional HTTP, much due to the high latency of the link (chapter 3.3.3.3 and 3.3.4.3). Optimizations to the HTTP protocol, e.g. pipelining will give better performance for the end user.

Our measurements with Iperf reveal some issues with high access delays (chapter 3.6). We can address packet loss in downlink streams to either paging or radio access bearer establishment, or both, in Telenor UMTS. Uplink we have approximately the same delay, but without the packet losses. The difference is that no paging is done with traffic initiated from the phone, so it is tempting to blame the paging for the packet loss. One interesting fact we discovered was that we superseded Telenor’s uplink bit rate in UMTS. Telenor states 64kbps, but we were able to stream UDP datagrams at the rate of approximately 120kbps.

In chapter 3.3 we benchmarked EDGE and UMTS and showed that UMTS gives significantly better performance regarding throughput and latency, however, in both technologies; HTTP performs badly compared to raw TCP throughput as seen in FTP. This can be contributed to the high latency of the links and thus that TCP performs badly for small files. HTTP mostly consists of many small files, which leads to a “go-and-wait” behavior, while FTP often is about downloading larger files. The slow start of TCP is very slow on high latency links, and although good throughput can be achieved, it takes time.

Measurements of both Netcom’s and Telenor’s 3rd generation cellular networks shows that Telenor has some issues with high access delays in the form of high establishment time of the radio access bearers (chapter 3.4). Netcom performed somewhat better regarding this, but had much larger dialup access delay than Telenor. Dialup delay time would however by most people be seen as less important than performance when connection is established. Such high delays may introduce problems with systems that need low latency, e.g. alarm systems or real time monitoring.

In chapter 4.5.5.5 we have discussed and made suggestions for what an end-to-end test tool application should report of configuration information and system information. Proper and systematical documentation of the test setup including configuration details, of phone, network used and computers is necessary to make the analysis of the results possible.

We have looked at service specific parameters in chapter 4.5.7 that should be measured to document the performance of that particular service. Round Trip Time measurements should be done, as it greatly affects the performance of all TCP based application protocols, e.g. FTP and HTTP. To measure FTP performance we suggest using a file of 500 kilobytes for GPRS and 1 megabyte for EDGE and UMTS. Using large file sizes reduces the effect of slow start on the reported throughput. The filesize used for upload may be lower, due to lower upload bit rate of the cellular networks. The FTP failure rate should be reported as well.

For measuring HTTP performance, we suggest making a snapshot of a real web page, like www.vg.no, and store it statically on a web server controlled by the ones performing the measurements. This will ensure that performance is measured on a web page that is similar to those in real life. A Modern web page consists of many embedded objects, often over hundred. The go-and-wait behavior of HTTP combined with high latency links greatly affects the total download time and the total throughput.

UDP throughput measurements with different bit rates give an impression of audio and video streaming capabilities. A tool like Iperf measures jitter and datagram loss, and may be used for simulating multimedia streams. Jitter measurements are important for deciding buffer sizes.

Teleca has a good product for testing end-to-end performance in cellular IP networks in TWSE2E. There is always room for improvements, and among the interesting things to research further on and eventually implement is UDP streaming measurements. TWSE2E is designed with traditionally client/MS initiated traffic in mind. Network initiated traffic is becoming more and more of interest, especially with machine-to-machine communication. Network initiated traffic introduces paging, which we have shown in chapter 3.6 is reasonable to assume is the reason for packet loss.

5 Conclusions and further work

5.1 Conclusion

In this thesis, we have looked at the performance indicators at the link layer and up to the application layer for some selected applications. One indicator that strongly affects the performance of the wireless network is the high latency. A much used application protocol as HTTP is highly affected by the high latency, due to TCP slow start.

By looking into technical specifications and experimenting with AT-commands we have found out that a lot of the commands specified are optional to implement, and that this can be a problem when developing a tool that executes AT-commands. Another problem with AT-commands is that they mostly are requests. One can request a certain QoS, but one cannot verify which QoS class one is assigned.

We have discussed what an end-to-end test application should include and why. A part of the assignment was to indicate those KPIs that affect the user experience of different services in GPRS and UMTS networks. We have done this for Ping, HTTP, FTP and UDP streaming. For UDP streaming we have suggested to use the Iperf tool as a way of simulating multimedia streams.

