USB Attached Network Performance

Uplink Performance

SHUANG DI

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2009

TRITA-ICT-EX-2009:2

USB Attached Network Performance: Uplink Performance

Shuang Di

2009-4-2

Submitted in partial fulfillment of the requirements for the degree of
Master of Science (Information Technology)

Supervisor & Examiner
Professor Gerald Q. Maguire Jr.

Department of Communication Systems
School of Information and Communication Technology
Royal Institute of Technology
Stockholm, Sweden

Abstract

More and more people all over the world are using a USB modem to connect to the Internet. This is especially true in Sweden- which has the highest 3G coverage in Europe. Comparable to fixed broadband access, using a 3G USB modem is a wonderful experience for the customer. Increasingly, anywhere we can use our cellular phone is a place that we are able to access the Internet, even if we are traveling in a train, walking in a park, or sitting on the beach. This makes working on the move much easier. However, customers do not know if the network throughput, delay, and the cost of connectivity match their needs. Additionally, there is also the question of the time to connect to the network when there has not been some traffic for a period of time. However, the throughput, delay, and "time to connect" when using such a modem to connect to a wireless network depend on several factors. In this thesis we will examine these factors. More specifically, the thesis will analyze in detail the effects of using a USB attached network interface, in order to gain greater understanding of such an interface’s network performance. We have chosen to focus on the uplink, as it was expected to have a low throughput and it has not had as much attention as the downlink performance.

Key words: USB modem, 3G, network performance

Sammanfattning

Fler och fler människor över hela världen använder ett USB-modem för att ansluta till Internet. Detta gäller i synnerhet i Sverige, som har den största 3G-täckning i Europa. Jämförbar med fast bredband, med hjälp av ett 3G USB-modemet är en underbar upplevelse för kunden. Allt var som helst vi kan använda vår mobiltelefon är en plats som vi har tillgång till Internet, även om vi färdas i ett tåg, promenad i en park, eller sitter på stranden. Detta gör att arbeta på resande fot mycket enklare. Men kunderna inte vet om nätet genomströmning, fördröjning, och kostnaden för anslutning passar deras behov. Dessutom finns det också en fråga om tid att ansluta till nätverket när det inte har funnits några trafik för en tid. Men den genomströmning, fördröjning och "tid för att ansluta" när du använder ett modem för att ansluta till ett trådlöst nätverk beror på flera faktorer. I denna avhandling kommer vi att undersöka dessa faktorer. Mer specifikt avhandlingen analyserar i detalj effekterna av att använda en USB bifogade nätverkskort, för att få större förståelse för ett sådant gränssnitt nätverk prestanda. Vi har valt att fokusera på upplänk, eftersom det förväntas ha låg kapacitet och det har inte fått lika mycket uppmärksamhet som de nerlänkning prestanda.

Nyckelord: USB modem, 3G, network performance

Acknowledgements

I would like to express my most sincere gratitude to Professor Gerald Q. Maguire Jr. for his strong support all the time and numerous bits of valuable advice concerning this thesis. He is not only a knowledgeable professor who has deep vision within his academic area, but also a great person with honest, diligent attitude and is always kind when helping his students.

Also I would like to thank Tele2's press relations - Annika Kristersson for the granting permission to include their figures in this thesis.

Thanks to my parents and my friends, for their support and encouragement at all times. I could not have finished my studies and live a lovely life without them.

Table of Contents

List of Figures

List of Tables

List of Acronyms and abbreviations

3G	Third Generation
AC/DC	Alternating Current/Direct Current
ACK	Acknowledgement
AT&T	American Telephone and Telegraph
ATM	Asynchronous Time Multiplexing
B/s	Byte per second
CDMA	Code Division Multiple Access
DC	Direct Current
DSL	Digital Subscriber Loop
DWDM	Dense Wavelength Division Multiplexing
EDR	Enhanced Data Rate
GB	Gigabyte
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSA	Global mobile Suppliers Association
GSM	Global System for Mobile communications
GUI	Graphical User Interface
Gbps	Gigabit per second
HSDPA	High Speed Downlink Packet Access
HSPA	High Speed Packet Access
HSUPA	High Speed Uplink Packet Access
HTTP	Hypertext transfer protocol
I/O	Input/Output
ICT	Information and Communication Technologies
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMT-2000	International Mobile Telecommunications-2000
IRPs	Integrated Resource Packages
ISP	Internet Service Provider
KB/s	Kilobyte per second
Kbit/s	Kilobit per second
LAN	Local Area Network
MAC	Media Access Control
MB/s	Megabyte per second
MDL	Memory Descriptor List
MN	Mobile Node
MSDN	Microsoft Developers Network
Mb/s	Megabyte per second
Mbps	Megabits per second
NAK	Negative Acknowledgement
NTP	Network Time Protocol

NYET	No response YET
OECD	Organization for Economic Co-operation and Development
OS	Operating System
OSP	Online service provider
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PID	Packet Identifier
PPP	Point-to-point Protocol
QoE	Quality of Experience
RFC	Request for Comments
RTCP	Real-time Control Protocol
SDU	Service Data Unit
SLAs	Service Level Agreements
SLIP	Serial Line IP
SOF	Start of Frame
UMTS	Universal Mobile Telecommunications System
URB	USB Request Block
USB	Universal Serial Bus
USIM	UMTS Subscriber Identification Module
WAN	Wide Area Network
WCDMA	Wideband Code Division Multiple Access
Windows CE	Windows Embedded Compact Edition
WLAN	Wireless Local Area Network

1. Introduction

This chapter introduces the core issues considered in this thesis. It also provides a clear description of the objectives and a statement of the problems that the thesis will focus on.

1.1 Problem statement

In order to conveniently access the internet, many people in Sweden are using USB modems. However, does the network throughput which advertisements suggest have any relationship to the real throughput which a user might experience? Customers pay a fixed fee for the services each month, but are the services actually worth that price? To answer these questions, an analysis of USB attached 3G network performance needs to be conducted.

Before we consider the measurements and analysis which are to be done, we first give an introduction in the following sections – to describe some useful concepts, issues, and background information about a USB attached 3G network interface.

1.2 USB modem and USB attached 3G network interface

Using USB modems is now quite popular in Stockholm. Open any local Swedish newspaper and you will see an advertisement about USB attached network service (see for example Figure 1). (Note that the specific service used during this project was a fix price ("fast pris") subscription at 199 kronor per month from Tele2.)

Figure 1: TELE2 USB-modem advertisements [1]

(Appears with permission of Tele2 press relations: Annika Kristersson)

Before considering USB modems, we first must answer the question: What is a modem? A modem (modulator/demodulator) is a device that modulates an analog carrier signal to encode digital information and demodulates such a carrier signal to decode the transmitted

information. The goal is to produce a signal that can be transmitted easily and decoded to reproduce the original digital data. Modems can be used over any means of transmitting analog signals, from light emitting diodes to radio. [2] There have been many different kinds of modems, including narrowband dial-up modems, broadband modems, and wireless modems.

A USB modem would be a modem which is attached to the universal serial bus (USB). However, being strict - the devices which are commonly called 3G USB modems, are not actually modems at all since they take digital information and format it into frames and send it across a digital network - the device is not actually modulating a carrier wave, but leaves that task to the underlying 3G radio interface! However, in keeping with the popular terminology - we will refer to them as USB modems throughout this thesis. As we will see these devices mix the characteristics of a network interface with a serial interface to a traditional modem, hence it will be important to understand just how these concepts are mixed.

Figure 2: USB modem (top) and the modem as attached to a laptop computer (bottom)

The latest 3G USB modems have an integrated UMTS Subscriber Identification Module (USIM) reader. The USIM contains the subscription information necessary to access the 3G network. They provide wireless access to the Internet using High Speed Downlink Packet Access (HSDPA) capable UMTS networks. Typical maximum download speeds up to 7.2

Mbps and upload speeds 384 kbps are quoted to customers, suggesting that the customer will have wireless connectivity that is similar in performance to access via digital subscriber line (DSL) or cable modems. The suggestion (of the advertisements) is that by using this device, customers are able to access the Internet anywhere and everywhere and use their notebook computers much as they would if connected to a fixed broadband network at home or in their office.

Today most of these USB modems connect to the computer through a USB 2.0 (Universal Serial Bus version 2.0) interface, which supports maximum data rates of up 480 Mbps and is convenient (as nearly all computers have one more USB 2.0 interfaces). Another advantage of USB is that it provides DC power; hence the modem gets its DC power directly from the USB interface, avoiding the need for a separate battery or external AC/DC adapter; which contributes to the convenience of such devices.

Thus to use such a USB modem the user simply plugs the device into any USB host equipped notebook or desktop computer and they are now able to connect to the Internet via a GSM/UMTS network. Unlike PCMCIA (short for Personal Computer Memory Card International Association, an international organization that defines and promotes the PC Card standards [3][4]) modems, a USB Modem can be connected not only to notebook computers, but to any computer having USB host support (which today is nearly universal).

We will focus in this thesis on customers connecting to the internet using the USB modem as their network interface. Because this interface uses the wide area cellular network as its access network, customers have to pay a telecommunications operator to use this network. Figure 3 is an example of an advertisement for this so called "Mobile broadband" ("Mobilt Bredband" in Swedish). This advertisement illustrates the service price of using a USB attached network interface in Stockholm, using TELE2, one of the biggest cellular telecommunication operators in Sweden.

Figure 3: A TELE2 3G USB modem advertisement [5]
(Appears with permission of Tele2 press relations: Annika Kristersson)

In the green-colored text on the right hand side of the picture, we see that it costs 99 Swedish kronor each month for accessing the network with a speed of up to 0.384M bit/s

(Mbps), while it costs 189 Swedish kronor every month with a speed of up to 7.2M bit/s. Compared to the broadband price (from this same operator to my home in Stockholm) of 156 kronor per month, this price is not so different -- and the customer gains the advantage of mobile access to the internet - rather than only having access from their home (or office).

1.2.1 Why do people use such a modem?

There has to be some reason that USB modems have become so popular. Vodaphone states in a recent advertisement some benefits of using a USB modem: [6]

- Boost your efficiency with 14 times faster download speeds than standard 3G.
- Work more efficiently with 22-times faster upload speeds than standard 3G
- Plug and go with any laptop or desktop, including Macs.
- Enjoy seamless coverage at home and roaming abroad.
- Connect with speeds of up to 7.2 Mbps.

The most attractive advantage is that working on the move is easier. Unlike fixed broadband service, users are no longer tied to an Internet port in the corner of their room, attached by an inconvenient long wire. They can go freely where they want without any restriction. While one might counter that the user could use a wireless local area network (WLAN) - such as WiFi, to avoid this wire - this limits the user's mobility (via a single WLAN access point) to ~100m. Additionally, while there are increasing numbers of WLAN access points which are available to user, there are two main problems: (1) there is no unified way to (legally) utilize them (as the user has to deal with a number of subscriptions, exchange agreements, etc.) and: (2) the coverage area is still rather limited in comparison to the major 3G networks. Thus finding a usable WiFi access point is impossible sometimes. (For measurements of WLAN coverage in Kista, see the thesis of Qiang Fu [7].) However, when we use a 3G USB modem our laptop could access the network even if we are traveling in a car, walking in a park, or sitting on the beach. Or at least that is the impression which the 3G operators would like their customers to have. Therefore one of the activities of this thesis is to determine if this is true or not.

Besides mobility, the newest 3G USB modems are advertised to connect with speeds of up to 7.2 Mbps. This speed would seem to meet most customers' requirements. As with this speed they would expect to be able to browse on-line newspapers, download pictures, check their e-mail (and downloading even rather large attachments), sell and buy stocks, ... , with little delay. However, again the question remains of if this is the case in practice or not. Hence measurements will be conducted to understand just how much throughput the user can expect to get in practice.

1.2.2 Why do we need performance analysis?

The most important question that customers ask is if the real network speed which they can achieve is the announced “7.2 Mbps” or not. Users today have learned that the fine print in their fixed broadband connection agreement of "speeds up to 24 Mbps" does not actually mean that they will every actually have a throughput of 24 Mbps, only that the operator promised never to *exceed* this speed!

As we know, the speed of a wireless network depends on several factors, including the customer’s physical distance from the wireless access point (or base station), the type of building materials (walls, doors, floors) between the access point and the customer's radio transceiver, the type of wireless service, the performance of the computer, the level of network congestion, the number of other users and their current traffic demands, and many other factors. Thus all of these factors can reduce the actual network speed to below the maximum theoretical value, here say 7.2 Mbps.

Thus there is a clear question of just how large an effect these factors are in practice. Alternatively the question is how often the user gets the performance which they expect. In this thesis we will attempt to address these and other questions - with a goal of understanding in detail the performance which a user experiences and what in particular are the bottlenecks to achieving this performance above the radio layer.

1.3 Objectives

The objective is to understand the performance as experienced in practice at the network, transport, and application layers using a 3G USB modem. We have chosen to focus on the uplink, as it was expected to have a low throughput and it has not had as much attention as the downlink performance. To achieve this goal, there are some software issues which must be addressed during the analysis. Figure 4 shows (schematically) the configuration which will be used for the tests. In this test a laptop computer will be connected to a Huawei E220 HSDPA USB modem and will communicate over Tele2's 3G network. This wide area cellular network is connected via a Gateway GPRS Support Node (GGSN) to the internet. The GGSN is responsible for the interworking between a GPRS network and external packet switched networks. We will use hosts and servers attached to the internet as corresponding nodes (and servers) to applications running on the laptop computer (which will be our mobile node (MN)).

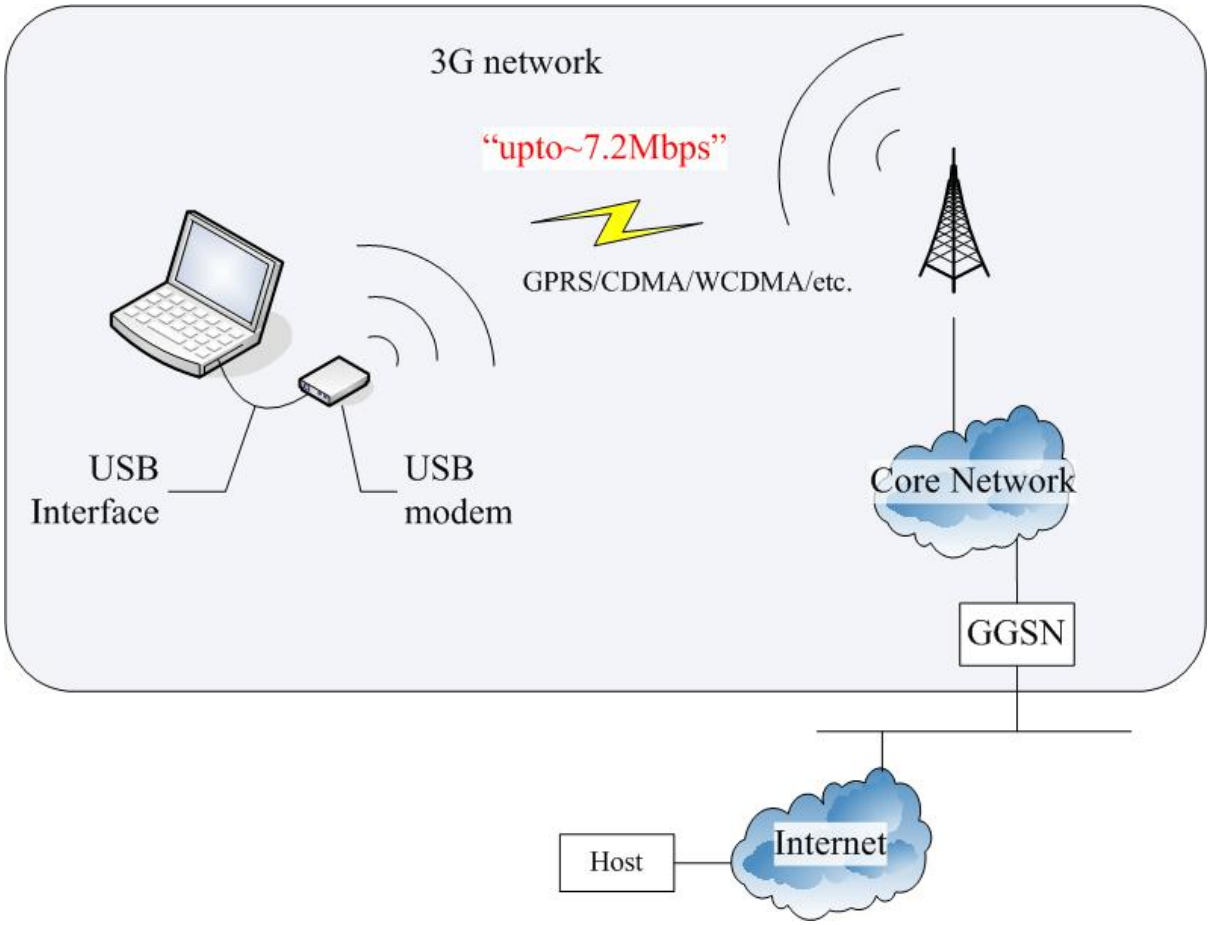


Figure 4: Using a USB attached 3G network interface to connect with the Internet

In Figure 4 the wireless network link is expected to support speeds "up to 7.2 Mbps" as advertised by the network operator. It is the performance across this link - coupled to the performance from the base station and through all of the routers to the destination - which this thesis will focus on. Hopefully it will be possible to split the analysis into two parts, but initially that is not certain.

1.4 Organization of this thesis

The first two chapters briefly describe the main issues which this thesis considers. Chapter one gave an introduction to the USB modem, USB attached network interface, and the benefits of using such a network interface. The chapter raised some questions concerning the actual network performance and identified the importance of performance analysis which will be the major objective of this thesis. The second chapter describes the background necessary for the reader for the remainder of the thesis and provides a comprehensive analysis of current Swedish wireless broadband market, examines the 3G network performance both in Sweden and else where, and examines typical methods of network performance analysis including relevant packet capture issues.

Chapter three goes more deeply into the performance analysis, examining detailed models and tools for performance analysis. Chapter four contains a description of the implementation issues. Chapter five concerns the analysis of the collected data. The final chapter will give some conclusions and suggest some future work.

2. Background

This chapter provides a background for the reader of this thesis. It presents an analysis of the Swedish and other wireless broadband markets and introduces the basic principles of performance analysis and describes how packet capture can be done for the traffic which will use the USB-modem as its network interface.

2.1 Swedish wireless broadband market

It is important to understand the current Swedish wireless broadband market to motivate why we wish to carry out a performance analysis of current wireless broadband service.

2.1.1 Swedish broadband market

Sweden has one of the highest fixed broadband penetration rates in Europe. This is the result of an effective government broadband policy and a population that is quick to adopt emerging technologies. Use of Digital Subscriber Loop (DSL) dominates broadband access, although Metropolitan Ethernet, fiber, cable, satellite, and WLAN access are also available. Considerable investment in fast mobile networks has also made mobile broadband a realistic proposition.

Figure 5 shows the growth in fixed broadband penetration rate per capita in several European counties. Sweden's broadband penetration rate has been the highest per capita penetration rate since 2004. Lead by Sweden, the broadband penetration rate of all these European countries have been continuously increasing during the past five years, from 2004 to 2008.

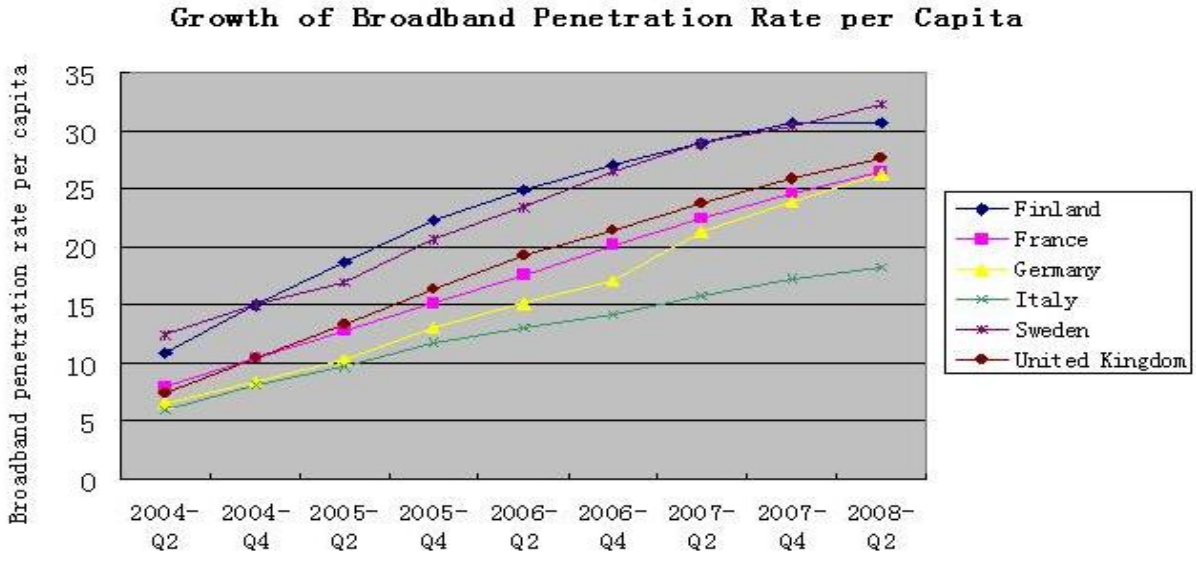


Figure 5: Growth of Broadband Penetration Rate per Capita (based on OECD data) [8]

2.1.2 Swedish wireless market

By 2005, the Swedish National Post and Telecom Agency reported that Sweden had achieved Europe's most extensive network coverage for third-generation (3G) mobile telecom services. According to their research, approximately 85 percent of Sweden's 9 million citizens were currently able to use 3G services, a figure that was expected to reach 100 percent during the next two years. Sweden also had a total of 9.3 million cell phone user accounts, which was more than its entire population. This unique combination of infrastructure, high penetration, and early adopter culture made Sweden an ideal market to launch, implement, and test the latest products, technologies, and services.

Swedish 3G services are relatively advanced. The new entrant “3” launched in 2003; and its competitors, TeliaSonera, Vodafone, and Tele2 introduced 3G services during 2004. Competition has since intensified significantly. 3G is expected to become a critical battleground between these operators. By 2009, it is expected that more than 60% of Swedish mobile phone users will be using 3G handsets. [9]

According to the IE Research report “1Q08 Mobile Forecast: Sweden, 2007 – 2010” [10], the wireless market in Sweden will expand in the total number of subscribers from 10.5 million to 11.3 million during the period 2007 - 2010. The wireless penetration level in Sweden is expected to reach 126% in 2010.

Table 2-1: Top 5 Mobile/Internet Index rankings, worldwide [11]

Economy	Mobile/Internet score (/100)	Ranking
Hong Kong, China	65.88	1
Denmark	65.61	2
Sweden	65.42	3
Switzerland	65.10	4
United States	65.04	5

Table 2-1 was extracted from the ITU (International Telecommunication Union) Mobile/Internet Index included in their report “Internet for a Mobile Generation” [11]. The index measures how each economy is performing in terms of information and communication technologies (ICT) while also capturing how poised it is to take advantage of future ICT advancements. The index covers 26 variables – which are sorted into three groups: infrastructure, usage, and market structure. These three components combine for a score with a low of 0 and a high of 100. The table is based on values from the Statistical Annex to the report, which provides comprehensive data on network and service development for over 200 economies. We can see that Sweden ranks third of the countries, which indicates that the Swedish wireless market has one of the most beneficial climates for business development of all these countries.

2.1.3 3G network performance

3G is so popular not only in Sweden, but all over the world. Research by GSA (the Global mobile Suppliers Association) that says by November 2007 there were 190 WCDMA networks in commercial service in 83 countries; with 154 HSDPA networks in commercial service in 71 countries, including 26 of the 27 Member States of the European Union. Most services benefit from HSDPA, which delivers high downlink (i.e., from the access network to the mobile terminal) data throughput capabilities and faster response times, both leading to an improved interactive experience for users. [12]

Evolution of downlink data throughput is a key trend. By November 2007, over 60% of HSDPA network operators had commercially launched or were deploying 3.6 Mbps peak downlink rate capability, including 29 networks supporting 7.2 Mbps peak. Several operators evolved their network capabilities to 14.4 Mbps during 2008. [12]

However, the actual data speed of 3G is determined based on a combination of factors including the chip rate, channel structure, power control, synchronization, customers' physical distance from the network access points, the level of network demand, etc. The International Telecommunication Union (ITU) has not provided a clear definition of the speeds users can expect from 3G equipment or providers. Thus users subscribing to 3G service can not point to a standard and say that the speeds specified in an offer are not being met. While the ITU states that "it is expected that IMT-2000 will provide higher transmission rates: a minimum speed of 2Mbps for stationary or walking users, and 348 kbps in a moving vehicle," [13]. However, the ITU does not actually clearly specify minimum or average speeds or what modes of the interfaces qualify as 3G, so various speeds are sold as 3G intended to meet customers' expectations of broadband speed. It is often suggested by industry sources that 3G users can be expected to receive 384 kbps at or below pedestrian speeds, but only 128 kbps in a moving car. [14] [15]

Figure 6, adapted from the European Information Technology Observatory in 2002, compares the range versus data rate of Bluetooth, wireless LAN, 2.5G, and 3G links. While the current networks have speeds higher than they had in 2002, the maximum speed of 3G networks has not remained at a fixed value. (Nor have Bluetooth and WLAN remained at these earlier speeds; with Bluetooth 2.0 + Enhanced Data Rate (EDR) of 3 Mbps and WLAN with pre-standard IEEE 802.11n of 100Mbps.)

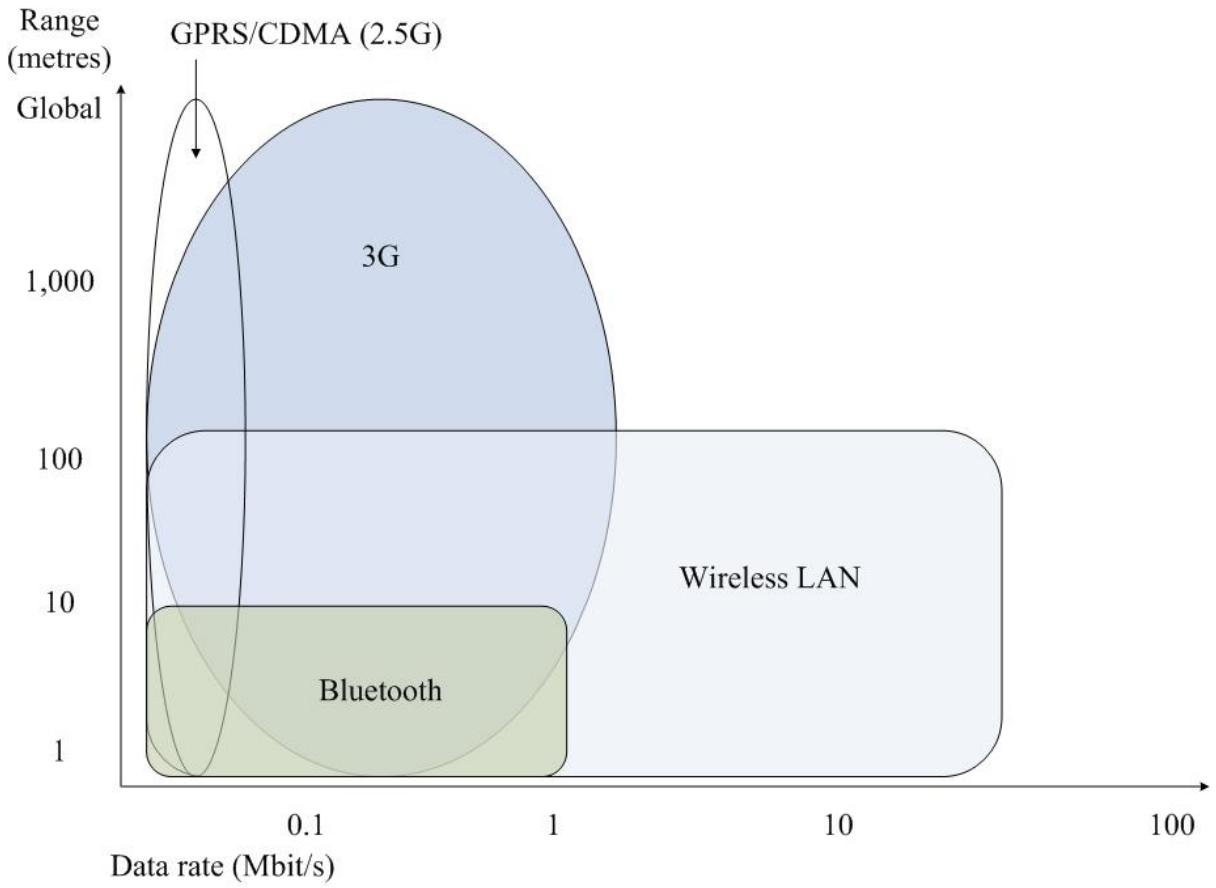


Figure 6: Range and data rate of wireless network [16]

From Figure 6, we can easily see that the data rate of 3G ranges from a very low value to a high value, such as 7.2 Mbps (today). However, this is a very wide range and customers are not guaranteed a specific network speed. The main factors which affect the 3G network speed as experienced by a user are:

- **Distance & environment:** the customers' physical distance from the network access point has a strong influence on the network performance. Additionally, the type of building materials (walls, doors, floors) between the customer's handset and the access points is another factor. For example, sometimes cellular phones have no signals in an elevator or basement because the building materials around them have attenuated most of the radio signal.
- **Modulation & coding:** network performance also depends on the chip rate, channel structure, and power control. For instance, the CDMA coding technique requires high-speed, chip-rate processing to perform the pseudorandom-number generation, channel spreading, mux/demux(two units in the DWDM(Dense Wavelength-Division Multiplexing) system), rake receiver and symbol-combining functions.
- **Load:** network load is another factor. With 3G the actual speed a customer will experience will depend on whether there is both sufficient signal strength to use a high data rate modulation and coding scheme at the user's particular location and the current

network demand by other users in the same cell who are using the service at the same time. One of the important contributions of HSDPA and HSUPA has been the ability to quickly multiplex a high speed channel between the users who are in locations where they can get high speed service. However, there is only a limited *shared* capacity in a given cell.

Sweden, although it has the highest 3G *coverage* in Europe, mainly provides high performance in or near major population centers. Figure 7 shows the coverage of one of the 3G mobile networks in Sweden (that of TeliaSonera), based upon computer calculations. Local discrepancies can occur, for example behind mountains, in deep valleys, indoors, or in cellars.

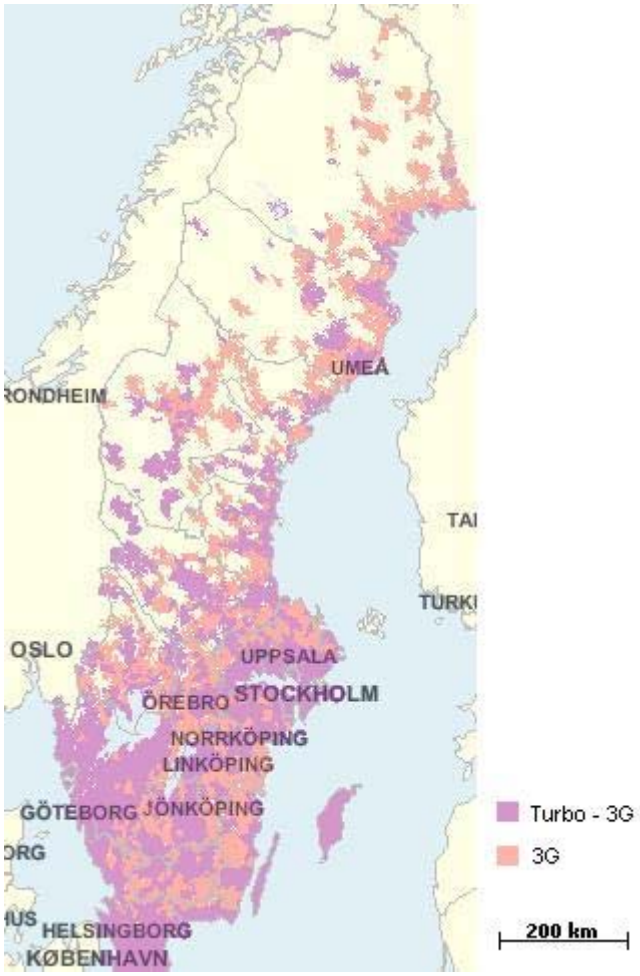


Figure 7: TeliaSonera's 3G coverage of Sweden as of 2009-3-16 [17]

The southern portion of Sweden, where the majority of the population lives, has very high mobile network coverage, including Turbo-3G and 3G. Thus customers in the south are very likely to experience better network performance than customers in the northern area, as the distance to an access point is likely to be shorter and the access points have been designed for high peak speeds (which also requires that the backhaul link from the base

station to the operator's backbone has to support higher data rates).

The Telia Trend Report 2007 indicated that more than one of out three people in Stockholm expect to be able to use wireless Internet connections in parks or other outdoor areas, followed immediately by people living in the southwestern province of Skåne, and in western Norrland, who have almost the same high expectations for wireless outdoor connections. [18]

In 2007, TeliaSonera Sweden launched tests of a new high-speed mobile broadband network. In areas of Stockholm and Gothenburg, the mobile network has been enhanced with HSPA (High-Speed Packet Access) technology, also called Turbo 3G, which gives customers wireless Internet connections that are almost ten times faster than the connection speeds of the ordinary 3G network. [19] Turbo-3G currently offers a maximum of 7.2 Mbps in those regions where it is available.

2.2 Network Performance analysis

This section discusses network performance analysis, including from which point of view the analysis is made, the measurement and analysis methods, and relevant performance parameters used in the evaluation.

2.2.1 Point of view

Although customers are exposed to plenty of news and advertisements concerning the rapidly developing mobile network and its high quality, what they actually care about is what benefits they have in their own situation. This raises two simple questions: what is the price and for a given price what is the performance. To evaluate whether the network performance is sufficient to meet a user’s expected price/performance ratio, numerous methods have been proposed; each of these focuses on different aspects of a network.

- **View of network administrators:** Viewed from above, the whole network consists of its construction, operations, maintenance, upgrade paths, etc.; the network consists of nodes (elements) and lines (links). Performance of each of these nodes and links determines the capacity and capabilities of the network.
- **View of network operation:** Rather than the point of view of network administrators who do care about the individual nodes and links; the marketing department of an operator cares about the behavior of the entire network, what kind of services might be provided to the users, and what SLAs (Service level Agreements) could be signed with customers. So far, these operators have utilized “network quality examination” systems or software to implement their analysis. It is important to note that here the concern is about *statistical* guarantees of performance, not the specific performance of individual

nodes, links, or even the experience of specific users -- but rather the statistical performance versus the target performance.

- **Quality of Service:** Theoretically the metric Quality of Service (QoS) means the network quality observed from the point of view of the users, however, “QoS” is also widely used by operators, manufacturers, and integrators for different parts of the network. Typical QoS measures include jitter, time delay, data loss rate, and so on.
- **Quality of Experience:** To focus on the users’ experience of the network, the concept Quality of Experience (QoE) was introduction and adoption of voice and video as packet data services over the internet. Quality of Experience was designed to integrate, both the external parameters and personal experience into an assessment of the quality actually delivered to users. [20] The use of QoE measures are important because the traditional QoS measurements of circuit-switched networks do not provide a good basis for evaluating the user's perceived experience of packet-switched services and locally executing applications - thus loss concealment techniques may make voice perfectly acceptable (i.e., with a Mean Opinion Score of 4 or 5) despite 5-50% packet loss.

In this thesis we will focus on the QoS and QoE points of view, in order to understand what is perceived by both the mobile device (concerning QoS) and by the user (QoE). Note that QoS measures are simpler to collect, but QoE measures are more important indications of user acceptance. We will next examine how these metrics can be measured and analyzed.

2.2.2 Measurement methods

In addition to the point of view issues, data collection modes and policies are the other two most significant factors in measurement and analysis of the network.

- **Collection modes:** nowadays we have two major methods of measurement: intrusive mode and non-intrusive mode. (1) The intrusive mode inserts testing packets to test the network, which to a certain extent artificially increases the network load. If there is network congestion, then this additional load would aggravate the congestion; on the other hand, if the network load is too low, the intrusive mode may result in an incorrect estimate of performance; intrusive mode data collection is affected by the type and length of the test (often via so called "probe" packets). (2) The non-intrusive mode utilizes passive network monitoring for data collection. This mode should give a more accurate result. For the majority of the data collection for this thesis, I have utilized the non-intrusive mode, and used network monitoring to collect data for performance analysis. Note that in order to have network traffic to monitor I will need to run applications on the mobile node that will generate and receive network traffic – hence, this traffic will suffer from some aspects of the intrusive mode. However, we assume these applications will be typical of the types of applications that users will run and will expect to run in the near future; hence, while they generate additional traffic - this traffic should be comparable to what a user might generate and receive.
- **Collection policies:** it is better to have definite policies for data collection, with a

specific measurement period, recording entries for each measurement into the log files. It is also helpful to trace data for these applications at different times during a day. However, due to the limited duration of this thesis project, there will not be a long period measurement. In practice the user connect to the network during peak hours; while they do care if the network performance can meet their demands, we will perform measurements mostly during the peak hours while the network is carrying heavy traffic loads.

2.2.3 Performance parameters

Along with collecting data, a well defined performance analysis needs to be designed to concentrate on several specific parameters. Therefore one of the important tasks before trying to collect any significant amount of data is to determine (1) what data is needed for the performance analysis and (2) how much data is needed to have the desired confidence in the results. This also implies that a decision has to be made first of the desired confidence level. For the purpose of this thesis we will use a confidence level of 95%.

Focusing on the target, the USB-attached 3G network, the following performance parameters should be considered:

- **Download speed (Peak and average):** This parameter enables a user to easily estimate how much time it will take to download files, web pages, images, and other media from the internet for users with specific kinds of connections. The peak value indicates the performance under the best condition while the average value describes the more typical performance.
- **Upload speed (Peak and average):** The upload speed is generally slower than download speed. Since people generally do more downloading, such as viewing web pages, than uploading; the download speed is more critical. Upload speed obviously becomes more important to someone who is going to do large amounts of uploading, for instance someone who works from home and wants to exchange files with a remote network.
- **Throughput:** network throughput is the average rate of successful message delivery over a communication channel. It is usually measured in bits per second (bit/s or bps), kilobits per second (kbit/s or kbps), megabits per second (Mbps), or Gigabits per second (Gbps), and sometimes in data packets per second or data packets per time slot.
- **Time delay:** Network delay is the round trip delay for a packet from end to end. In an IP network, the network delay can be measured using tools such as “ping” or protocols such as RTCP. The time delay includes the time that routers take to process the packet header, the time packet waits in router queues, the time required to transmit a packet’s bits onto the link; and the time used for the signal to propagate through the medium it is being transmitted through.

2.2.4 Packet capture issues

Packets capture is widely implemented in network analysis tools such as SnoopyPro [21], Wireshark [22], Sniffer Pro [23], SniffUsb [24], etc. Among these different programs, Wireshark is the world's foremost network protocol analyzer, and is the de facto standard across many industries and educational institutions, while Snoopy Pro is a USB sniffer application which users run on Windows machines to reverse engineer what the Windows driver for a device is doing. Snoopy Pro consists of a user mode program and filter driver which can be attached to various USB drivers to display the Integrated Resource Packages (IRPs) passing down to the lower layer drivers. One of the important aspects of the practical work in this thesis project is to ensure that Wireshark can get capture the frames sent over the USB interface to the USB modem. This will enable us to both record all of the time stamped frames, but also perform off-line analysis of this traffic.

2.2.5 Packet capture using Wireshark

Wireshark is a free packet sniffer application. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark is able to display the encapsulation and the fields along with decoding their meanings for a large variety of different networking protocols. Wireshark uses Pcap to capture packets, so it can only capture packets on the network interfaces supported by Pcap. Wireshark has a rich feature set which includes the following: [25] [26]

- Deep inspection of hundreds of protocols, with more being added all the time.
- Live capture and offline analysis.
- A packet browser.
- Multi-platform: Runs on Windows, Linux, OS X, Solaris, FreeBSD, NetBSD, and many others.
- Live data can be read from Ethernet, FDDI, PPP, token ring, IEEE 802.11, classical IP over ATM, and loop-back interfaces (at least on some platforms; not all of those types are supported on all platforms).
- The most powerful display filters in the industry.
- VoIP (Voice over IP) analysis.
- Captured network data can be browsed via a GUI, or via the terminal (command line) version of the utility (called t-shark).

By capturing packets from the interfaces (described in the above list), Wireshark collects data from the network. Figure 8 shows an example of the log file when using Wireshark to monitor a wireless network card in my laptop.

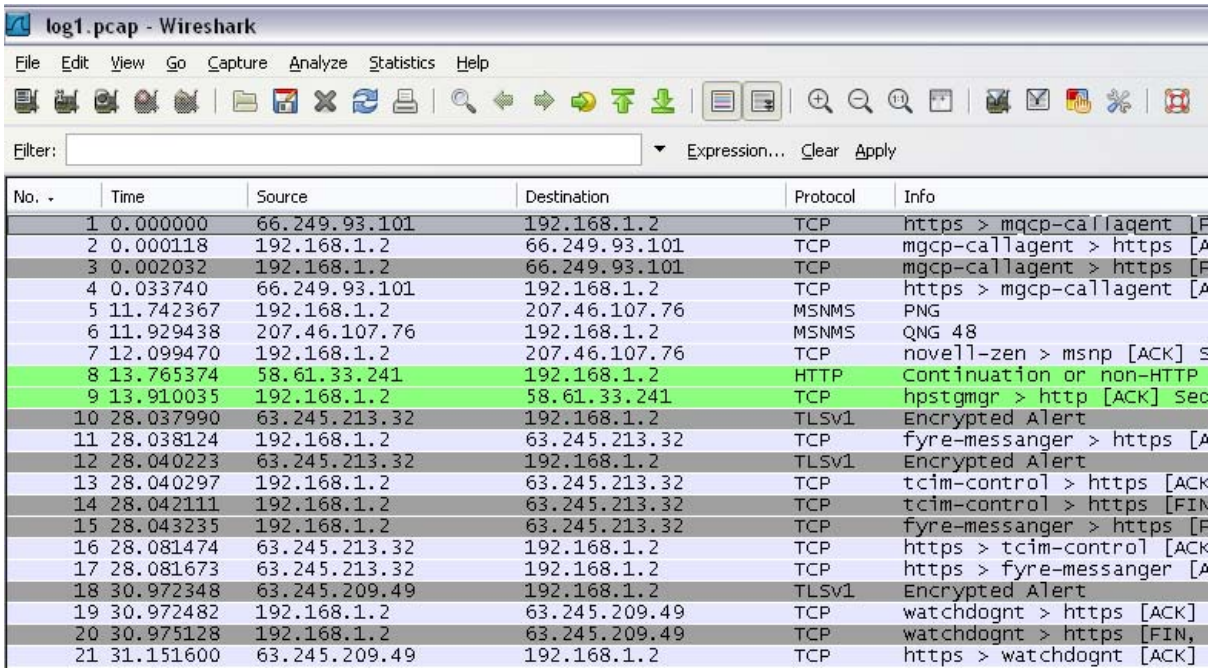


Figure 8: Wireshark log file example 1

Inside the main window, the frame number, arrival time, source address, destination address, protocol type, and additional information of each of the captured packets (passing by the wireless network card) are shown. By clicking on a specific packet, say No.8, detailed information of this packet is displayed in a minor dialog:

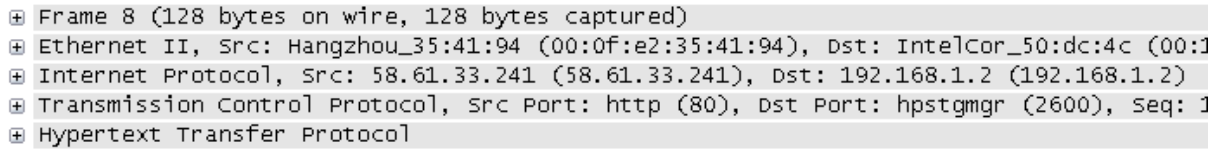


Figure 9: Wireshark log file example 2

Frame No.8 is an HTTP packet. Wireshark displays all the information of the Ethernet, IP, TCP, and HTTP headers of the packet. The Ethernet header includes the MAC address of the destination and the source, and protocol type (in this case IP) – see Figure 10.