It is also clear that some of the parameters that indicate the performance are not possible, or hard to detect. Terminal capabilities can be obtained by looking into the specifications of the terminal, while operator setup is partly unknown. The data link layer is affected by the radio parameters, coding scheme and timeslot capabilities. Traffic load can only be assumed from the time of day the measurement is performed. Coding scheme used in GPRS and EDGE can only be estimated by looking at the actual traffic transmitted and received. The coding scheme is important in deciding the available throughput and why the performance is as it is. This is partly related to the great limitation of an end-to-end tool. One can conclude that the network has problems, but not why, with an end-to-end tool. On the other side, an end-to-end tool is excellent for concluding that a network actually performs well and is healthy.

5.2 Further work

Although we have touched into the area of cell reselection time theory in chapter 4.5, all our measurements have been done static. Mobility greatly affects the performance. Outage time will greatly affect TCP connections and slow start, and it is reasonable to believe packet loss will be a problem. Mobility introduces a lot of new parameters, like speed, distance from base station etc. Measurements regarding cell reselection, i.e. outage time would be an interesting feature to look further into.

Although we have evaluated UDP streaming, the actual implementation in the end-to-end test tool is remaining. This can be done either with Iperf or by developing and implementing our own UDP stream tool.

References

[1] Wikipedia, “Global System for Mobile communications”, February 2006, http://en.wikipedia.org/wiki/GSM

[2] J. Schiller, *Mobile Communications*. England:Addison-Wesley, 2003.

[3] Wikipedia, “Base Station Subsystem”, February 2006, http://en.wikipedia.org/wiki/Base_Station_Subsystem

[4] Wikipedia, “Network switching subsystem”, February 2006, http://en.wikipedia.org/wiki/Network_Switching_Subsystem

[5] UMTS World, “UMTS overview”, July 2002, http://www.umtsworld.com/technology/overview.htm

[6] Wikipedia, “Universal Mobile Telecommunications System”, February 2006, http://en.wikipedia.org/wiki/Umts

[7] Ericsson Radio Systems, “Basic Concepts of WCDMA Radio Access Network”, 2002, http://www.ericsson.com/technology/whitepapers/e207_whitepaper_ny_k1.pdf

[8] Wikipedia, “GPRS Core Network”, February 2006, http://en.wikipedia.org/wiki/GPRS_Core_Network

[9] 3G Partnership Project, “3GPP TS 23.060 v4.3.0, GPRS Service Description ”, January 2001, http://www.3gpp.org/ftp/Specs/archive/23_series/23.060/23060-430.zip

[10] D. Wisely, P. Eardley, L. Burness, *IP for 3G*. England: Wiley, 2002.

[11] Wikipedia, “Network Address Translation”, February 2006, http://en.wikipedia.org/wiki/Network_address_translation .

[12] Wikipedia, “Firewall”, February 2006, http://en.wikipedia.org/wiki/Firewall_(networking) .

[13] Wikipedia, “IP v4”, February 2006, http://en.wikipedia.org/wiki/IPv4 .

[14] Wikipedia, “IP v6”, February 2006, http://en.wikipedia.org/wiki/IPv6 .

[15] Wikipedia, “Enhanced Data Rates for GSM Evolution”, February 2006, http://en.wikipedia.org/wiki/Enhanced_Data_Rates_for_GSM_Evolution

[16] Ericsson AB , “Edge white paper”, 2003, http://www.ericsson.com/technology/whitepapers/edge_wp_technical.pdf

[17] Wikipedia, “Hayes command set”, February 2006, http://en.wikipedia.org/wiki/AT_Commands

[18] 3G Partnership Project, “3GPP TS 07.07 v7.8.0, AT command set for GSM Mobile Equipment”, March 2003, http://www.3gpp.org/ftp/Specs/archive/07_series/07.07/0707-780.zip

[19] 3G Partnership Project, “3GPP TS 27.007 v6.8.0, AT command set for User Equipment”, March 2005, http://www.3gpp.org/ftp/Specs/archive/27_series/27.007/27007-680.zip

[20] Wikipedia, “Transmission Control Protocol”, February 2006, http://en.wikipedia.org/wiki/Transmission_control_protocol

[21] Wikipedia, “User Datagram Protocol“, February 2006, http://en.wikipedia.org/wiki/User_Datagram_Protocol

[22] J. Postel, “User Datagram Protocol”, August 1980, http://www.faqs.org/rfcs/rfc768.html

[23] Wikipedia, “File Transfer Protocol”, February 2006, http://en.wikipedia.org/wiki/Ftp

[24] Wikipedia, “Hypertext Transfer Protocol”, February 2006, http://en.wikipedia.org/wiki/http

[25] M.Meyer, “TCP Performance over GPRS”, 1999, http://www.cs.helsinki.fi/u/gurtov/reiner/wcnc99.pdf

[26] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, I. Pratt, “Performance Optimizations for Wireless Wide-area Networks: Comparative Study and Experimental Evaluation”, 2004.