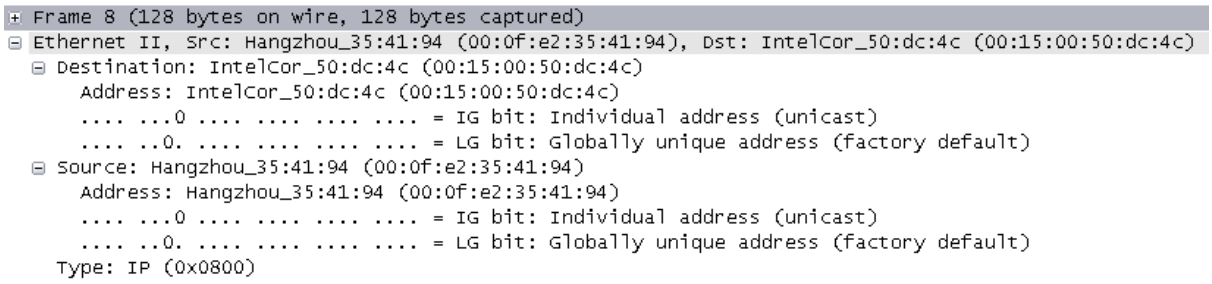


Figure 10: Wireshark log file example 3

After the Ethernet header is the Internet Protocol header, which displays the source and

the destination IP addresses, protocol version, IP header length, total length, time stamp, etc. (see Figure 11)

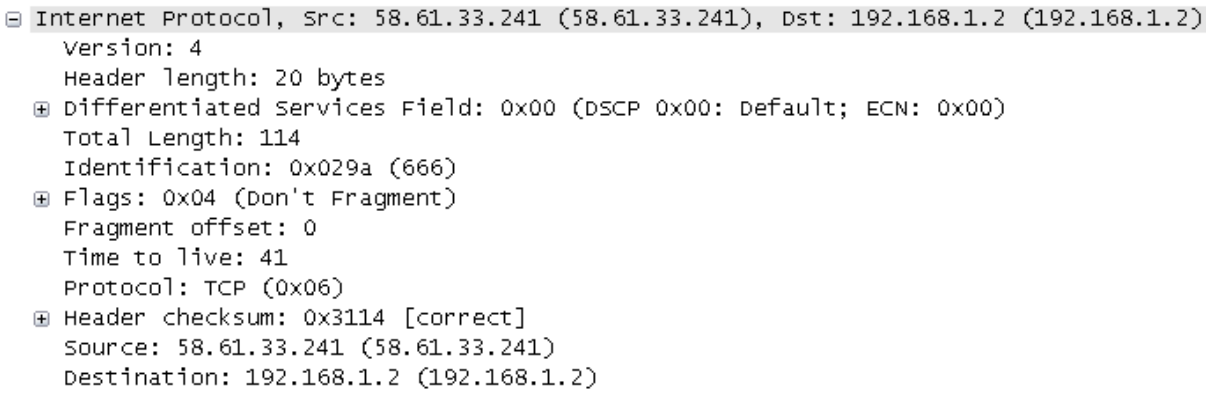


Figure 11: Wireshark log file example 4

Following the IP header is a Transmission Control Protocol header, displaying the TCP port of the source and the destination, ACK number, header length, checksum, and so on (see Figure 12). Finally there is Hypertext Transfer Protocol data.

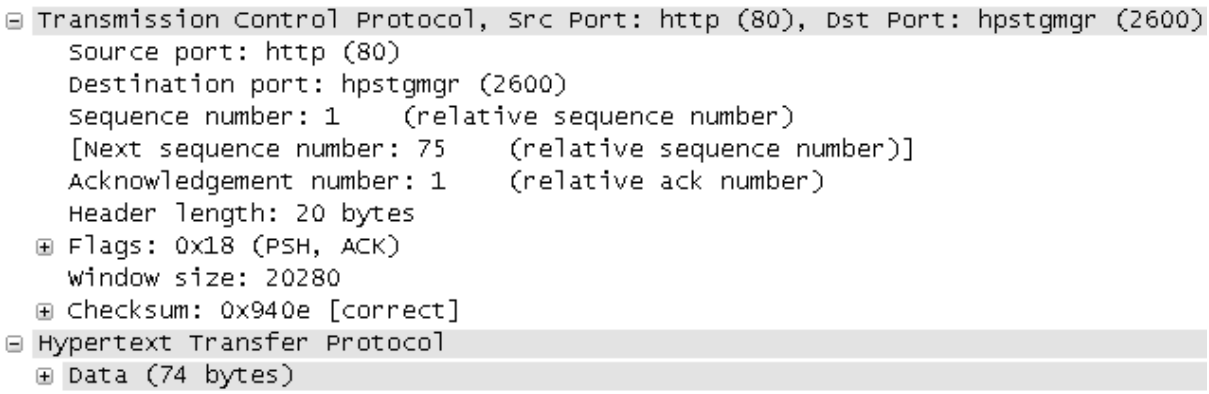


Figure 12: Wireshark log file example 5

In the case of a USB attached network, the network interface is connected to a host computer through an USB cable. The operating system "converts" the USB packets into network traffic (e.g. Ethernet packets) to provide a network interface that looks like an ordinary network interface. Thus Wireshark could capture packets from the USB device for (raw) USB traffic (if supported) or an Ethernet device for "normal" IP packets. The USB bus will add additional overhead, so the raw USB traffic will have greater volume than the Ethernet traffic.

2.2.6 USB interface monitoring

USB (Universal Serial Bus) is a serial bus standard to interface computer to various devices. The USB standard specifies relatively low tolerances for compliant USB connectors, intending to minimize incompatibilities in connectors produced by different

vendors (a goal that has been very successfully achieved).

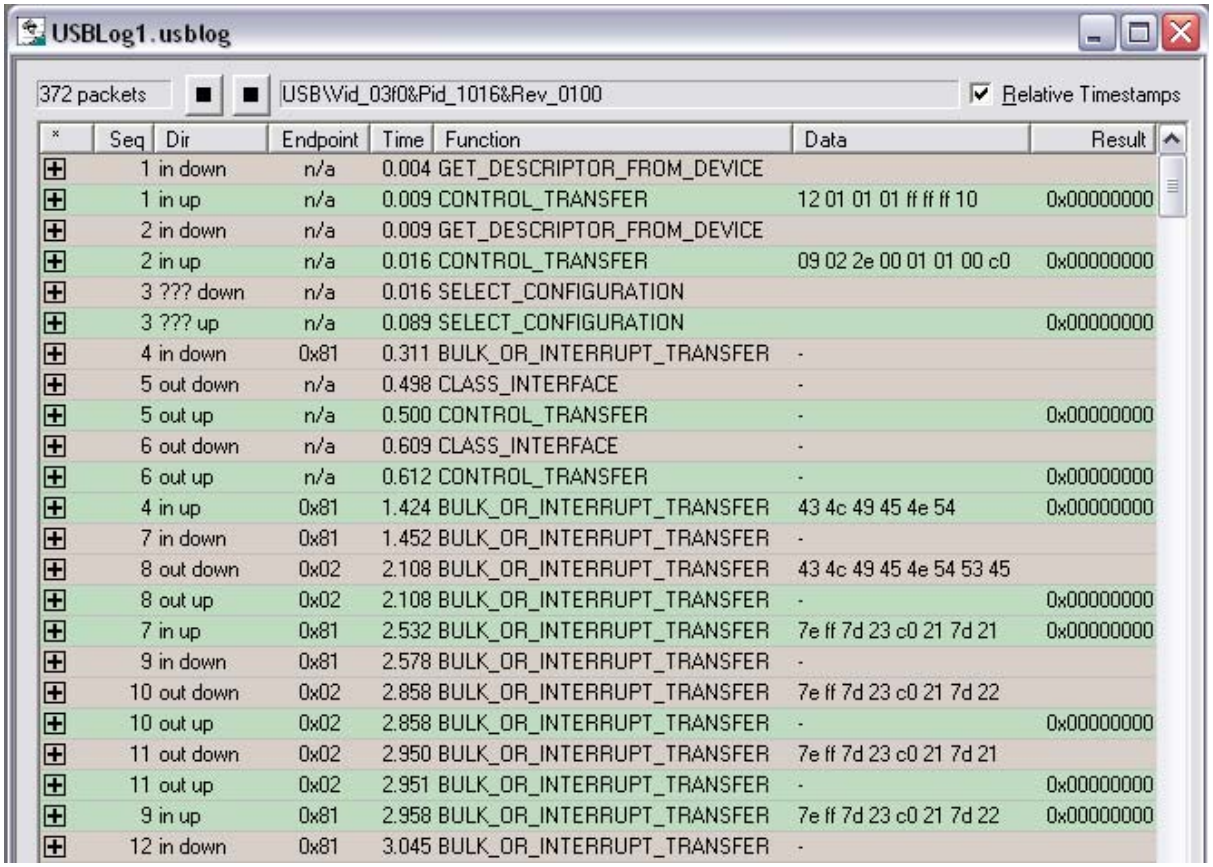
USB supports the data rates shown in Table 2-2.

Table 2-2: Data rates supported by USB [27]

Type	Specification	Rate in bps	Rate in B/s
Low Speed	1.1, 2.0	1.5 Mbps	187.5 KB/s
Full Speed	1.1, 2.0	12 Mbps	1.5 MB/s
Hi-Speed	2.0	480 Mbps	60 MB/s
Super-Speed	3.0	4.8 Gbps	600 MB/s

In the above table, the low speed is mostly used for Human Interface Devices (HID) such as keyboards, mice, and joysticks. Full speed was the fastest rate before the USB 2.0 specification and many devices fall back to Full Speed. Full Speed devices divide the USB bandwidth between them on a first-come first-served basis and it is not uncommon to run out of bandwidth with several isochronous devices (devices that support isochronous data transfers).

Focusing on a USB attached network interface, as the network packets are transmitted through the USB interfaces, capturing these packets requires capturing USB traffic. There are many kinds of USB sniffer applications. SnoopyPro is a USB port sniffer and monitor tooling with a data logger. This USB monitoring utility can capture, view, log, analyze, and test USB device activity.



*	Seq	Dir	Endpoint	Time	Function	Data	Result
⊞	1	in down	n/a	0.004	GET_DESCRIPTOR_FROM_DEVICE		
⊞	1	in up	n/a	0.009	CONTROL_TRANSFER	12 01 01 01 ff ff 10	0x00000000
⊞	2	in down	n/a	0.009	GET_DESCRIPTOR_FROM_DEVICE		
⊞	2	in up	n/a	0.016	CONTROL_TRANSFER	09 02 2e 00 01 01 00 c0	0x00000000
⊞	3	??? down	n/a	0.016	SELECT_CONFIGURATION		
⊞	3	??? up	n/a	0.089	SELECT_CONFIGURATION		0x00000000
⊞	4	in down	0x81	0.311	BULK_OR_INTERRUPT_TRANSFER	-	
⊞	5	out down	n/a	0.498	CLASS_INTERFACE	-	
⊞	5	out up	n/a	0.500	CONTROL_TRANSFER	-	0x00000000
⊞	6	out down	n/a	0.609	CLASS_INTERFACE	-	
⊞	6	out up	n/a	0.612	CONTROL_TRANSFER	-	0x00000000
⊞	4	in up	0x81	1.424	BULK_OR_INTERRUPT_TRANSFER	43 4c 49 45 4e 54	0x00000000
⊞	7	in down	0x81	1.452	BULK_OR_INTERRUPT_TRANSFER	-	
⊞	8	out down	0x02	2.108	BULK_OR_INTERRUPT_TRANSFER	43 4c 49 45 4e 54 53 45	
⊞	8	out up	0x02	2.108	BULK_OR_INTERRUPT_TRANSFER	-	0x00000000
⊞	7	in up	0x81	2.532	BULK_OR_INTERRUPT_TRANSFER	7e ff 7d 23 c0 21 7d 21	0x00000000
⊞	9	in down	0x81	2.578	BULK_OR_INTERRUPT_TRANSFER	-	
⊞	10	out down	0x02	2.858	BULK_OR_INTERRUPT_TRANSFER	7e ff 7d 23 c0 21 7d 22	
⊞	10	out up	0x02	2.858	BULK_OR_INTERRUPT_TRANSFER	-	0x00000000
⊞	11	out down	0x02	2.950	BULK_OR_INTERRUPT_TRANSFER	7e ff 7d 23 c0 21 7d 21	
⊞	11	out up	0x02	2.951	BULK_OR_INTERRUPT_TRANSFER	-	0x00000000
⊞	9	in up	0x81	2.958	BULK_OR_INTERRUPT_TRANSFER	7e ff 7d 23 c0 21 7d 22	0x00000000
⊞	12	in down	0x81	3.045	BULK_OR_INTERRUPT_TRANSFER	-	

Figure 13: SnoopyPro log file example 1

Figure 13 shows an example of a SnoopyPro log file after using this application to capture data transmitted through the USB interface between my laptop and a PDA (Personal Digital Assistant). When clicking on the forth packet we can see the detailed information and data which was transferred (see Figure 14). There information includes a URB header, header length, sequence number, function, flags, and data transfer buffer in the packet.

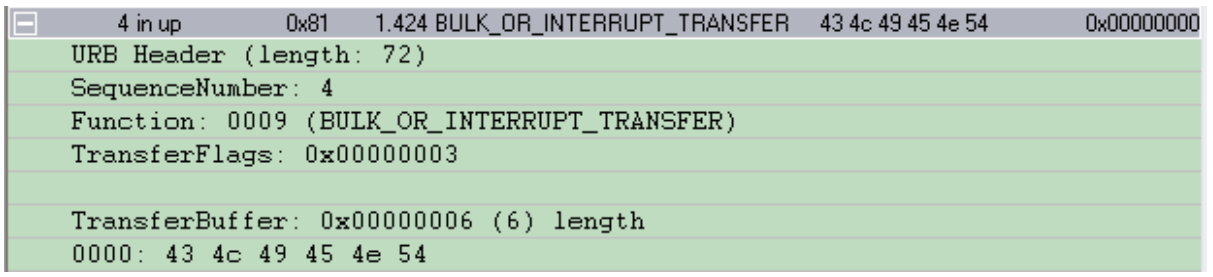


Figure 14: SnoopyPro log file example 2

Details of this traffic will be explained in section 5.1.3 , where IP traffic will be sent to/from the 3G network interface via USB.

3. Performance analysis models and tools

During preparation for collecting data, a well defined performance analysis needs to be designed. Therefore one of the important tasks is to specify our analysis models and tools. Issues to be considered include: what data is needed for the performance analysis, how much data is needed to have the desired confidence in the results, what sort of traffic is needed, and what tools should be used for the performance analysis.

3.1 What data is needed?

Focusing on the target, the USB-attached 3G network, the most important things that customers wonder about are if the network throughput and delay match their needs. Additionally, there is also the question of the time to connect to the network when there has not been some traffic per a period of time. Each of these is described more fully below.

3.1.1 Network throughput

Network throughput is the average data rate of *successful* message delivery over a communication channel (This is also referred to as goodput.). The data can be delivered over a physical or logical link, over a wireless channel, or passing through a certain network node, such as the data passed between two specific computers. [28]

While the customer cares about the “maximum throughput”, it is necessary to know there are different representations of “maximum throughput”: maximum theoretical throughput, maximum achievable throughput, peak measured throughput, and maximum sustained throughput. Maximum theoretical throughput is closely related to the channel capacity of the system, and is the maximum possible quantity of data that can be transmitted under ideal circumstances. [28] Unfortunately, this number can be deceptive, thus the “7.2 Mbps” of the advertisement will not be actually achieved. Maximum achievable throughput is typically an optimistic assumption of network performance, but provides more useful insight into the expected system performance than maximum theoretical throughput. [28] Peak measured throughput is measured by a real, implemented system, or a simulated system, over a short period of time, while maximum sustained throughput is averaged or integrated over a long time (sometimes considered to be infinite). This thesis will utilize the “peak measured throughput” as the analysis target, which is more practical and useful for the customer.

3.1.2 Time delay

The IP network delay is generally the round trip delay for an IP packet within an IP network. Network delay can be measured using tools such as “ping” or protocols such as RTCP. [29] However, in this thesis we will not talk about the round trip delay as the “time

delay" but will use this term to refer to one-way delay. This one-way time delay is defined as the period of time from when a packet is sent out until it is received by the destination node. This definition is simpler to understand as the packet sending delay. As with the IP network delay, this time delay consists of the following aspects:

- Processing delay: this is the time that routers take to process the packet header and decide where to forward the packet;
- Queuing delay: the time that packets wait in router queues;
- Transmission delay: the time to transmit the packet's bits onto the link;
- Propagation delay: the time required for the signal to propagate through the medium it is being transmitted through.

These four types of delay will help us to understand what happens during the overall time delay. However, in the data analysis we will not break down the time delay into these different parts of the delay, as these are not the focus of the research in this thesis.

3.1.3 Connect time

Here we refer the connect time to the amount of time required to connect to an online service provider (OSP) or an Internet service provider (ISP) while using a USB modem. There are two steps for a USB modem to connect to the network:

- Start the device: when we are using a USB modem, first we have to enable the USIM in the modem before connect to the networks. Each time plugging-in the USB modem, customers have to type their passwords to start the modem (also shown in the figures in section 5.1.2 . This step is important for security but inconvenient for the user. If we count this start time into the connect time, it will vary in an extremely wide range depending on the time that the customer uses to type in the passwords.
- Connect to the network: after enabling the modem, the time from when we click the "connect" button on the control panel (also shown in the figures in section 5.1.2) until it connects to the 3G/Turbo 3G/GPRS network is the real connect time. During the connect time, operations include "connecting to device", "starting transmitting packet", "authenticating the password", and finally "connect succeed". This process usually takes around 5 seconds.

3.1.4 Cumulative resource limits

Different telecommunication operators have placed different limitations on the resources used by a customer. As I am using Tele2's service, they have announced on their website that:
"Tele2 förbehåller sig rätten att begränsa överföringshastigheten till 30 Kbit/s om

användandet inom en kalendermånad överstiger 1 GB respektive 5 GB/mån (beroende på vilket abonnemang du har). Vid nästa månadsskifte får du automatiskt full hastighet igen."[30]

This announcement means: 'Tele2 reserves the right to limit the transfer speed to 30 Kbit/s when the usage within a calendar month exceeds 1 GB or 5 GB (depending on which subscription you have). The next month, you will automatically get the full speed again.' Therefore Tele2 does not the limit the total connect time, but does limit the amount of data sent. The customer must select a suitable type of service according to their preferences.

3.2 How much data is needed?

One of the important tasks before trying to collect any significant amount of data is to determine how much data is needed to have the desired confidence in the results. This also implies that a decision has to be made first of the desired confidence level. For the purpose of this thesis we will use a confidence level of 95%.

3.2.1 Confidence interval and confidence level

Before talking about confidence level, another concept "confidence interval" needs to be introduced. The confidence interval is the plus-or-minus figure usually reported in newspaper or television opinion poll results. The "plus-or-minus" indicates margin of error, which is used along with confidence interval. Figure 15 shows an example: if we use a margin of error of ±5% and 50% percent of the samples are "true", we can say that if we test the total quantity of items, we will observe that between 45% (50-5) and 55% (50+5) will be "true", that is, the confidence interval is 45% to 55%.

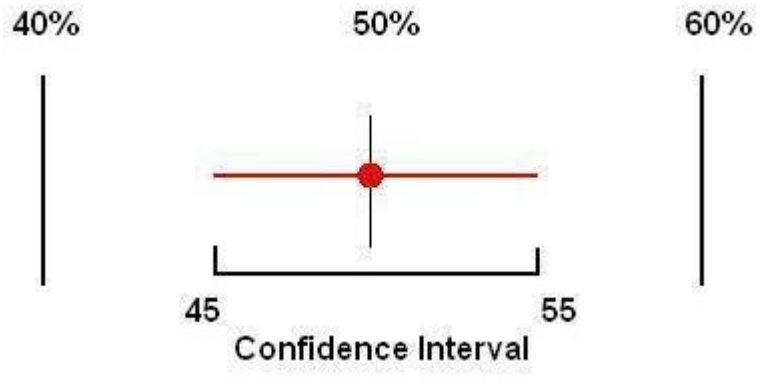


Figure 15: Confidence interval example of ±5%

The confidence level indicates how sure of the result we can be. It is expressed as a percentage and represents how often the true percentage will fall within the confidence interval. A 95% confidence level means we can be 95% certain, that is, 95 out of 100 times

the sample values will fall within the confidence intervals, or 5 times out of 100 the percentages will *not* fall within the confidence interval; while a 99% confidence level means we can be 99% certain with only 1% of the samples being outside of the confidence interval. The 95% confidence level is more widely used by researchers.

3.2.2 Sample size

Having decided upon a confidence level of 95%, we next need to determine the sample size. This can be done by two steps:

- Pre-sampling to calculate the “Standard Deviation” (also called “mean square error”);
- Calculate the number of sample using specific formula.

The Standard Deviation is a simple measure of the variability or dispersion of a data set. A low standard deviation indicates that all of the data points are very close to the mean, while a high standard deviation indicates that the data are “spread out” over a large range of values. [31] In the real world, to find out the standard deviation of an entire population is unrealistic, except when every member of a population is sampled. Therefore the standard deviation is usually *estimated* by examining a random sample taken from the population, which is called sample standard deviation, defined by:

$$s = \sqrt{\frac{\sum_{i=1}^{n} \left(s_i - \overline{s}\right)^2}{n-1}}$$ (Formula 3.1) [31],

$$s = \sqrt{\frac{\sum_{i=1}^{n} \left(s_i - \overline{s}\right)^2}{n}}$$ (Formula 3.2) [31],

Here n is the pre-sample size, s_i is the ith sample, and $\overline{s}$ is the mean of selected samples. For example, given two groups of samples: {0, 0, 10, 10} {4, 4, 6, 6} have the same mean value: 5; however, the first group’s standard deviation is 5 (calculated by formula 3.2) while the second group’s standard deviation is 1. In statistics, Formula 3.2 is commonly used as another estimator instead of formula 3.1 during pre-sampling to estimate standard deviation. After pre-sampling we can calculate the standard deviation using formula 3.2. With a confidence level of 95% and standard deviation S, we can calculate a minimum sample size by

$$n = \frac{z^2 S^2}{e^2 + \frac{z^2 S^2}{N}}$$ (Formula 3.3) [32];

For a 95% confidence level, z has a value of 1.96 [31]; e is the acceptable standard error that can be set to meet our analysis demand; N is total population. Using formula 3.3 and the parameters e and S we can calculate a necessary sample size for our data collection.

3.3 What sort of traffic is needed?

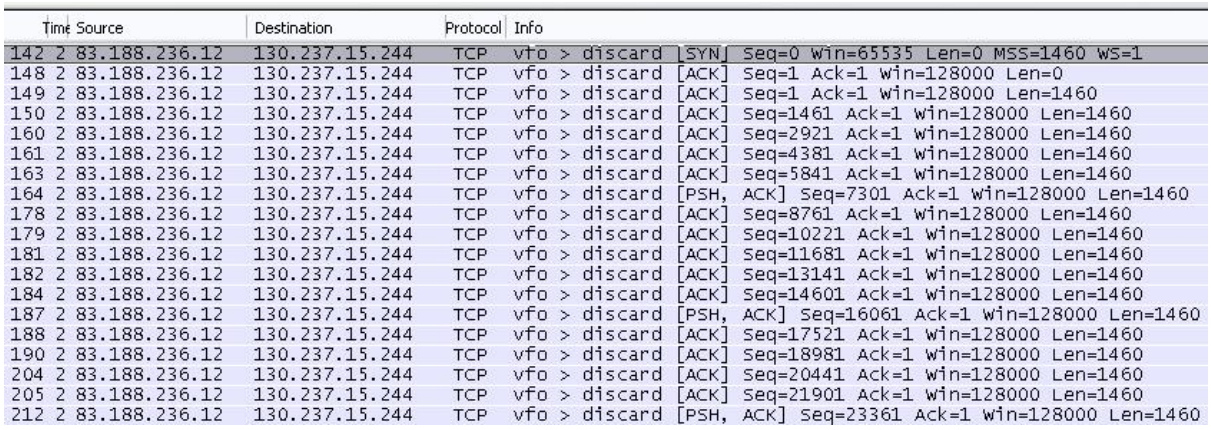
TCP traffic and UDP traffic are commonly used for network performance analysis, while there are a lot of differences between these two types of traffic. Considering the different properties of TCP and UDP traffic will help us to determine an appropriate type of transport protocol to use for data collection.

TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP), providing reliable, ordered delivery of a stream of bytes from one program on one computer to another program on another computer. Besides, TCP controls message size, the rate at which messages are exchanged, and network traffic congestion. [33] However, these advantages can became problems during performance analysis, such as the Nagle Algorithm [34] and delayed ACK (TCP does not immediately ACK every single received TCP segment) complicate the analysis of our data without offering insight into the specific features of the USB attached network interface. These algorithms are summarized in Table 3-1.

Although TCP is more reliable and controllable under normal circumstances, UDP which uses a simple transmission model without implicit hand-shaking dialogues for guaranteeing reliability, ordering, or data integrity, is more suitable for the performance analysis. As UDP provides an unreliable service where datagrams may arrive out of order, be duplicated, or be lost without notice, we can more easily discover the simple time delay of packets sent by a well defined sending process.

Table 3-1: TCP Algorithm and problems [35]

Algorithm	Description	Problems
Nagle Algorithm	Small segments cannot be sent until the outstanding data is acknowledged. This can prevent tiny-gram congestion.	When a TCP connection has outstanding data that has not yet been acknowledged, small segments cannot be sent until the outstanding data is acknowledged.
Sliding Window	The receiver sends the ACK with the advertised window size, avoiding overflowing the buffer.	A fast Sender can be stopped by a closed advertised window.
Slow Start	To avoid network congestion, the sender sets a congestion window: 1, 2, 4... If the router starts discarding packets, then the congestion window will be decreased.	The sender can transmit only up to the minimum of the congestion window and the advertised window.
Congestion Avoidance	Growth of congestion window is additive. Indications of packet loss are timeouts or duplicate ACKs.	When congestion occurs, the transmission rate of packets should be slowed down.



Time	Source	Destination	Protocol	Info
142 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
148 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=1 Ack=1 Win=128000 Len=0
149 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=1 Ack=1 Win=128000 Len=1460
150 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=1461 Ack=1 Win=128000 Len=1460
160 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=2921 Ack=1 Win=128000 Len=1460
161 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=4381 Ack=1 Win=128000 Len=1460
163 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=5841 Ack=1 Win=128000 Len=1460
164 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [PSH, ACK] Seq=7301 Ack=1 Win=128000 Len=1460
178 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=8761 Ack=1 Win=128000 Len=1460
179 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=10221 Ack=1 Win=128000 Len=1460
181 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=11681 Ack=1 Win=128000 Len=1460
182 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=13141 Ack=1 Win=128000 Len=1460
184 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=14601 Ack=1 Win=128000 Len=1460
187 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [PSH, ACK] Seq=16061 Ack=1 Win=128000 Len=1460
188 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=17521 Ack=1 Win=128000 Len=1460
190 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=18981 Ack=1 Win=128000 Len=1460
204 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=20441 Ack=1 Win=128000 Len=1460
205 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [ACK] Seq=21901 Ack=1 Win=128000 Len=1460
212 2	83.188.236.12	130.237.15.244	TCP	vfo > discard [PSH, ACK] Seq=23361 Ack=1 Win=128000 Len=1460

Figure 16: TCP traffic capture on Wireshark

Figure 16 shows TCP packets captured using Wireshark, we can see “SYN” packet, “ACK” packets, and “PSH, ACK” packets have different lengths, including not only the data generated from the packet-generator but also some connection and control messages; while Figure 17 shows UDP packets captured using Wireshark, which shows only the packets generated by the packet-generator, showing that measuring using UDP for one-way time delay calculating and analysis.

Time	Source	Destination	Protocol	Info
1	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
2	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
3	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
4	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
5	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
6	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
7	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
8	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
9	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
10	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
11	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
12	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
13	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
14	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard
15	2 83.188.221.214	130.237.15.244	UDP	Source port: caiccipc Destination port: discard

Figure 17: UDP traffic capture on Wireshark

3.4 Tools for analysis

For the experimental design, we must select appropriate tools and applications. Along with traffic analysis, a packet generator is needed for generating traffic. A good traffic capturer and analyzer play a core role throughout data collection and analysis. How to ensure time synchronization should be considered when calculating one-way time delay.

3.4.1 Packet generator

For pre-measurements and subsequent measurements, both a TCP packet generator and UDP packet generator are needed. As a TCP packet generator we used, TCP-spray is a well known network utility which is commonly used to measure network throughput between two computers. Running the windows version of TCP-spray in the command line, we can see the usage of this program from its help information:

Usage: tcpspray [-v] [-h] [-b blksize] [-n nblks] host
-v verbose
-h print the help message
-b block size in bytes <default value is 1024>
-n number of blocks to send <default value is 100>
host expected receiver's address

This program enables us to set any packet size and numbers of packets to generate for different experimental purposes. We can collect the resulting data as we want. Figure 18 shows an example of using TCP-spray:

```
D:\others\movie>tcpspray -b 8192 -n 300 130.237.15.244

Transmitted 2465792 bytes in 66.281 seconds (36.330 .005 kbytes/s)
```

Figure 18: Example of using TCP-spray

In this case TCP-spray has sent three hundred blocks each with a size of 8192 byte to the

host 130.237.15.244; the result is that 2465792 bytes was transmitted in 66.281 seconds, and the average transmit speed is 36kbytes/s. We should note that the TCP-spray program uses a TCP service called “discard” on the target computer, but that service is not always present. This service exists on most UNIX machines, but even so, many administrators have turned it off to keep the machine as secure as possible. Discard is also part of the "Simple TCP/IP Services" offered by Microsoft’s Windows NT. In order to use the TCP-spray program, we need to enable the “discard” service on the destination host and open the firewall for this traffic.

With equivalent functions as TCP-spray, a UDP packet generator was implemented (see chapter four), by which the user can set the packet size and number of packets freely. This means we can generate and send a large number of large packets to create heavy traffic, which is useful for the peak measured throughput analysis. We should also notice that in this case a UDP “discard” service need to be enabled on the receiver.

3.4.2 Wireshark

As discussed in section 2.2.5 , Wireshark is a good application for network troubleshooting and analysis. As we use packets generator to generate and send packets, Wireshark will listen to the interface and capture the transmitted traffic. After capturing the desired number of bytes (packets), we can use Wireshark to calculate some statistical values and results.

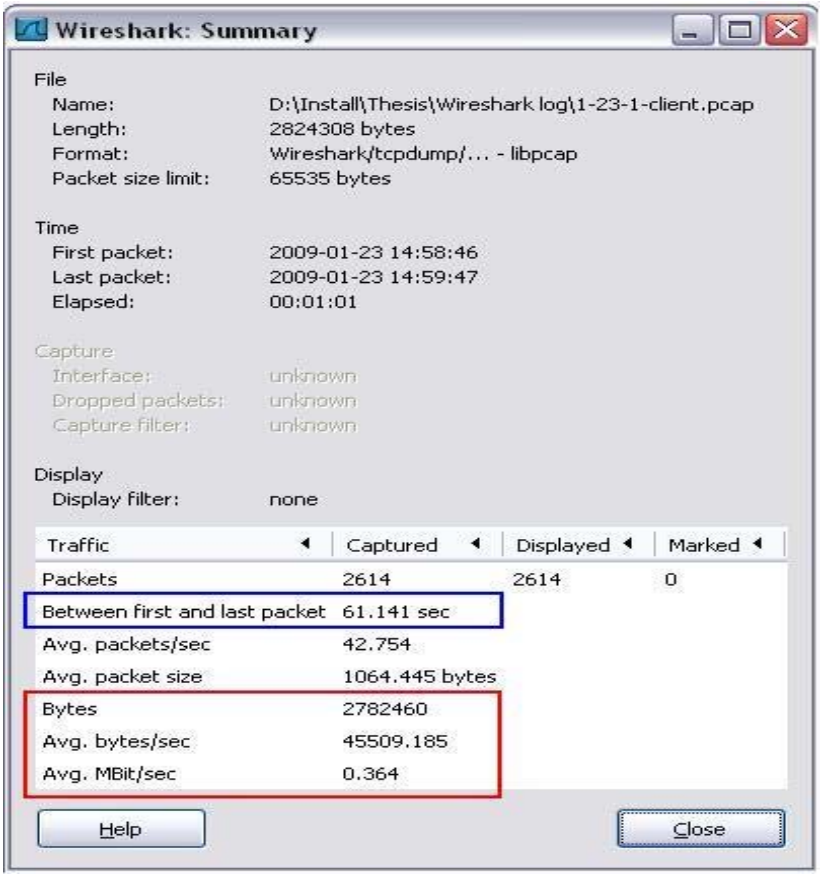


Figure 19: Wireshark summaries

Figure 19 shows a summary of a previous period of traffic transmission. In the red rectangle we can see that 2782469 bytes of data has between transmitted, during a transmit period of 61.141 seconds (displayed in the blue rectangle), yielding an average speed of 45509.185 bytes/sec. As Figure 20 displays, Wireshark can produce I/O graphs providing a more visible understanding of the performance. We note that this graph shows that there is a significant start up time (nearly 5 seconds) before the throughput increases and there is a shut down time of several seconds as the rate drops to zero. Hence the real average data rate is quite different from the peak rate and the average sustained rate.

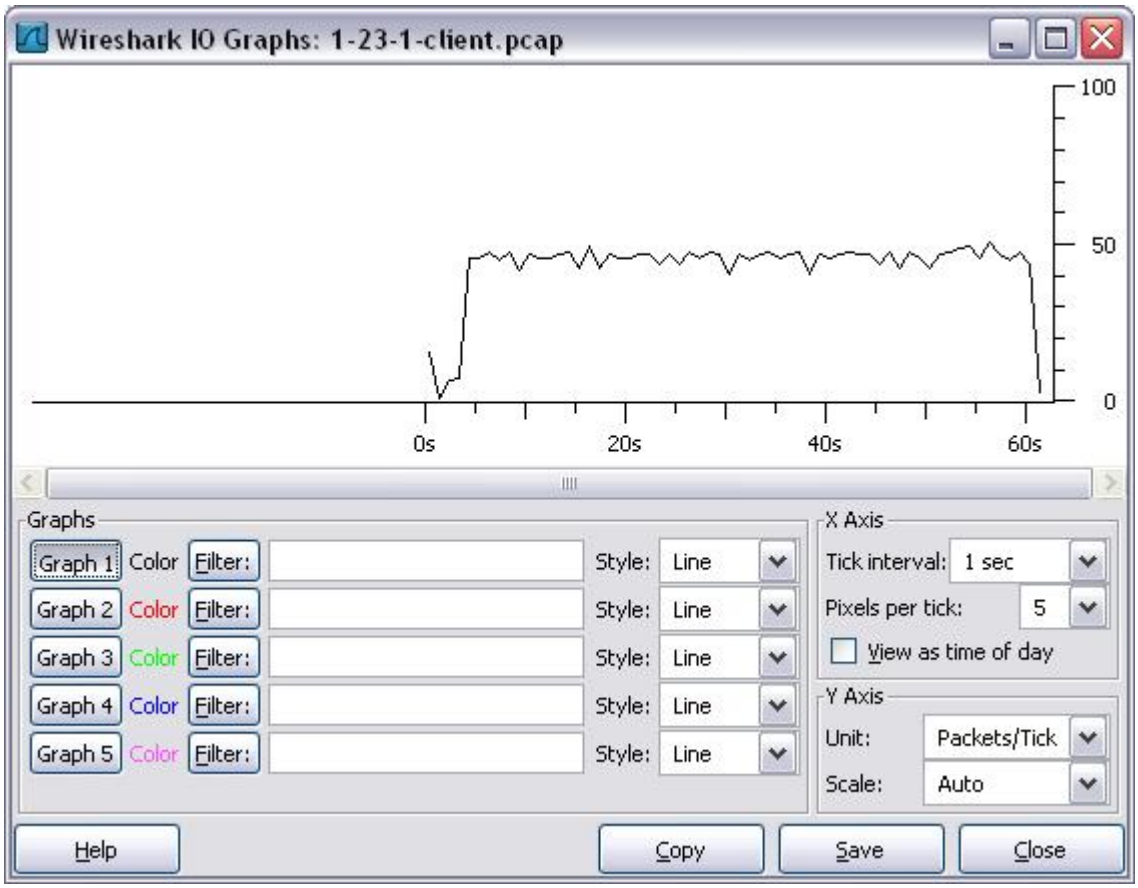


Figure 20: Wireshark IO Graphs

3.4.3 NTP server

The network time protocol (NTP) is a protocol designed to synchronize the clocks of computers over a network. In this thesis, as we would like to calculate time delay of packet transmission from a node to another node, we have to make certain that the two nodes are synchronized to same clock, thus time synchronization need to be configured on both hosts.

Lots of servers providing NTP service can be found on the Internet. In this case, we select “server 0.se.pool.ntp.org” as the NTP server for our time synchronization. When the two nodes are synchronized to same clock, the packets arrival time displayed on Wireshark became accurate and useful. This enables the merger of the sending hosts Wireshark capture

with the receiving host Wireshark capture. As Figure 21 shows, Wireshark can be configured to display time using a high precision format “yyyy-mm-dd hh:mm: ss.xxxxxx”:

No.	Time	Source	Destination	Protocol
1	2009-01-14 15:46:04.923828	217.78.20.129	83.188.219.162	ISAKMP
2	2009-01-14 15:46:10.861328	217.78.20.129	83.188.219.162	ISAKMP
3	2009-01-14 15:46:19.126953	83.188.219.162	130.237.15.244	TCP
4	2009-01-14 15:46:19.376953	130.237.15.244	83.188.219.162	TCP
5	2009-01-14 15:46:19.376953	83.188.219.162	130.237.15.244	TCP
6	2009-01-14 15:46:19.376953	83.188.219.162	130.237.15.244	TCP
7	2009-01-14 15:46:19.376953	83.188.219.162	130.237.15.244	TCP
8	2009-01-14 15:46:20.439453	130.237.15.244	83.188.219.162	TCP
9	2009-01-14 15:46:20.439453	83.188.219.162	130.237.15.244	TCP
10	2009-01-14 15:46:20.439453	83.188.219.162	130.237.15.244	TCP
11	2009-01-14 15:46:20.439453	130.237.15.244	83.188.219.162	TCP
12	2009-01-14 15:46:20.439453	83.188.219.162	130.237.15.244	TCP
13	2009-01-14 15:46:20.455078	83.188.219.162	130.237.15.244	TCP
14	2009-01-14 15:46:20.705078	130.237.15.244	83.188.219.162	TCP
15	2009-01-14 15:46:20.705078	83.188.219.162	130.237.15.244	TCP
16	2009-01-14 15:46:20.705078	83.188.219.162	130.237.15.244	TCP
17	2009-01-14 15:46:20.908203	130.237.15.244	83.188.219.162	TCP

Figure 21: Time formats in Wireshark

When the packet generator sends a packet ‘x’, there will be a time ‘a’ recorded on the sender. Then we can get a corresponding time ‘b’ on the receiver when packet ‘x’ arrives at the destination interface. Therefore the time delay can be obtained directly as ‘b-a’. Given the time delay of each packet, we can analyze the time delay distribution and characterize this distribution’s average value as a result. Additionally, we can learn the variance from this average (or mean).

For Windows system, we have to manually set the clock synchronization to an NTP server in the “Date and Time Prosperities” window (shown in Figure 22).

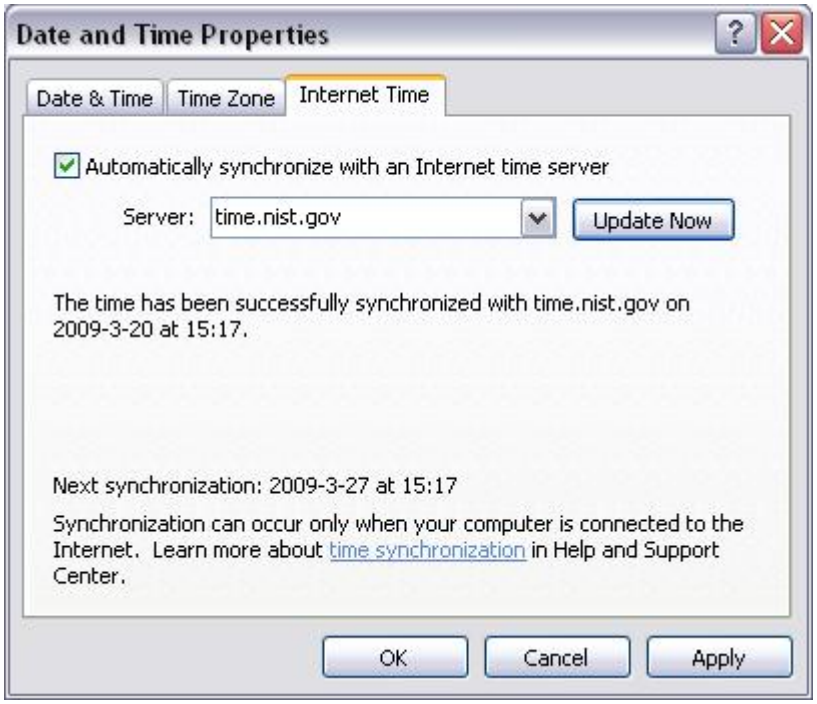


Figure 22: Clock synchronization on Windows machine

Each time sending the UDP packets, we manually update the clock at first to make sure that the time is successfully synchronized to the NTP server. On the other hand, Linux machine does not need manually adjustment. Pre-configurations of NTP services enable Linux machine automatically update its clock per a period of time.