[27] P. Benko, G. Malicsko, A. Veres, “A Large-scale, Passive Analysis of End-to-End TCP Performance over GPRS“, 2004, http://www.ieee-infocom.org/2004/Papers/39_2.PDF

[28] Wikipedia, “Opera Mini”, May 2006, http://en.wikipedia.org/wiki/Opera_Mini

[29] G. Gómez et al, *End-to-end Quality of Service over Cellular Networks*. England: Wiley, 2005.

[30] R.Jain, *The art of computer systems performance analysis, Techniques for Experimental Design, Measurement, Simulation and Modeling*, Wiley, 1991.

[31] R. Sánchez, M. Martínez, S. Hierrezuelo, J. Guerrero and J. Torreblanca, “Service Performance Verification and Benchmarking” in *End-to-end Quality of Service over Cellular Networks*. England: Wiley, 2005, pp. 186-242.

[32] PSINet Technical Library, “Networking Basics: Differences between Architectures”, 2000, http://www.support.psi.net/support/common/networking/diff.html

[33] R. Sánchez, G. Gómez, P. Ameigeiras, J. Navarro and G. Ramos, “End-to-end Service Performance Analysis” in *End-to-end Quality of Service over Cellular Networks*. England: Wiley, 2005, pp. 139-185

[34] K. Premkumar and A. Chockalingam, “Performance Analysis of RLC/MAC and LLC Layers in a GPRS Protocol Stack”, *IEEE Transactions on Vehicular Technology*, vol 53, NO. 5, September 2004.

[35] R. Ludwig, A. Konrad, A. D. Joseph, R. H. Katz, “Optimizing the End-to-End Performance of Reliable Flows over Wireless Links,” *Wireless Networks*, vol. 8, NO. 2-3, March-May 2002.

[36] Speedguide.net, “What is the bandwidth delay product”, March 2006. http://www.speedguide.net/faq_in_q.php?qid=185

[37] R. Chakravorty, J. Cartwright and I. Pratt, “Practical Experience with TCP over GPRS”, submitted to IEEE GLOBECOM 2002, Taipei, Taiwan, 2002.

[38] P. Hakalin, P. Tapia, J. Ramiro-Moreno, R. Rodríguez, M. C. Aguayo-Torres and R. Sánchez, “Cellular Wireless Technologies,” in *End-to-end Quality of Service over Cellular Networks*. England: Wiley, 2005, pp. 13-49.

[39] Telenor Mobil, “Mer om UMTS og EDGE,” March 2006, http://telenormobil.no/tjenester/3g/merom.do

[40] Mono Project, 2006, http://www.mono-project.com/Main_Page

APPENDIX A: Installation of Iperf and Jperf

Installation of Iperf

Download the binary executable of version 1.7.0 from here http://dast.nlanr.net/Projects/Iperf/#download . No installation is necessary; just uncompress it using tar xvzf filename.tar.gz or WinRAR or another archiving tool capable of uncompressing tar.gz files. A newer version is available in source only (version 2.0.2). The newest version is also available in Debians (GNU/Linux) package system. To install, simply type apt-get install iperf. Be aware that there have been some reports of incompatibility between version 2.0.2 and 1.7.0. Probably a good choice would be to run the same version in both ends.

Firewall configuration

The host running the instance of Iperf in server mode must accept incoming connections on port 5001. To perform UDP tests it must allow incoming UDP datagrams, and for TCP testing it must allow incoming TCP connections. Port 5001 is the default port, but it can be changed with the –p parameter. If the server node is behind NAT, forwarding must be configured in the router. The easiest way to run Iperf is by running both the client and the server on hosts with public IP’s.