4. Implementation issues

In the implementation phase, the experimental design process was divided into experiment architecture, related programming, sampling, and data collecting.

4.1 Experiment architecture

The experiment involves two parties: sender and receiver; the sender generates packets and sends them to the receiver as shown in Figure 23.

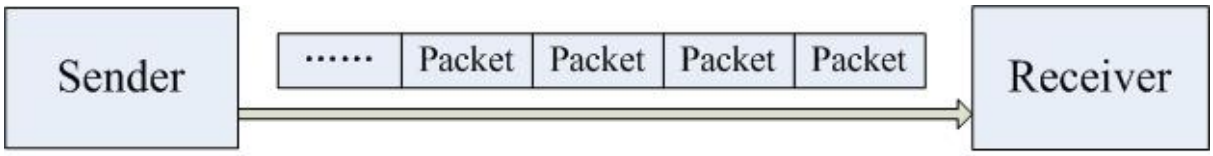


Figure 23: Two parties communicating

In this case, we use a notebook computer as the sender. Since the sender connects to 3G networks through a USB-attached network interface, it should be mobile host, therefore a notebook computer is more suitable for this host rather then a fixed computer. The sender's configuration is shown in Figure 24. While running the packet generator on the sender, monitoring tools such as Wireshark and the USB sniffer will be used to listen to the interface and capture packets passing by the interface, i.e. establishing when each packet is sent to the USB modem.

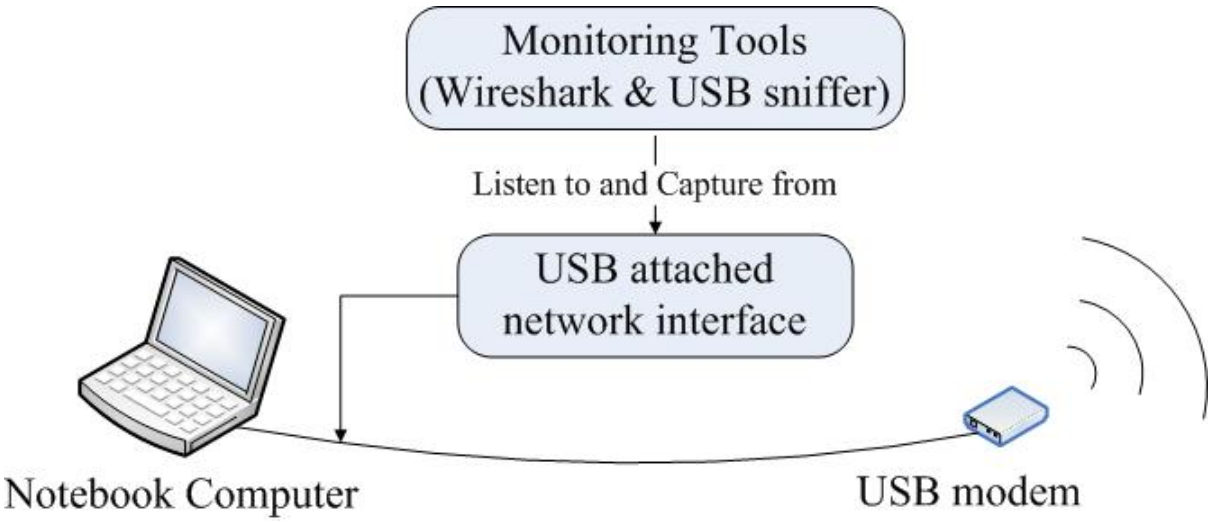


Figure 24: Sender's configuration

A server was set up as the receiver. This server was directly connected to the Internet. The specific port for the discard service on the sender and receiver were opened in their respective firewalls to send a large numbers of packets. While the sender generates and sends packets over the networks, Wireshark captures these packets on the local area interface of the server. The complete experiment architecture is shown in Figure 25.

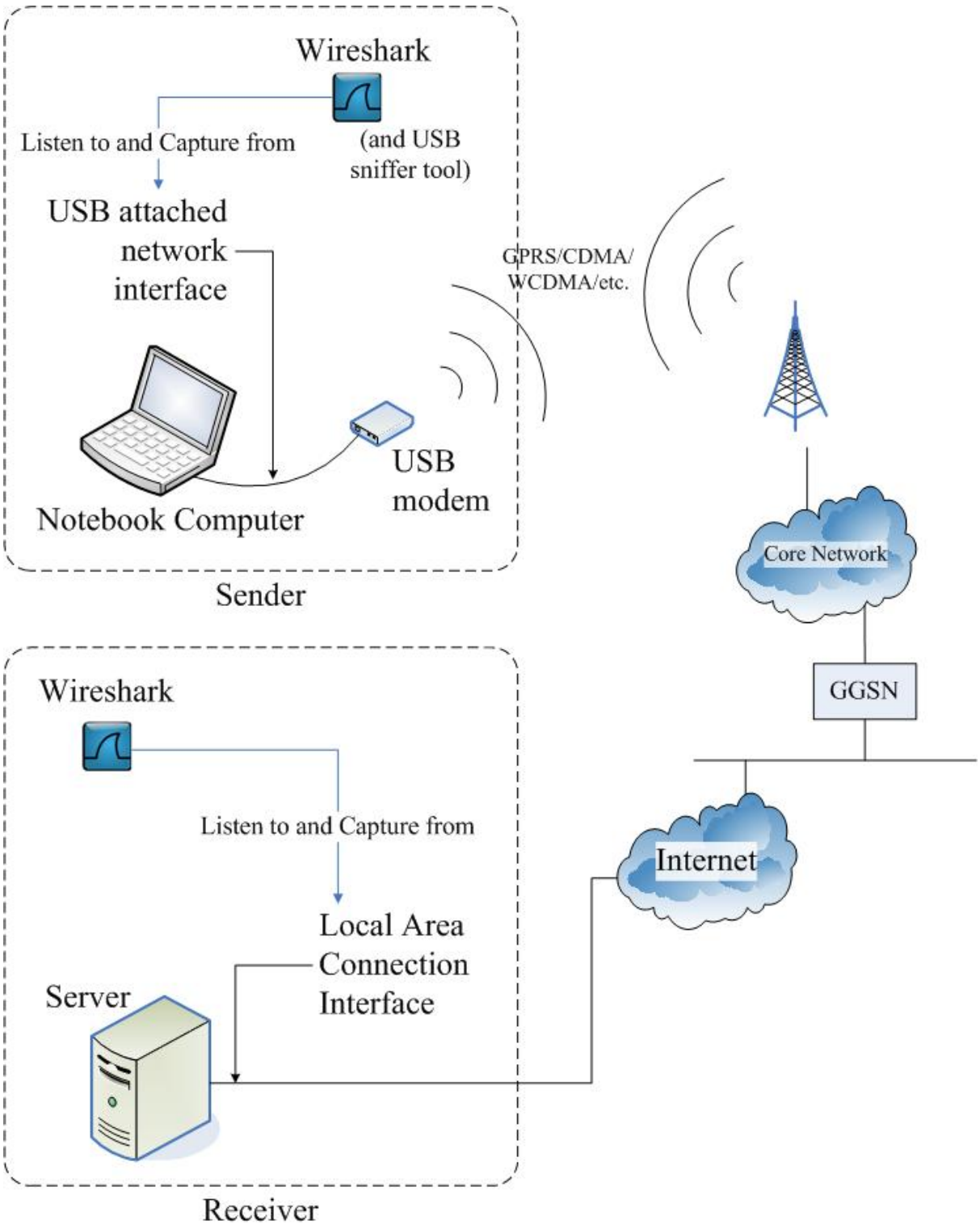


Figure 25: Complete experiment architecture

In the actual experiment, the notebook computer is running Windows XP while the server is running the Linux operating system. To make sure the packet can successfully transferred from the sender to the receiver, some configuration is need before collecting data, such as firewall configuration, system time synchronization, installation of related programs and applications, etc.

4.2 Related programming

For generating and sending UDP packets, a program was written in the C programming language. This program acts as packet generator. Following code and corresponding explanation (as comments is green) were written: [36]

```
#include <Winsock2.h>
//Winsock2.h is used for Windows socket programming;

#include <stdio.h>

#define Numer_of_Packets_to_Send 2000
//set how many packets will be sent, we can change it to any value we want;

#define bigBufferSize 1024
//set block size; the two lines of codes above indicate that 2000 packets of 1024
//bytes of data will be generated and sent when running the program;

void main ()
{
    WORD wVersionRequested;
    //this parameter specifies which version of Winsock we want to load,
    //for example, the value for a Pocket PC needs to be version 1.1;

    //initialize Winsock:
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD (1, 1);
    //create the value of Winsock version

    err = WSAStartup (wVersionRequested, &wsaData);
    if (err!= 0) {
        return;
    }
    //make sure that the correct version of winsock.dll is loaded into memory;
```

```
if ( LOBYTE( wsaData.wVersion ) != 1 ||HIBYTE( wsaData.wVersion ) != 1 ) {
    WSACleanup( );
    return;
}
//make sure that the correct version of winsock.dll is loaded into memory;
//otherwise quit;

SOCKET sockClient=socket (AF_INET,SOCK_DGRAM,0);
//open a socket on the client side to send UDP packets to the network;

SOCKADDR_IN addrSrv;
//create a structure to hold the server's address;
//initialize the structure to the server's address:

addrSrv.sin_addr.S_un.S_addr=inet_addr ("130.237.15.244");
//set the server address, here it is set to 130.237.15.244;

addrSrv.sin_family=AF_INET;
addrSrv.sin_port=htons (9);
//9 is the UDP port number for Discard;
// initialize the structure of server address ends;

int sizeOfBufferToSend = bigBufferSize;
// send the whole buffer - even if there are nulls within it;
int sequenceNumber=0;
// create a sequence number for UDP packets;
SYSTEMTIME my_tv;
//create a field in UDP packets to present time;
char bigBuffer[bigBufferSize]; //empty buffer;
int i;

//set socket to non-blocking mode; this setting can be removed when we need
//blocking mode in the experiment;
unsigned long argp = 1;
HRESULT ret = ioctlsocket (sockClient,FIONBIO,(unsigned long*)&argp);
if (ret == SOCKET_ERROR)   {
    closesocket (sockClient);
    if (WSACleanup() == SOCKET_ERROR) {
        perror ("Could not cleanup sockets");
        return; }
    return; }
```

```
    //start sending packets process:
    for (i=0; i < Numer_of_Packets_to_Send; i++) {
        GetSystemTime (&my_tv);//get system time;
        Sprintf (bigBuffer, "sequence number = %d; at time %ld. %ld\n",
        sequenceNumber++, my_tv.wSecond, my_tv.wMilliseconds);
        //put the sequence number and system time into the buffer;
        if ((sendto (sockClient, bigBuffer, sizeOfBufferToSend, 0, (SOCKADDR*) &
        addrSrv, sizeof (SOCKADDR))) == -1) {
            perror ("Unable to send to socket");
            closesocket (sockClient);
            exit (1);
        }
    }

    closesocket (sockClient);
    //close the socket on the client;

    WSACleanup ();
    //to maintain compatibility with desktop applications that have been ported to
    //Windows CE
}
```

Notice that there is a paragraph of codes above to set the socket to non-blocking mode. In the socket configurations, if we do not set the socket to non-blocking mode, the speed of sending will be limited to the rate at which the operating system can empty the socket's buffer; otherwise the "sendto" function will quit when the sending buffers of the operating system are all full. The default mode is blocking mode, this way the program will run at the rate of the network link. In the blocking mode, if space is not available at the sending socket to hold the message to be transmitted, the function "sendto" will block until space is available, and the program will wait for "sendto" to finish. In our experiment, we performed both blocking mode and non-blocking mode sending to see the differences.

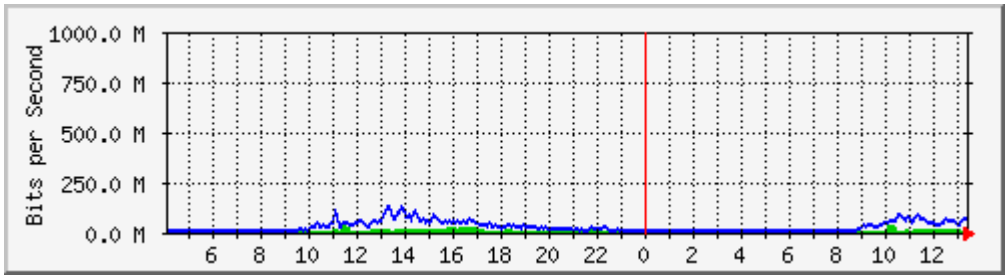
4.3 Sampling and data collecting

As discussed in section 3.1.1 , we chose "peak measured throughput" as our analysis target, as this was more relevant for the end customer. Peak measured throughput is measured by a real, implemented system over a short period of time, thus we need to find out when the peak traffic occurs, then generate samples during the peak hours, and collect significant amount of data.

4.3.1 Peak traffic

According to previous research [37] in Kista (a suburb of Stockholm, Sweden, where we had the experiments), traffic analysis shows the load out from the wireless that was split on various interfaces in the routers as displayed in Figure 26 and Figure 27. These statistics were last updated Tuesday, 20 January 2009 at 15:00.

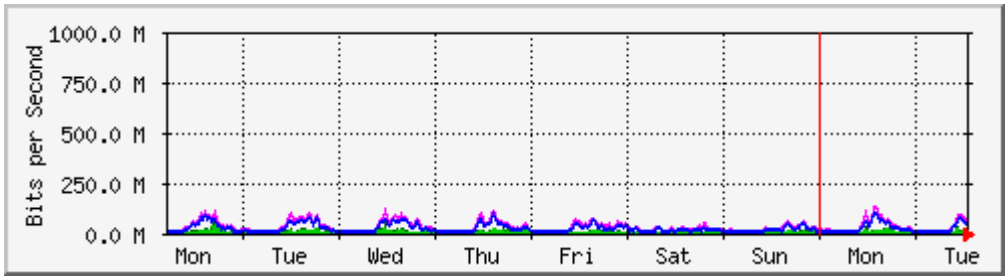
`Daily' Graph (5 Minute Average)



Max In: 59.4 Mb/s (5.9%) Average In: 12.1 Mb/s (1.2%) Current In: 26.0 Mb/s (2.6%)
Max Out:98.0 Mb/s (9.8%) Average Out:23.9 Mb/s (2.4%) Current Out:47.5 Mb/s (4.7%)

Figure 26: Daily statistics of wireless traffic [37]

`Weekly' Graph (30 Minute Average)



Max In: 59.6 Mb/s (6.0%) Average In: 8140.1 kb/s (0.8%) Current In: 26.5 Mb/s (2.6%)
Max Out:134.1 Mb/s (13.4%) Average Out:18.1 Mb/s (1.8%) Current Out:64.7 Mb/s (6.5%)

Figure 27: Weekly statistics of wireless traffic [37]

From Figure 26 we can see that the most traffic is around 14:00 – 16:00 everyday, and from Figure 27 we can see that the most traffic during a week is on Friday. Note that this is **not** the actual peak hour of the 3G cellular network, which is around 18:00 [38]. However, during these hours most of customers are connecting to the Internet; this is more relevant for the end users. Therefore we select “14:00 – 16:00, Friday” as the period of peak traffic, and we also perform additional measurements on 3G network’s peak hour (at 18:00) for comparison.

4.3.2 Sample size calculation

For sampling the traffic between 14:00 – 16:00, that is, a 120 minute interval, we can chose 1 minute to be our basic unit of time, thus the population becomes 120.

Correspondingly, in formula 3.3 (see also section 3.2.2), we can say the value of “N” is 120.

The first step is to calculate the standard deviation “S” using formula 3.2.

$$s = \sqrt{\frac{\sum_{i=1}^{n} (s_i - \overline{s})^2}{n}}$$ (Formula 3.2) [31],

Picking a few pre-samples, we calculate the mean of these samples and get an estimate for “$\overline{s}$”, where “s_i” is each sample, “n” as the number of these samples. Substituting these factors into the formula; we estimate the value of “S”. In my experiments, I got a value of 0.2405 in this case. This value is called the sample standard deviation, but is usually used to estimate the standard deviation. We can regard them as a same value.

Next we can calculate the expected sample size using formula 3.3 (see also section 3.2.2).

$$n = \frac{z^2 S^2}{e^2 + \frac{z^2 S^2}{N}}$$ (Formula 3.3) [32];

Using the 95% confidence level we selected (see section 3.2.1), the value of “z” is 1.96 [31]. Therefore, S=0.2405 and N=120, thus we can select an acceptable sample standard error “e” as 15% and calculate the value of “n”:

$$n = \frac{1.96^2 \times 0.2405^2}{0.15^2 + \frac{1.96^2 \times 0.2405^2}{120}} \approx 9.1$$

This result indicates that we have to pick at least 10 samples from the population of 120 to reach the 95% confidence level. Given the fixed 95% confidence level, the more samples we pick, the less the value of the standard error (“e”), which means that the experiment results will be more reliable.

4.3.3 Data collecting

Based upon the previous calculations I decided the sample size would be 12, that is, twelve samples within a population of 120 will be picked. It means 1 unit (i.e. minute of traffic) will be sampled in every 10 units (minutes) of traffic. Each sample will be analyzed to obtain a corresponding result, thus there will be a group of results with twelve items.

Figure 28 shows the whole procedure.

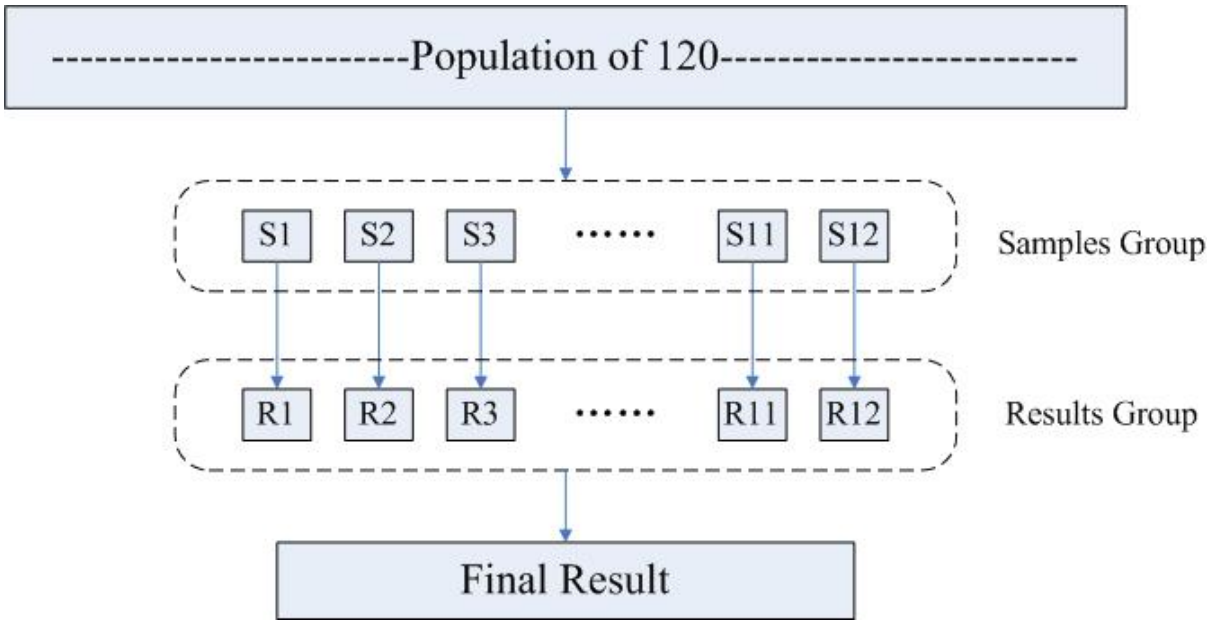


Figure 28: Sampling, data collection and analysis procedure

Notice here we did not divide the population of 120 into 12 groups, with each group having 10 units. Instead we pick 12 samples from the population of 120 at random, which is more likely (but not really, since the ideal condition is hard to achieve in practice) to the model of “simple random sample”. A simple random sample of size n consists of n samples from the population chosen in such a way that every set of n samples has an equal chance to be the sample actually selected, which is ideal for statistical purposes. [39][40]

5. Experimental data analysis

Along with implementation, groups of experimental data have been collected. This chapter provides detailed analysis of the experimental data that was collected: how the traffic was transferred from the sender to the receiver, what the average throughput was based upon the experiments, and what the variation of time delay was during the experiment period.

5.1 Traffic analysis

When the packets generator starts to run, traffic is sent by the application and passes through the USB interface, the 3G-modem, the wide network operator's core network, various networks, and then arrives at the server's network interface. A trace route of the path is shown in Figure 29. We concentrate on the traffic analysis of the USB interface and the USB modem's 3G network interface, in order to gain a detailed understanding of the traffic on such a USB-attached network interface.

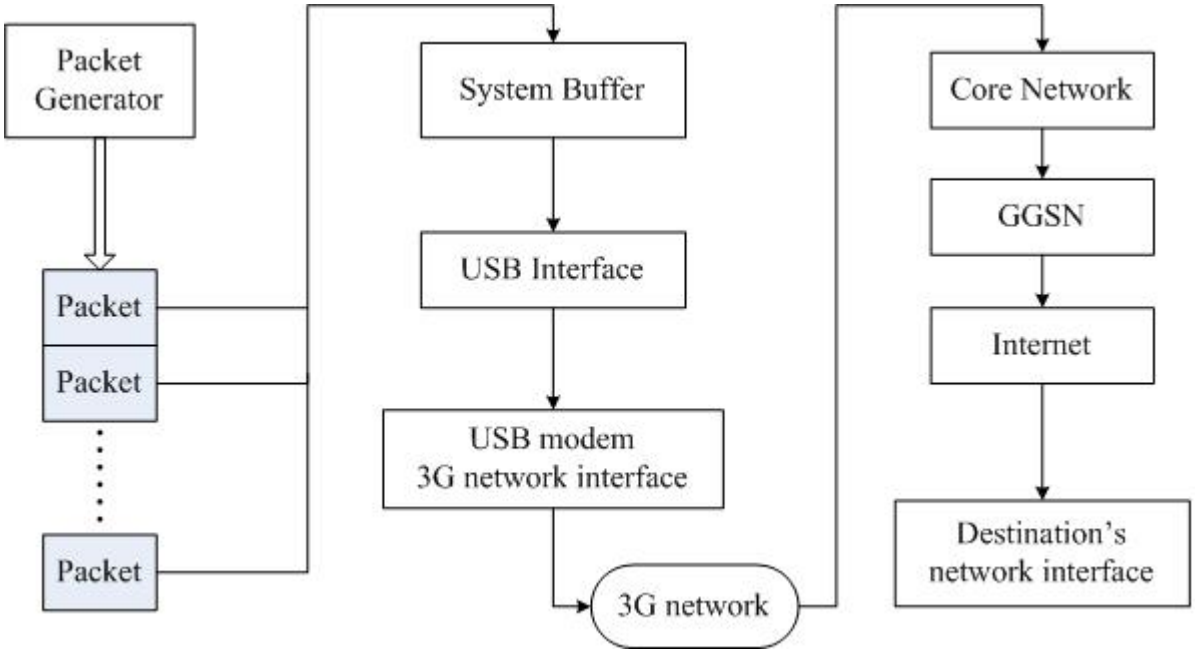


Figure 29: A trace route of traffic path

5.1.1 Traffic analysis of the USB interface

USB traffic takes the form of frames. Initially, all frames are sent from the host to the device (or sometimes several devices). Some of these frames direct a device to send frames in reply. All frames are made of 8-bit bytes, transmitted least-significant bit first. The first byte is a

frame (packet) identifier (PID) byte, which indicates what type of frame it is. [41]

USB frames can be classified into three basic types: Token, Data, and Handshake (actually there is another special type of frame which is usually only supported by low speed devices). [42] Each type of USB frame has a different format. Table 5-1 shows the PIDs for each type of frame.

Table 5-1: USB packet type [42] [43]

Packet type	**PID**
Token	OUT (Host to device transfer); IN (Device to Host transfer); SOF (Start of Frame marker); SETUP (Host to device control transfer)
Data	DATA0, DATA1 (Data packet); DATA2 (High-Speed Data packet) MDATA (Split/High-Speed Data packet)
Handshake	ACK (The data frame was received error free) NAK (Receiver cannot accept data or the transmitter could not send data) STALL (Endpoint halted or control pipe request is not supported) NYET (No response yet)
Special	PRE, ERR, SPLIT, and PING

Figure 30 shows the process of sending USB frames between a host and target device. The transactions occur in three phases: the Token, Data, and Handshake phase. In the “Token phase”, the host initializes the transfer by generating a TOKEN packet, where the type of TOKEN indicates whether the host will transmit or receive data. Then the process goes to the “Data phase” where the transmitter sends a DATA packet. The target USB device will send a NAK or STALL frame to indicate if they are not able to serve the IN token. Finally the process arrives at the “Handshake phase” and the receiver returns an ACK if the transmission was successful. USB devices can additionally send extra HANDSHAKE frame types to indicate problems or errors.

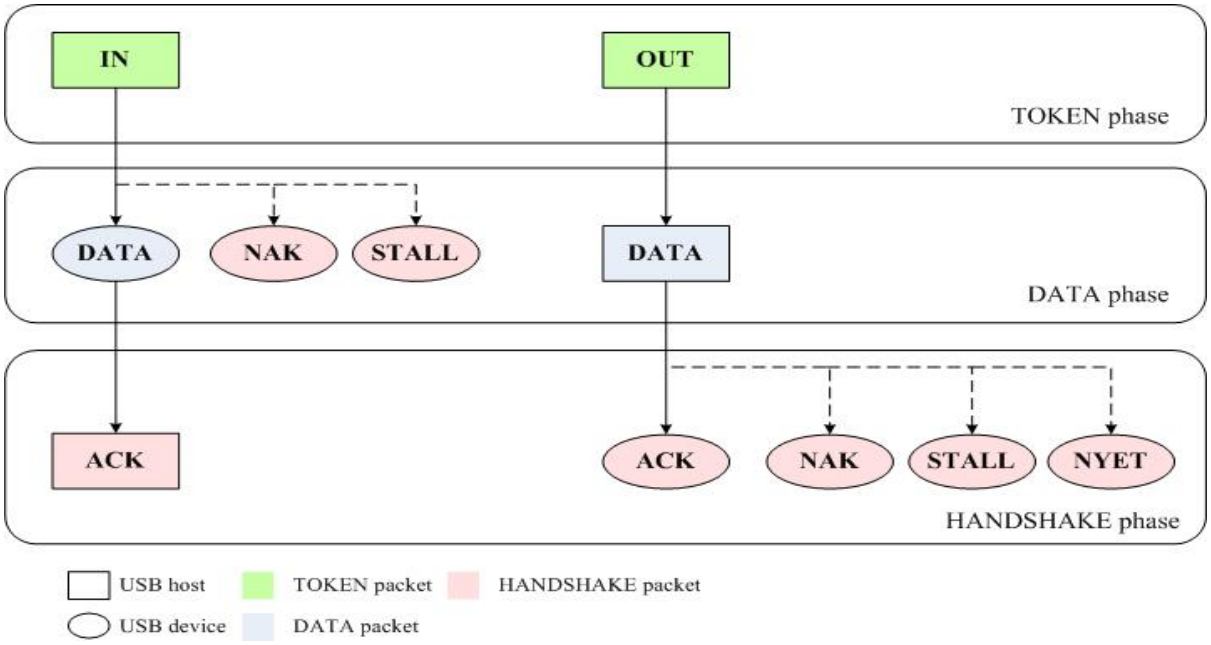


Figure 30: Three phases of USB transmission [43]

All of the transfers described above follow this general scheme with the exception of the isochronous transfer, in which case no Handshake phase occurs — the device reads or writes a single frame of data each millisecond. Fortunately, the driver can send several consecutive frames with a single URB. [43][44] While sending UDP and TCP packets, interrupt transfer mode is commonly utilized. During UDP packet transmission in our experiments we can see this interrupt transfer mode in the URB header; Figure 31 displays the structure of captured USB packets.

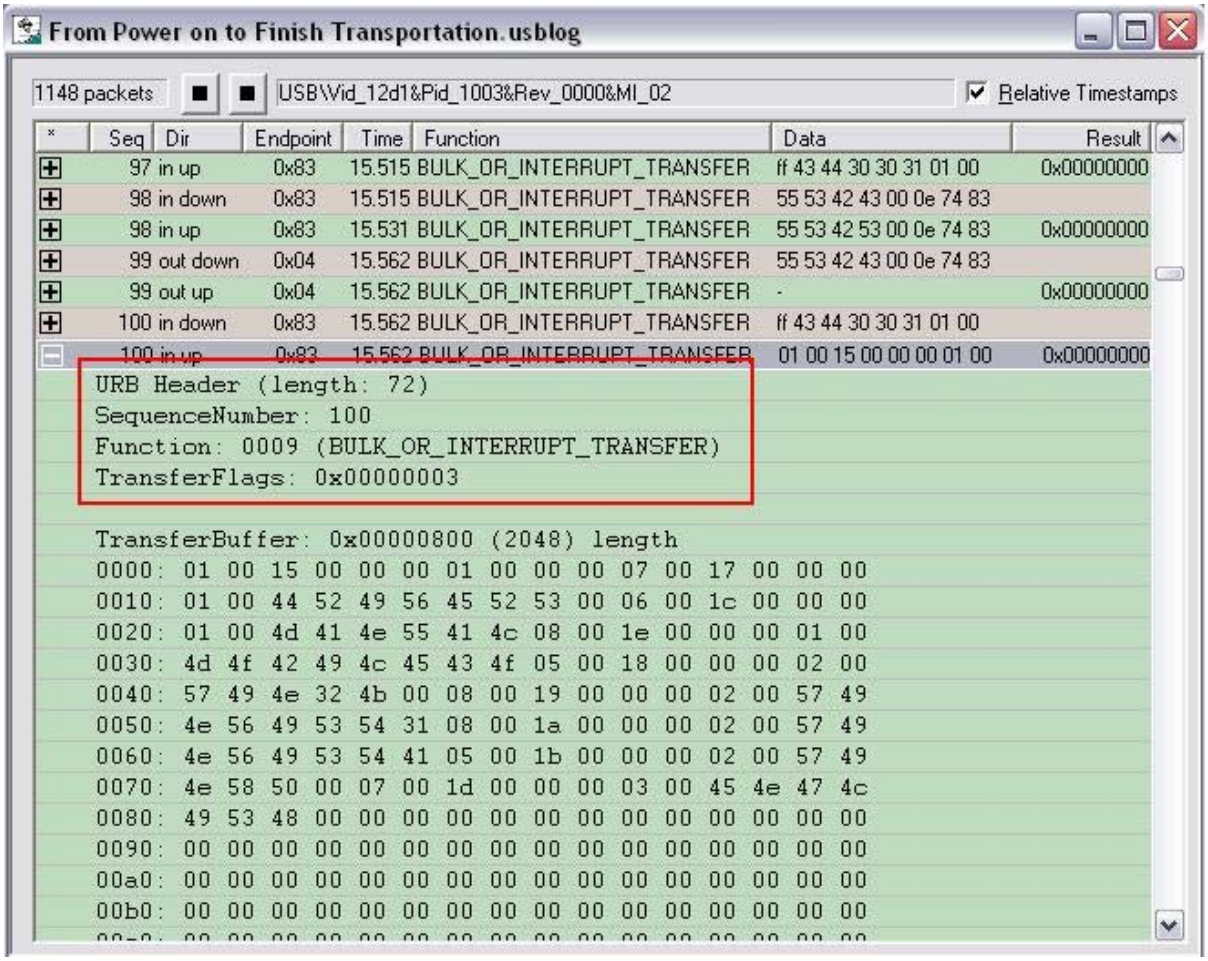


Figure 31: Structure of captured USB frame

We can see a URB header (marked by the red rectangle), with length of 72 bytes, a sequence number, and the specific URB function, followed by transfer flags and the transfer buffer. The URB header has the structure [45]:

```
struct _URB_HEADER {
        USHORT Length;
        USHORT Function;
        USBD_STATUS Status;
};
```

The Length parameter specifies the length of the frame, in bytes, i.e. the length of the URB. For URB requests that use data structures other than _URB_HEADER, this value must be set to the length of the entire URB request structure, not simply the _URB_HEADER size. The status parameter contains a USB_STATUS_XXX code on return from the host controller driver, indicating the USB transfer/command status.

The Function parameter specifies a numeric code indicating the requested operation for this URB. A specific value must be set. In Figure 5-3 we can see the value of “Function” is URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (=0009). This means operation transfers data from a bulk pipe or interrupt pipe or to a bulk pipe. If set; the URB will be used with _URB_BULK_OR_INTERRUPT_TRANSFER as the data structure.

The _URB_BULK_OR_INTERRUPT_TRANSFER structure is defined as [46]:

```
struct _URB_BULK_OR_INTERRUPT_TRANSFER {
    struct _URB_HEADER Hdr;
    USBD_PIPE_HANDLE PipeHandle;
    ULONG TransferFlags;
    ULONG TransferBufferLength;
    PVOID TransferBuffer;
    PMDL TransferBufferMDL;
    struct _URB *UrbLink;
    };
```

The TransferBuffer parameter is displayed in Figure 31. The TransferBuffer is a pointer to a memory resident buffer for the transfer or is NULL if a memory descriptor list (MDL) is supplied in the parameter TransferBufferMDL. The contents of this buffer depend on the value of TransferFlags. If USBD_TRANSFER_DIRECTION_IN (is the value of the TransferFlags parameter) is specified, then this buffer will contain data read from the device on return from the host controller driver. Otherwise, this buffer contains driver-supplied data for transfer to the device (the direction is OUT).

When the packet generator starts, data transmission occurs between host and device, in this case, the USB modem. The time required for this transmission is not explicitly calculated in the traffic time delay (see section 3.1.2) that has been defined in this thesis. However, time spent on transferring data from the host to USB modem is implicitly included together with network delay from the point of view of the customer. Fortunately this delay does not actually bother the user because of the high data rate supported by USB interface (see section 2.2.6).

5.1.2 Traffic analysis of a 3G modem

After being transmitted through the USB interface, data arrives at the 3G modem, preparing to be sent out. Using Wireshark to monitor the network interface of the 3G modem, we can see that Wireshark recognizes the interface as a WAN (PPP/SLIP) interface, as shown in Figure 32.

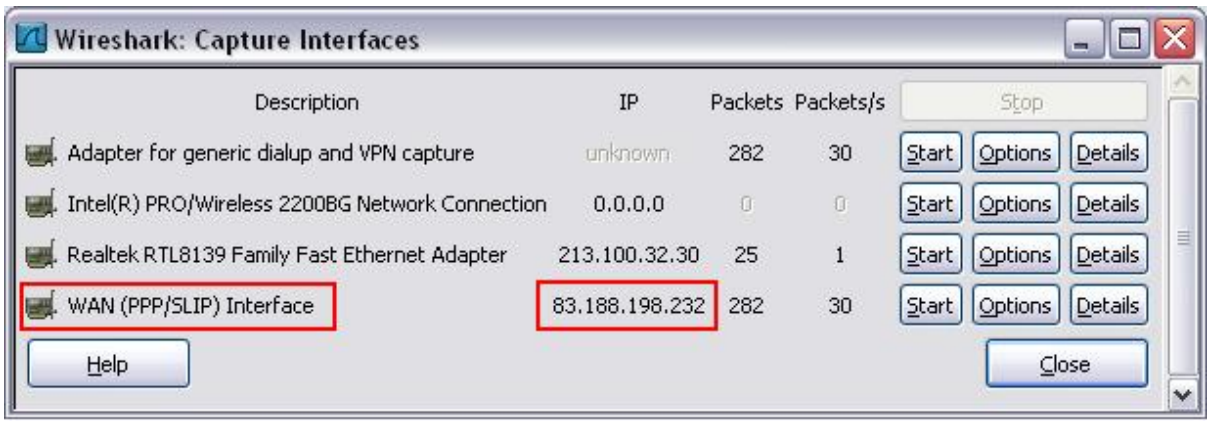


Figure 32: 3G modem network interface on Wireshark

Wireshark indicates that the “WAN (PPP/SLIP) Interface” interface has a (temporarily) assigned IP address, in this case of 83.188.198.232 (each time connect to the 3G network the 3G modem will be assigned a new IP address). The network interface is connected to a Wide Area Network (WAN), supporting one of the following protocols: PPP or SLIP. The Point-to-Point Protocol (PPP) is a data link protocol that is commonly used to establish a direct connection between two network nodes. PPP can optionally provide connection authentication, transmission encryption privacy, and compression. Most Internet service providers (ISPs) use PPP for customer dial-up access to the Internet. [47] The Serial Line Internet Protocol (SLIP) is also a protocol for connection to the Internet via a dial-up connection, but has mostly been replaced by PPP nowadays.

To understand how the 3G modem connects the computer to the network, let us go examining the interactions before running the packet generator - specifically from the time when we first plug-in the 3G modem. We must first perform an authentication as shown in Figure 33 in order to enable the USB modem (actually to enable the USIM in the USB modem) via the device’s control panel displayed in Figure 34.

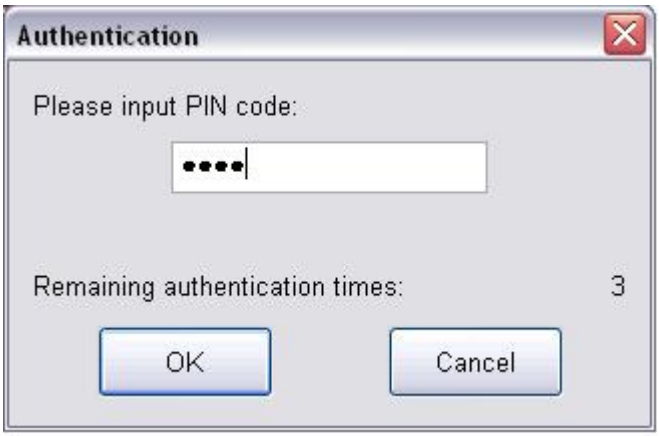


Figure 33: Enabling the 3G modem

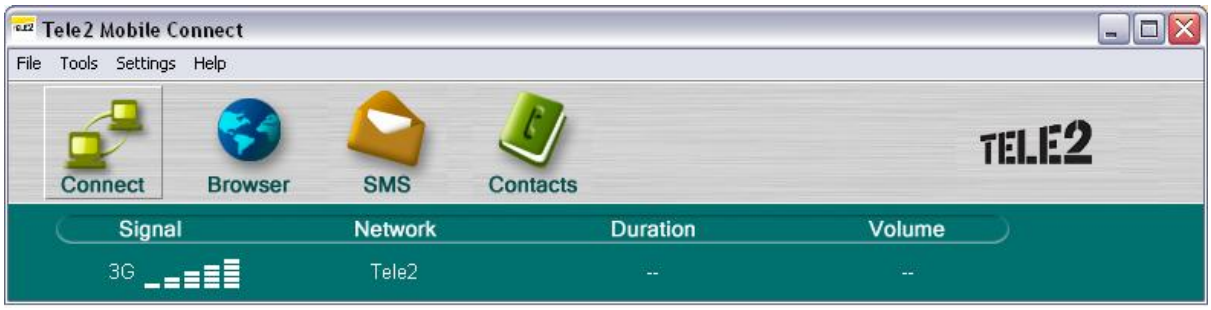


Figure 34: 3G modem control panel

After clicking the “Connect” button, the modem will connect to Tele2’s 3G network. As Figure 32 indicated, the 3G modem uses the PPP/SLIP protocols; this means the modem has dialed-up access to the 3G network. For detailed information, we can click the connection icon in the taskbar to check the connection’s properties. Figure 35 shows overall information about the 3G modem’s status, indicating that it is using PPP (as the server type), it supports TCP/IP transportation, and it is using MD5-CHAP for authentication.

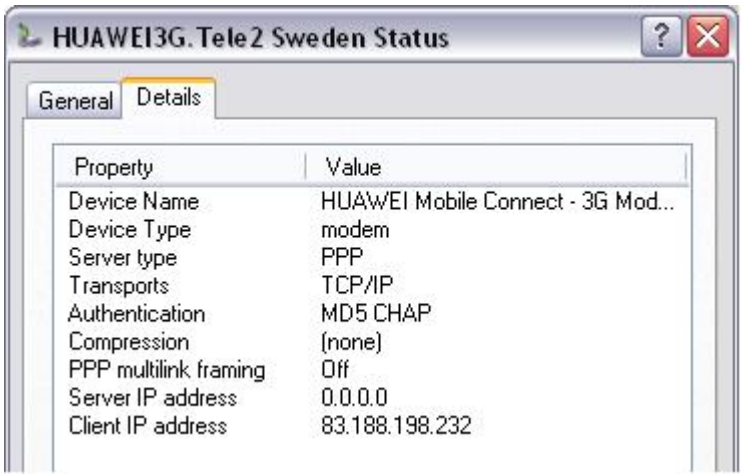


Figure 35: 3G modem connection details

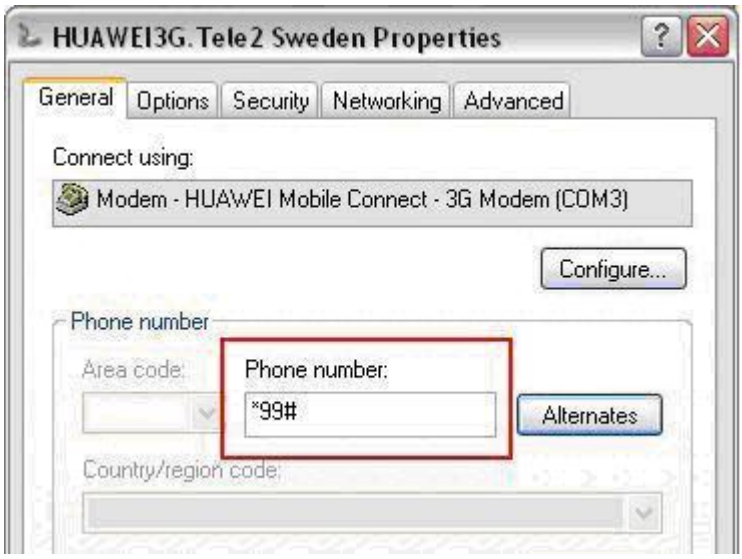


Figure 36: “Phone number” in the modem properties window

There is an item marked “Phone number” in the modem’s properties window (see Figure 36). The value is “*99#”, that means, by dialing this special number, the 3G modem connects to Tele2’s 3G network. And we can also see the two types of dial-up server in Figure 37, a PPP server for the Windows operating system, and a SLIP server for UNIX connections.