Installation of Jperf

1. Make sure you have SUN Java Runtime Environment (JRE) installed. If not, download it from http://java.com/en/download/index.jsp. Jperf worked with Java Runtime Environment Version 5.0 Update 6.
2. Download the Jperf binary from http://dast.nlanr.net/projects/jperf/jperf-1.0.tar.gz
3. Untar it with tar xvzf jperf-1.0.tar.gz in Linux, or use i.e. WinRAR to uncompress it in Windows.
4. Copy the three jar files into the \lib\ext directory of your JRE installation. In our case that was C:\Program Files\Java\jre1.5.0_06\lib\ext
5. Make sure Iperf is in your path. The easiest way to ensure this in Windows is to put Iperf.exe in the \Windows\System32 directory. In Linux, put it in on of the bin directories. I.e. /bin , /usr/bin or /usr/local/bin .
6. Open a console window and type java Jperf. (Notice the uppercase J). If you get the response below, Iperf is not in your path.

```
C:\>java Jperf
CreateProcess: iperf -v error=2. (Iperf is probably not in your path.)
```

APPENDIX B: List of Abbreviations

2.5G	2nd and a half generation mobile telephone system (see GPRS)
2.75G	2nd and three quarters generation mobile telephone system (unofficial, see EDGE)
2G	2nd Generation mobile telephone system (see GSM)
3G	3rd Generation mobile telephone system (see UMTS)
ACK	Acknowledgement
API	Application Programming Interface
APN	Access Point Name
ARQ	Automatic Repeat Request
AT commands	Modem initialization command (Attention)
AuC	Authentication Center
BCCH	Broadcast Common Control Channel
BDP	Bandwidth Delay Product
BER	Bit Error Rate
BLER	Block Error Rate
BO	Buffer Occupancy
BSC	Base Station Controller
BSS	Base Station Subsystem
BTS	Base Station Transceiver
CDMA	Code Division Multiple Access
CDR	Call Data Records
COM	Communication
CPCH	Common Packet Channel
CS	Circuit Switched
DCH	Dedicated Channel
DNS	Domain Name System
DSCH	Downlink Shared Channels
EDGE	Enhanced Data rates for GSM Evolution
EFL	Effective Frequency Load
E-GPRS	(see EDGE)
EIR	Equipment Identity Register
FACH	Forward Access Channel
FDD	Frequency Divided Duplex
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GMM	GPRS Mobility Management.
GMSK	Gaussian Minimum-Shift Keying
GNU	GNU's Not Unix
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HLR	Home Location Register
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IDE	Integrated development environment
IMSI	International Mobile Subscriber Information
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6

ISDN	Integrated Services Digital Network
kbps	Kilobits per second
KPI	Key Performance Indicators
LA	Location Area
LAC	Location Area
LLC	Logical Link Control
MAC	Media Access Control
MCS	Modulation and Coding Scheme
ME	Mobile Equipment
MMS	Multimedia Messaging Service
MS	Mobile station
MSC	Mobile services Switching Center
MSISDN	Mobile Subscriber ISDN
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NACC	Network Assisted Cell Change
NAT	Network Address Translation
NCCR	Network Controlled Cell Reselection
NSAPI	Network Layer Service Access Point Identifier
NSS	Network and Switching Subsystem
OMC	Operations and Maintenance Center
OSS	Operation Subsystem
PCCCH	Packet Common Control Channel
PCU	Packet Control Unit
PDP	Packet Data Protocol
PS	Packet Switched
PSK	Phase Shift Keying
QoS	Quality of Service
RAB	Radio Access Bearer
RACH	Random Access Channel
RAN	Radion Access Network
RF	Reduction Factor
RLC	Radio Link Control
RRC	Radio Resource Control
RRM	Radio Resource Management
RSS	Radio Subsystem
RSSI	Received Signal Strength Indicator
RTO	Retransmission Timeout value
RTP	Realtime Transport Protocol
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SACK	Selective Acknowledge
SF	Spreading Factor
SGSN	Serving GPRS Support Node
SIP	Session Initiation Protocol
SMS	Short Message Service
SS7	Signalling System no. 7
TA	Terminal Adaptor
TBF	Temporary Block Flow
TCP	Transmission Control Protocol
TDMA	Time Divided Multiple Access
TE	Terminal Equipment

TSL	Timeslot
TTL	Time To Live
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTRAN	UMTS Terrestrial Radio Access Network
VLR	Visitor Location Register
VoIP	Voice over IP
WAP	Wireless Application Protocol
WCDMA	Wideband Code Divided Multiple Access