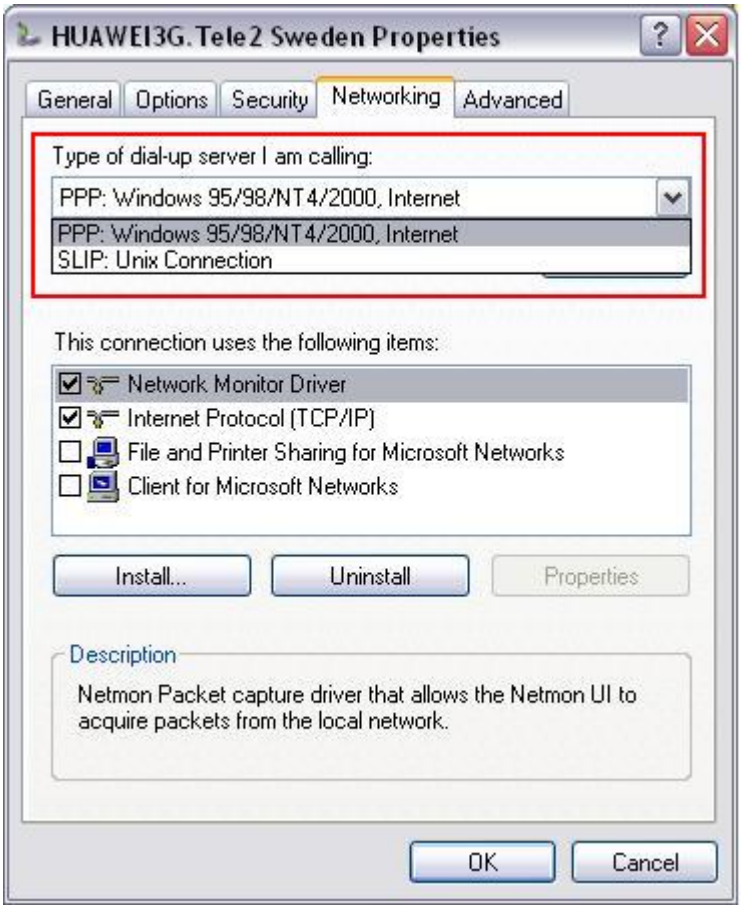


Figure 37: “PPP/SLIP” configuration on 3G modem

Now that it is clear how the 3G modem connects the computer to the network, we can continue with the interface monitoring by Wireshark. The packet generator produces numerous 1024 bytes packets (see section 4.2). Data encapsulation occurs when transferring these packets: each protocol creates a protocol data unit (PDU) for transmission that includes the headers required by that protocol and the data to be transmitted. This data becomes the service data unit (SDU) of the next layer below it.

This procedure is shown in Figure 38; where the 1024 bytes of data has given a UDP header on the transport layer, an IP header on the network layer, an Ethernet header and trailer on the data link layer, before being sent out on the physical layer.

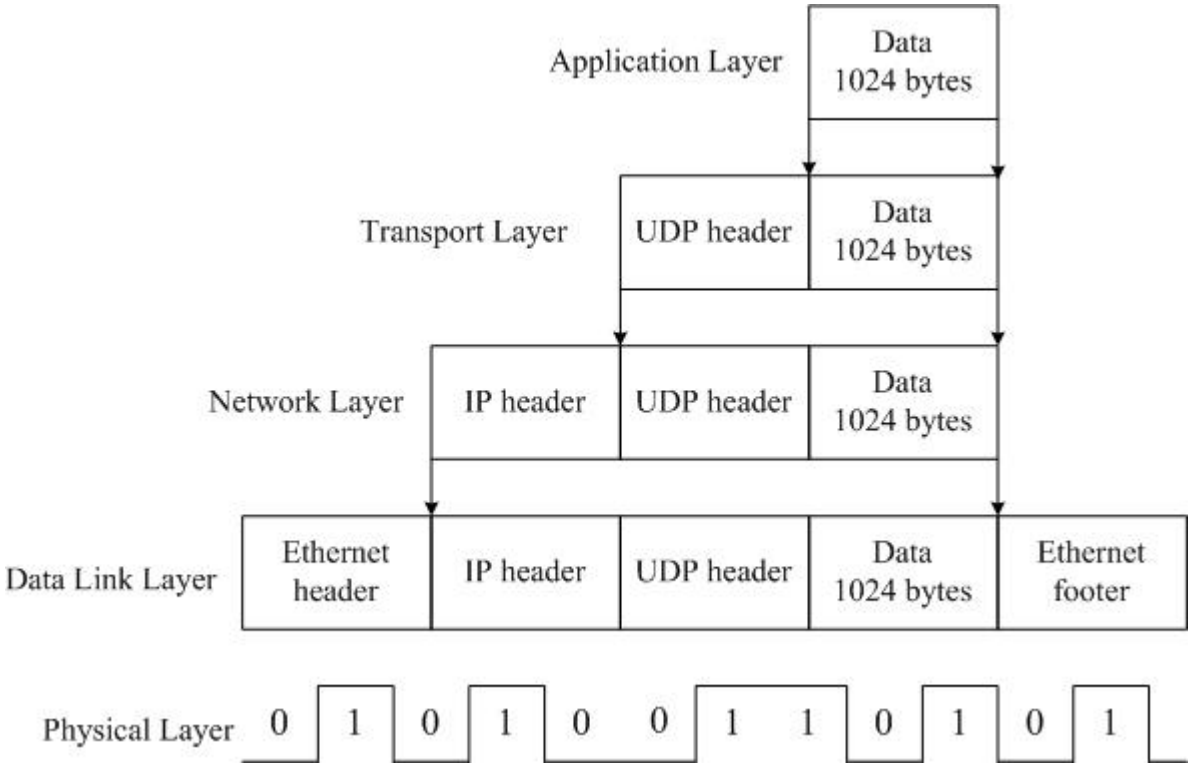


Figure 38: Data encapsulation example

Correspondingly, the packets captured by Wireshark shows the same packet structure, as Figure 39 displays; the 1024 bytes of data becomes a frame of 1066 bytes, including an Ethernet II header, and Internet Protocol header, a user Datagram Protocol header, and 1024 bytes of data.

```
⊞ Frame 1 (1066 bytes on wire, 1066 bytes captured)
⊞ Ethernet II, Src: Xerox_00:00:00 (00:00:03:00:00:00), Dst: a4:98:20:00:03:00 (a4:98:20:00:03:00)
⊞ Internet Protocol, Src: 83.188.230.133 (83.188.230.133), Dst: 130.237.15.244 (130.237.15.244)
⊞ User Datagram Protocol, Src Port: etebac5 (1216), Dst Port: discard (9)
⊞ Data (1024 bytes)
```

Figure 39: UDP frame structure

```
⊟ Frame 1 (1066 bytes on wire, 1066 bytes captured)
    Arrival Time: Mar 13, 2009 14:01:55.761647000
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 1066 bytes
    Capture Length: 1066 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
```

Figure 40: UDP frame information

Figure 40 shows some information about a UDP frame, including arrival time, frame number, frame length, etc. we can see the “Protocols in frame” item shows what types of protocols are included in this frame: “eth:ip:udp:data” indicates Ethernet, IP, UDP are included.

Figure 41 shows the Ethernet header for this UDP datagram, we can see the MAC (Media Access Control) address of the destination and source as well as the frame type (IP). Figure 42 shows the IP header for this UDP datagram, with its version number, header length, total length (from IP header to the end of the packet), identification, flags, fragment offset, time to live, inside data protocol, header checksum, and the IP address of source node and destination nodes. Finally, Figure 43 displays the UDP header, including the source port from the PC, destination port (port 9 is used for discard service of the server), length (from UDP header to the end of the packet), and checksum.

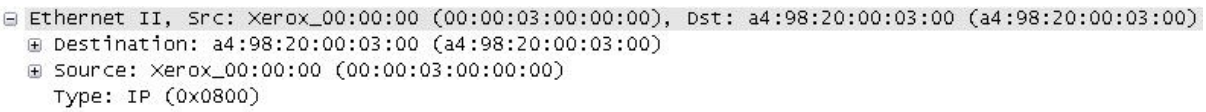


Figure 41: Ethernet II header in a UDP datagram

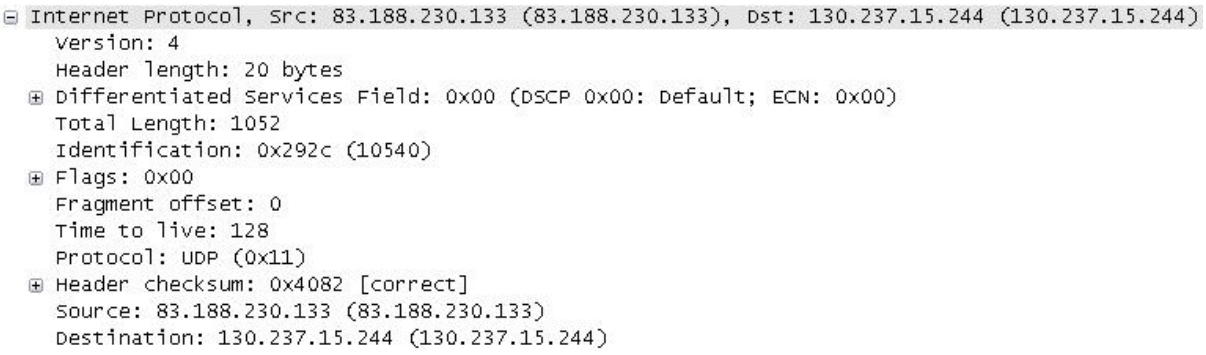


Figure 42: Internet Protocol header in a UDP datagram

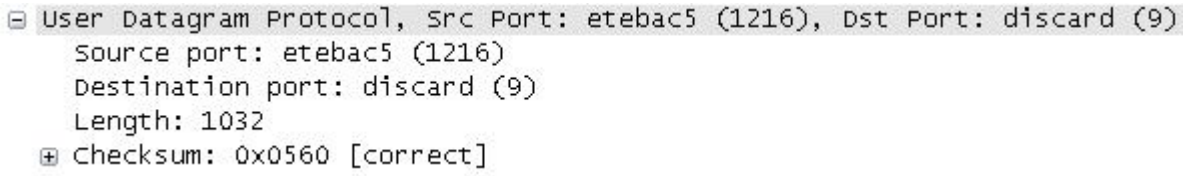
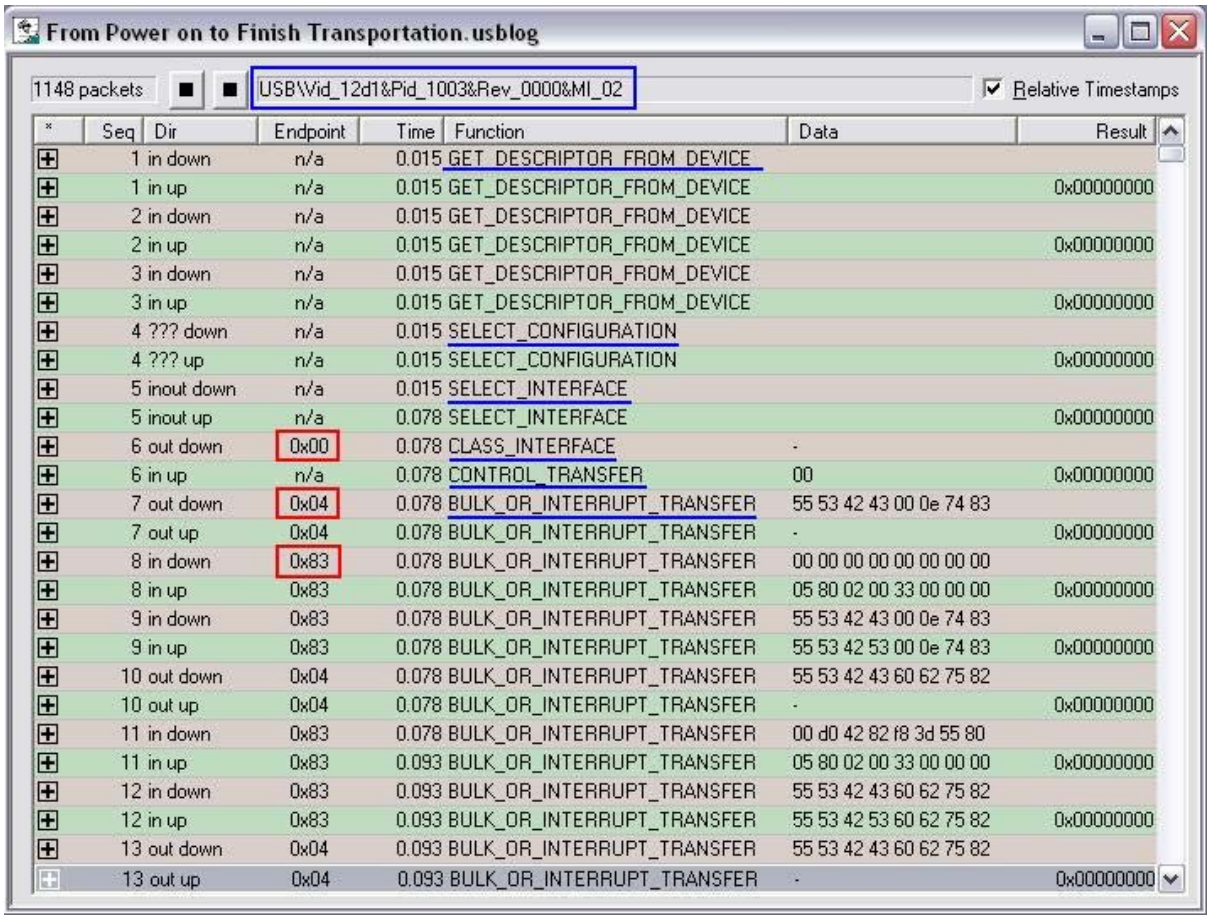


Figure 43: User Datagram Protocol header in a UDP datagram

5.1.3 Tracing packets using USB sniffer and Wireshark

From when we plug in the USB modem, it starts transferring URB packets via the USB interface. Figure 44 shows these URB packets captured by the USB sniffer.



*	Seq	Dir	Endpoint	Time	Function	Data	Result
⊞	1	in down	n/a	0.015	GET_DESCRIPTOR_FROM_DEVICE		
⊞	1	in up	n/a	0.015	GET_DESCRIPTOR_FROM_DEVICE		0x00000000
⊞	2	in down	n/a	0.015	GET_DESCRIPTOR_FROM_DEVICE		
⊞	2	in up	n/a	0.015	GET_DESCRIPTOR_FROM_DEVICE		0x00000000
⊞	3	in down	n/a	0.015	GET_DESCRIPTOR_FROM_DEVICE		
⊞	3	in up	n/a	0.015	GET_DESCRIPTOR_FROM_DEVICE		0x00000000
⊞	4	??? down	n/a	0.015	SELECT_CONFIGURATION		
⊞	4	??? up	n/a	0.015	SELECT_CONFIGURATION		0x00000000
⊞	5	inout down	n/a	0.015	SELECT_INTERFACE		
⊞	5	inout up	n/a	0.078	SELECT_INTERFACE		0x00000000
⊞	6	out down	0x00	0.078	CLASS_INTERFACE	-	
⊞	6	in up	n/a	0.078	CONTROL_TRANSFER	00	0x00000000
⊞	7	out down	0x04	0.078	BULK_OR_INTERRUPT_TRANSFER	55 53 42 43 00 0e 74 83	
⊞	7	out up	0x04	0.078	BULK_OR_INTERRUPT_TRANSFER	-	0x00000000
⊞	8	in down	0x83	0.078	BULK_OR_INTERRUPT_TRANSFER	00 00 00 00 00 00 00 00	
⊞	8	in up	0x83	0.078	BULK_OR_INTERRUPT_TRANSFER	05 80 02 00 33 00 00 00	0x00000000
⊞	9	in down	0x83	0.078	BULK_OR_INTERRUPT_TRANSFER	55 53 42 43 00 0e 74 83	
⊞	9	in up	0x83	0.078	BULK_OR_INTERRUPT_TRANSFER	55 53 42 53 00 0e 74 83	0x00000000
⊞	10	out down	0x04	0.078	BULK_OR_INTERRUPT_TRANSFER	55 53 42 43 60 62 75 82	
⊞	10	out up	0x04	0.078	BULK_OR_INTERRUPT_TRANSFER	-	0x00000000
⊞	11	in down	0x83	0.078	BULK_OR_INTERRUPT_TRANSFER	00 d0 42 82 f8 3d 55 80	
⊞	11	in up	0x83	0.093	BULK_OR_INTERRUPT_TRANSFER	05 80 02 00 33 00 00 00	0x00000000
⊞	12	in down	0x83	0.093	BULK_OR_INTERRUPT_TRANSFER	55 53 42 43 60 62 75 82	
⊞	12	in up	0x83	0.093	BULK_OR_INTERRUPT_TRANSFER	55 53 42 53 60 62 75 82	0x00000000
⊞	13	out down	0x04	0.093	BULK_OR_INTERRUPT_TRANSFER	55 53 42 43 60 62 75 82	
⊞	13	out up	0x04	0.093	BULK_OR_INTERRUPT_TRANSFER	-	0x00000000

Figure 44: URB packets captured by the USB sniffer

We can see that at the beginning the USB modem (with an assigned address of “USB\Vid_12d1%Pid_1003&Rev_0000&MI_02”, marked by the blue rectangle) performs several initializing configurations before starting “BULK_OR_INTERRUPT_TRANSFER”, including “GET_DESCRIPTOR_FROM_DEVICE”, “SELECT_CONFIGURATION”, “SELECT_INTERFACE”, “CLASS_INTERFACE”, and “CONTROL_TRANSFER”, with Function number of 000b, 0000, 0001, 001b, 0008 (shown in Figure 45 to Figure 49).

We found the device used three endpoints during the whole process: “0x00”, “0x04”, and “0x83” (marked by the red rectangle in Figure 44). All USB devices support endpoint 0x00 when powered up. This endpoint is the target of the default pipe. After the attachment of a device has been detected, the software uses endpoint 0x00 to initialize the device, perform generic (i.e. non device-specific) configuration, and obtain information about the other endpoints provided by the device. The other two endpoints are defined in the “SELECT_CONFIGURATION” packet, as shown in Figure 46, the device set up one interface (with number 0x02) and two bulk pipes (with endpoint address 0x83 and 0x04) for the following transfer. The “SELECT_INTERFACE” packet set up basic configurations for the interface and two pipes, such as maximum packet size, maximum transfer size, pipe flags, etc., as shown in Figure 47.

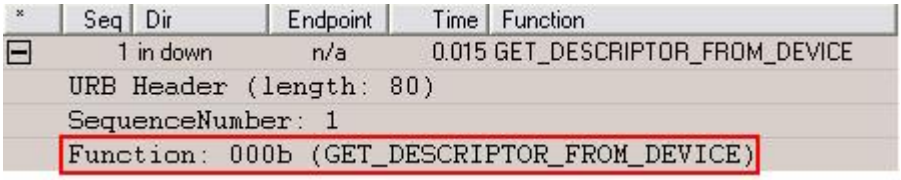


Figure 45: URB packet – GET_DESCRIPTOR_FROM_DEVICE

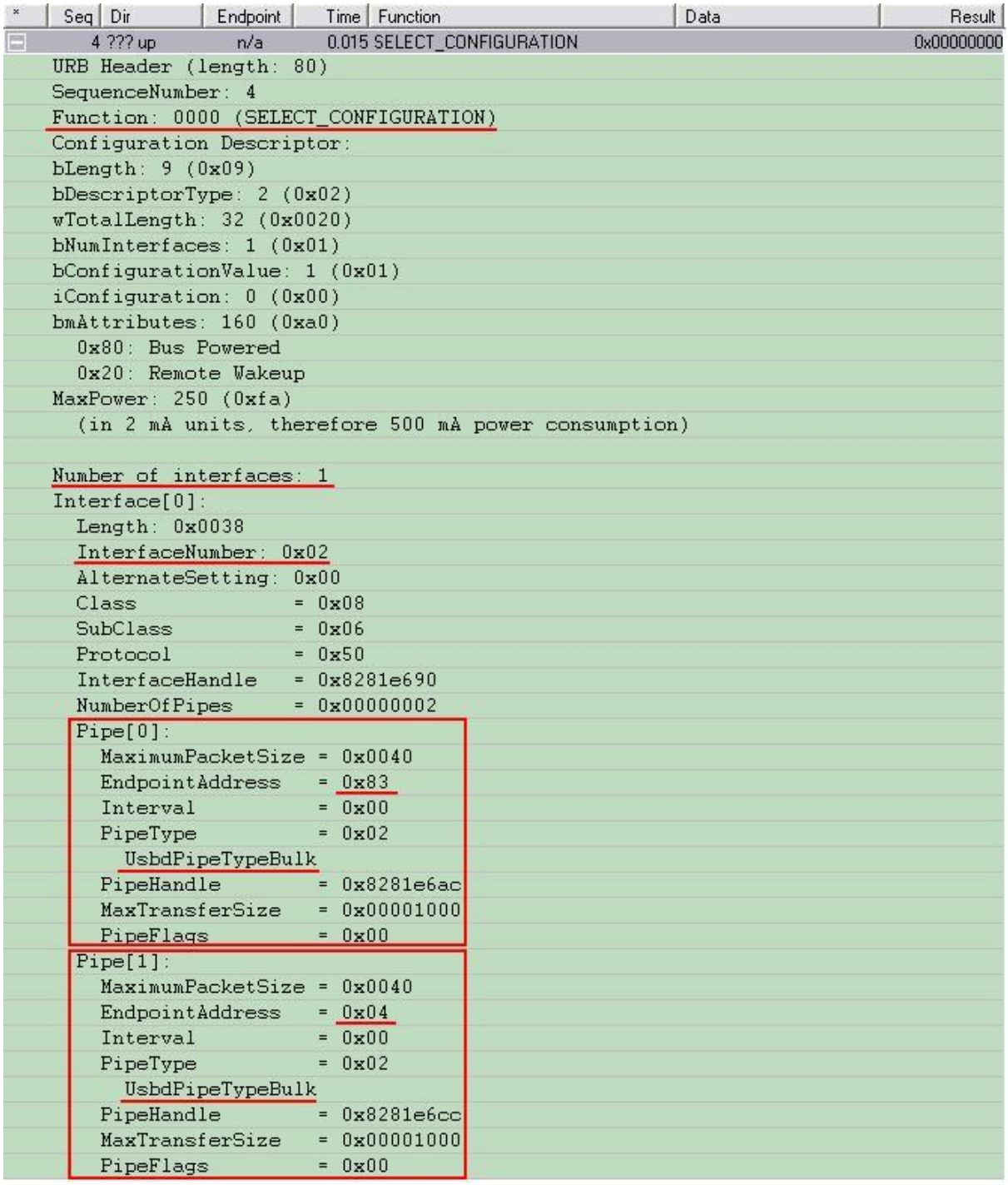


Figure 46: URB packet – SELECT_CONFIGURATION

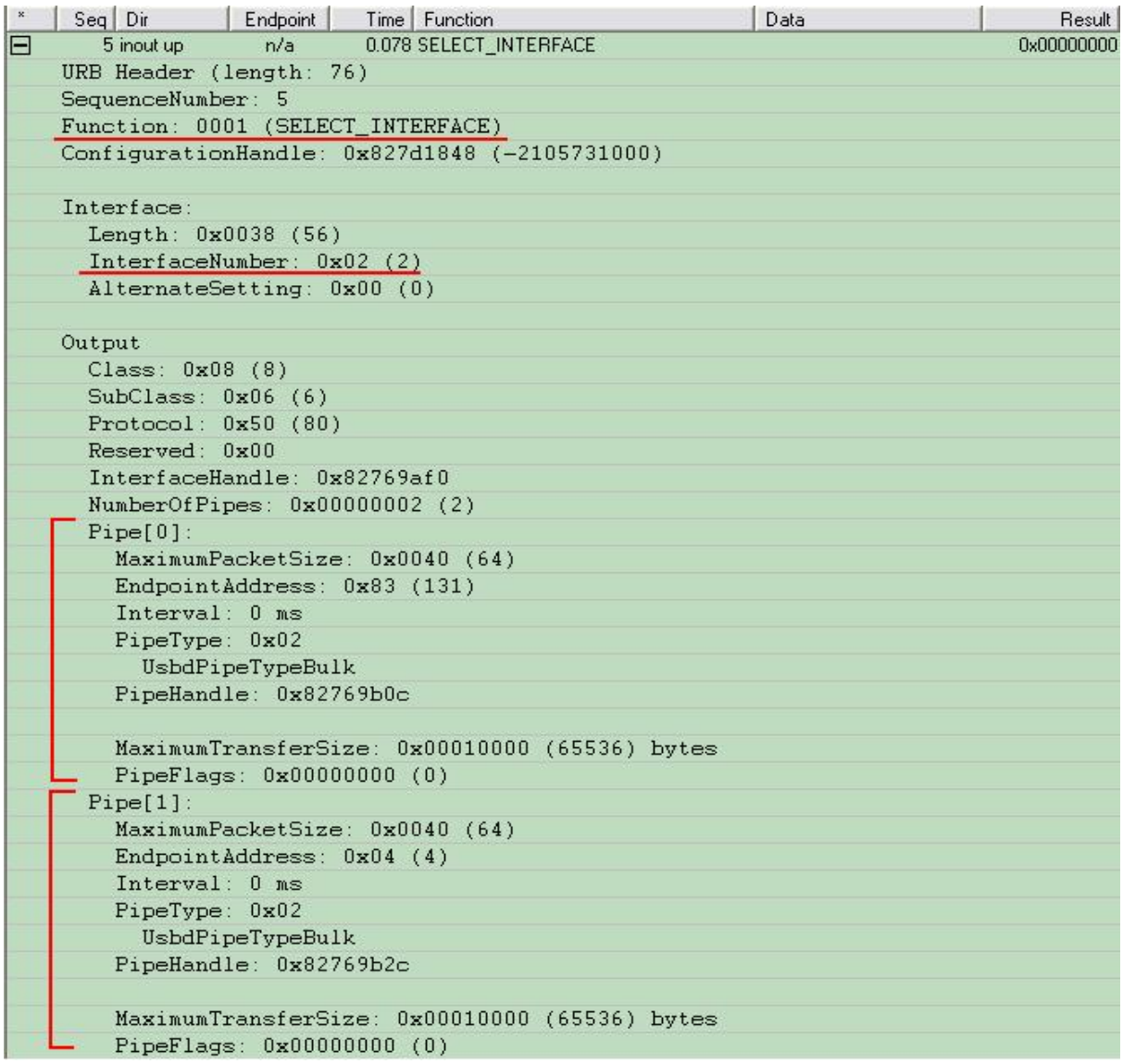


Figure 47: URB packet – SELECT_INTERFACE

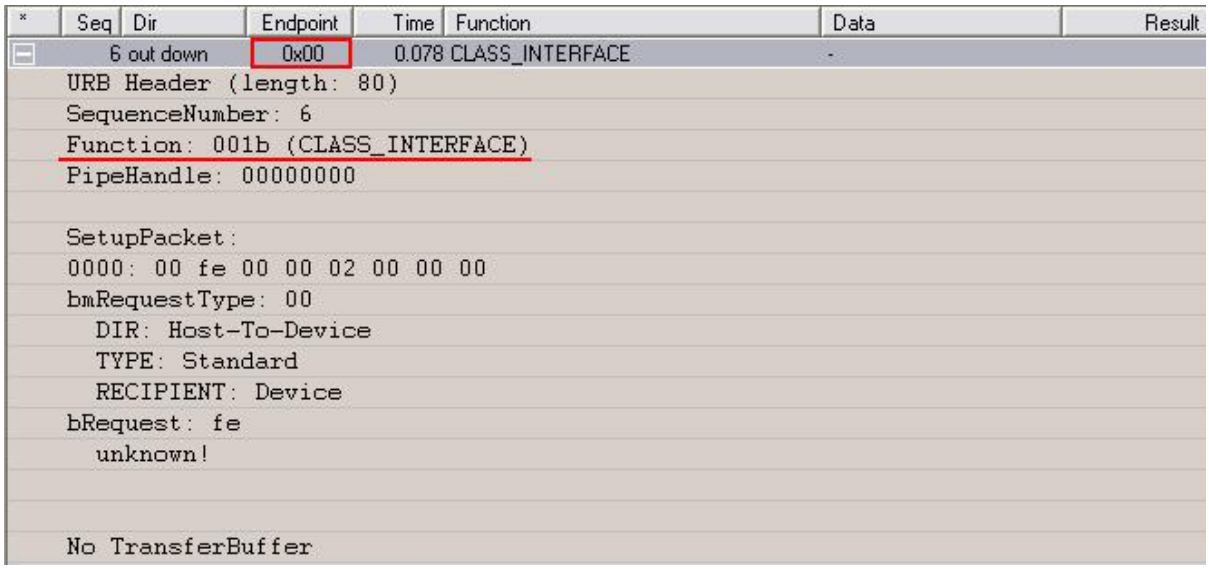


Figure 48: URB packet – CLASS_INTERFACE

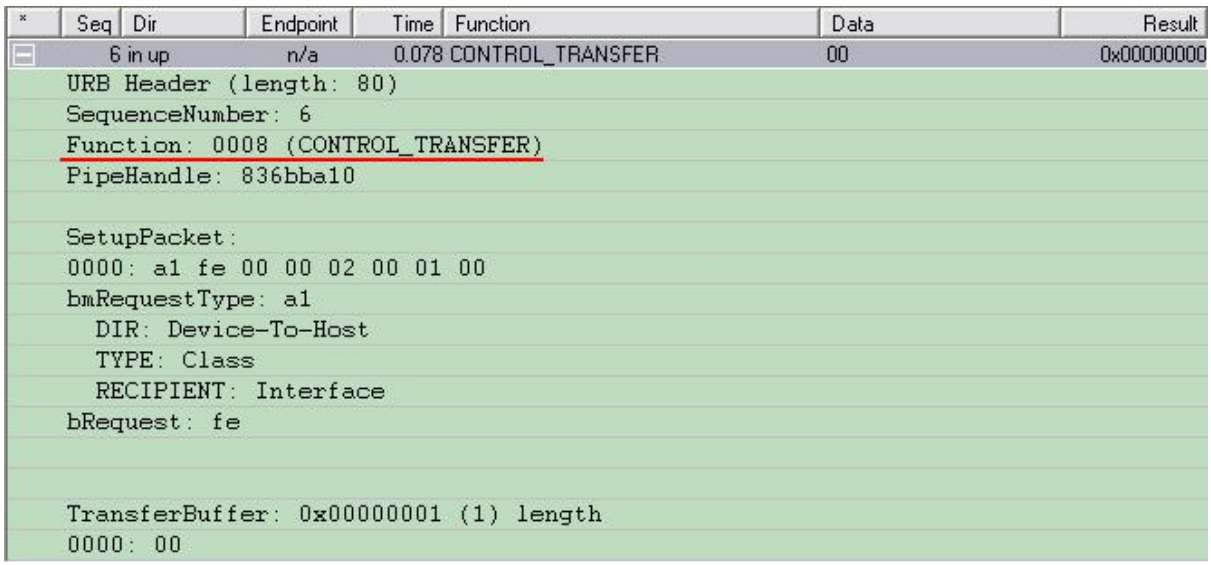


Figure 49: URB packet – CONTROL_TRANSFER

After initializing the device, the data transfer between the device and the host will start using pipe 0x83 and 0x04 with BULK_OR_INTERRUPT_TRANSFER packets until we end the device. Unfortunately, we can not go into the packets to check if there is transferring a UDP packet simply by a USB sniffer. While the Wireshark can not recognize the 3G network interface until the USB modem has been assigned an IP address, it only can capture PPP control messages including a PPP termination request and a PPP termination ACK besides UDP packets, as shown in Figure 50 and Figure 51.

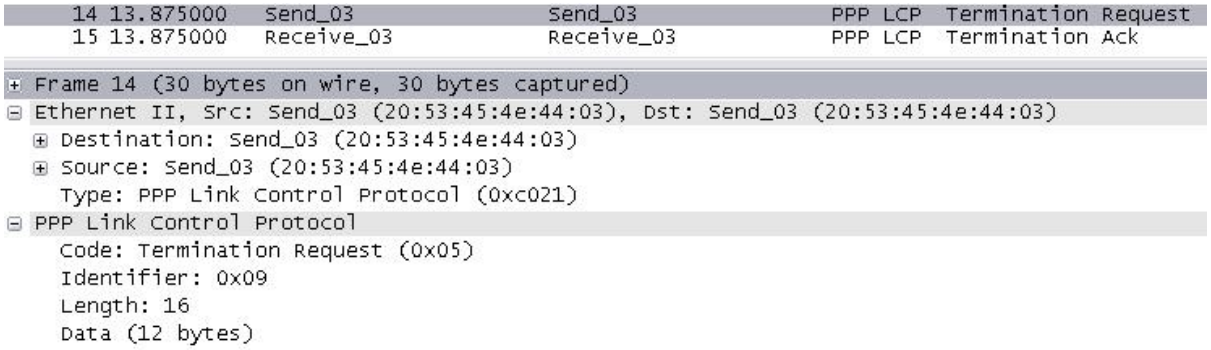


Figure 50: PPP termination request captured by Wireshark

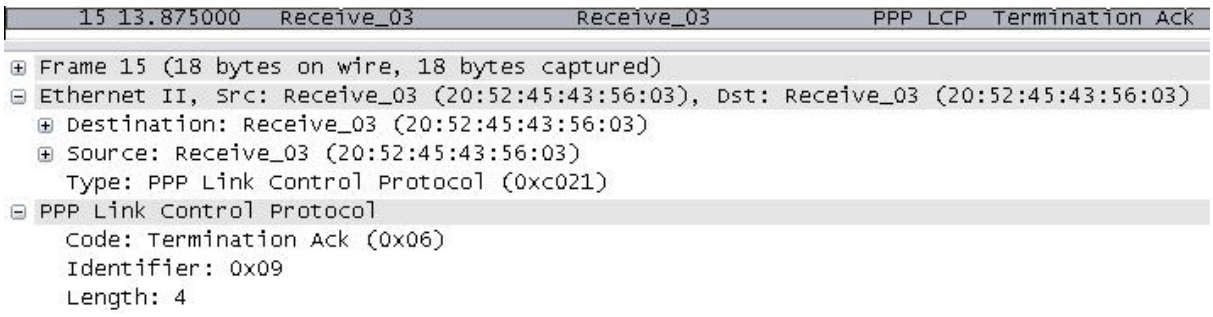


Figure 51: PPP termination ACK captured by Wireshark

5.1.4 Throughput analysis

Network throughput describes the *average* rate of successful message delivery over a communication channel. Wireshark performs throughput analysis by calculating the average number of packets sent per second, average bytes sent per second, and average Mbps (see also section 3.4.2 , Figure 19). In the pre-measurements we chose one minute of traffic as 1 unit; this traffic is approximately 2000 packets with each packet having 1024 bytes of data. For the ease of calculation, we utilize 2000 packets (each packet has 1024 bytes of data) as 1 unit of sample. There was a little difference between the results as measured at the client side and at the server side. This is because the time delay varies in different packets.

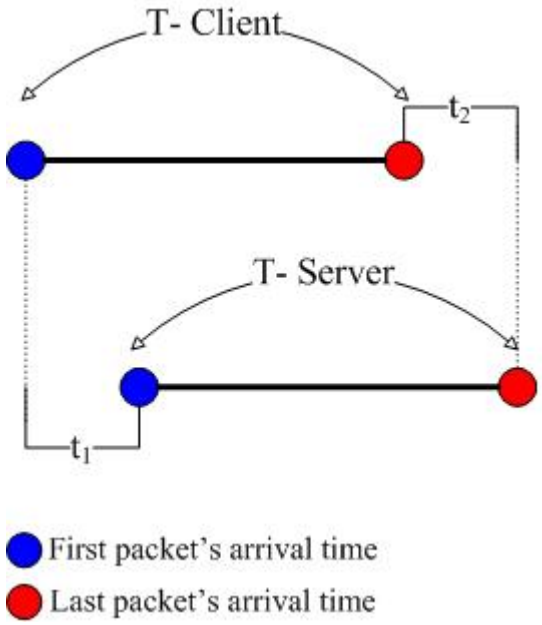


Figure 52: Comparison of traffic arrival time between client and server

In Figure 52, T-Client is the time from when the first packet arrives at the client's interface to when the last packet arrives (during this time 2000 packets have been sent), while T-Server represents a same period on the server. The time delay of sending the first packet from the client to the server is t_1, while the time from when the last packet arrives at the client's interface to when it arrives at the server's interface is represented by t_2. If t_1 is not equal to t_2, T-Client will not equal to T-Server. In this way, same bytes of data will be transferred via the client's and the server's interfaces within different periods of time. Therefore, the throughput results will be different on the client and the server, but the difference is a very small value that we can ignore it and now going to concentrate on the client side. The experimental results of both blocking mode and non-blocking mode are shown in Table 5-2 and Table 5-3.

Table 5-2: Experimental throughput (blocking mode)

Sample Number	Avg.packets/s	Avg.bytes/s	Avg.Mbps
1	34.651	36831.329	0.295
2	36.775	39115.380	0.313
3	31.306	33152.000	0.265
4	34.289	36430.311	0.291
5	36.766	38976.454	0.312
6	35.651	37786.571	0.302
7	36.535	38859.869	0.311
8	38.434	40771.821	0.326
9	39.324	41919.508	0.335
10	38.173	40495.198	0.324
11	33.204	35395.071	0.283
12	37.475	39868.213	0.319
Average	36.049	38300.144	0.306

Table 5-3: Experimental throughput (non-blocking mode)

Sample Number	Avg.packets/s	Avg.bytes/s	Avg.Mbps
1	22.261	23730.087	0.190
2	21.930	23377.455	0.187
3	56.889	60643.556	0.485
4	55.018	58648.702	0.469
5	61.440	65495.040	0.524
6	54.857	58477.714	0.468
7	60.308	64288.000	0.514
8	61.440	65495.040	0.524
9	61.490	65548.549	0.524
10	56.889	60643.556	0.485
11	61.440	65495.040	0.524
12	59.077	62976.000	0.504
Average	52.753	56234.895	0.450

Average throughput in blocking mode is: 36.049 packets/s, 38300.144 bytes/s, and 0.306 Mbps. While average throughput in non-blocking mode is: 52.753 packets/s, 56234.895 bytes/s, and 0.450 Mbps. Besides the average value, we also consider about the standard deviation, which can indicates the variability or dispersion of the data. A low standard deviation indicates that all of the data points are very close to the mean, while a high standard deviation indicates that the data are “spread out” over a large range of values. Therefore, the lower standard deviation is, the more reliable the experimental results will be.

Consider two groups of samples for blocking mode (E1) and non-blocking mode (E2) of the average Mbps value, thus the sample group becomes:

E1 = {0.295,0.313,0.265,0.291,0.312,0.302,0.311,0.326,0.335,0.324,0.283,0.319,0.306};
E2 = {0.190,0.187,0.485,0.469,0.524,0.468,0.514,0.524,0.524,0.485,0.524,0.504,0.450};

As the average of E1 and E2 are 0.306 and 0.450 Mbps for blocking mode and non-blocking mode respectively, we can use the standard deviation formula (see also section 3.2.2):

$$s = \sqrt{\frac{\sum_{i=1}^{n} (s_i - \overline{s})^2}{n}}$$ (Formula 3.2) [31],

For group E1, substitute the twelve samples S_1 to S_{12} into Formula 3.2:

$$S_{blocking} = \sqrt{\frac{\sum_{i=1}^{12} (S_i - 0.306)^2}{12}} \approx 0.020$$;

For group E2:

$$S_{non-blocking} = \sqrt{\frac{\sum_{i=1}^{12} (S_i - 0.450)^2}{12}} \approx 0.124$$;

Therefore the average throughput in blocking mode is 0.306 Mbps with a standard deviation of 0.020; and the average throughput in non-blocking mode is 0.450 Mbps with a standard deviation of 0.124. Throughput in non-blocking mode is higher than in blocking mode, since if space is not available at the sending socket to hold the message to be transmitted, non-blocking mode will simply quit without blocking “sendto” function to wait until space is available, where blocking mode decreases its rate.

5.1.5 Time delay analysis

Time delay indicates the period of time from when a packet is sent out (to the 3G modem) until it is received by the destination (the server). Figure 53 shows some sample statistics from captured traffic.

	A	D	E	F	G	H	I	L	M	N	O	P
1	Packet No.	Time	Source	Dest	Pro	Info	Packet No.	Time	Source	Dest	Pro	Delay
2	1	21.121022	83.188.	130.	UDP	Sour	1	21.268728	83.188.	130.	UDP	0.147706
3	2	21.121022	83.188.	130.	UDP	Sour	2	21.288396	83.188.	130.	UDP	0.167374
4	3	21.136647	83.188.	130.	UDP	Sour	3	21.308814	83.188.	130.	UDP	0.172167
5	4	21.136647	83.188.	130.	UDP	Sour	4	21.340914	83.188.	130.	UDP	0.204267
6	5	21.136647	83.188.	130.	UDP	Sour	5	21.41754	83.188.	130.	UDP	0.280893
7	6	21.136647	83.188.	130.	UDP	Sour	6	21.41759	83.188.	130.	UDP	0.280943
8	7	21.136647	83.188.	130.	UDP	Sour	7	21.417639	83.188.	130.	UDP	0.280992
9	8	21.136647	83.188.	130.	UDP	Sour	8	21.428722	83.188.	130.	UDP	0.292075
10	9	21.152272	83.188.	130.	UDP	Sour	9	21.448691	83.188.	130.	UDP	0.296419
11	10	21.152272	83.188.	130.	UDP	Sour	10	21.468059	83.188.	130.	UDP	0.315787
12	11	21.246022	83.188.	130.	UDP	Sour	11	21.497862	83.188.	130.	UDP	0.25184
13	12	21.246022	83.188.	130.	UDP	Sour	12	21.518079	83.188.	130.	UDP	0.272057

Figure 53: Sample statistics from captured traffic

Column A to column H are traffic captured from the client's interface, while column I to column O is traffic captured from the server's interface. As we have put a sequence number into the UDP packets, we can sort the packets by sequence number and find out the lost packets. Column D indicates the time when a packet was captured on the client's network interface and column L indicates the time it was captured on the server's network interface, thus we can calculate the value of time delay by "L - D". The results of time delay are displayed in column P.

Using the graph function, we can obtain a scatter plot of the values in column P; see Figure 54 and Figure 55.

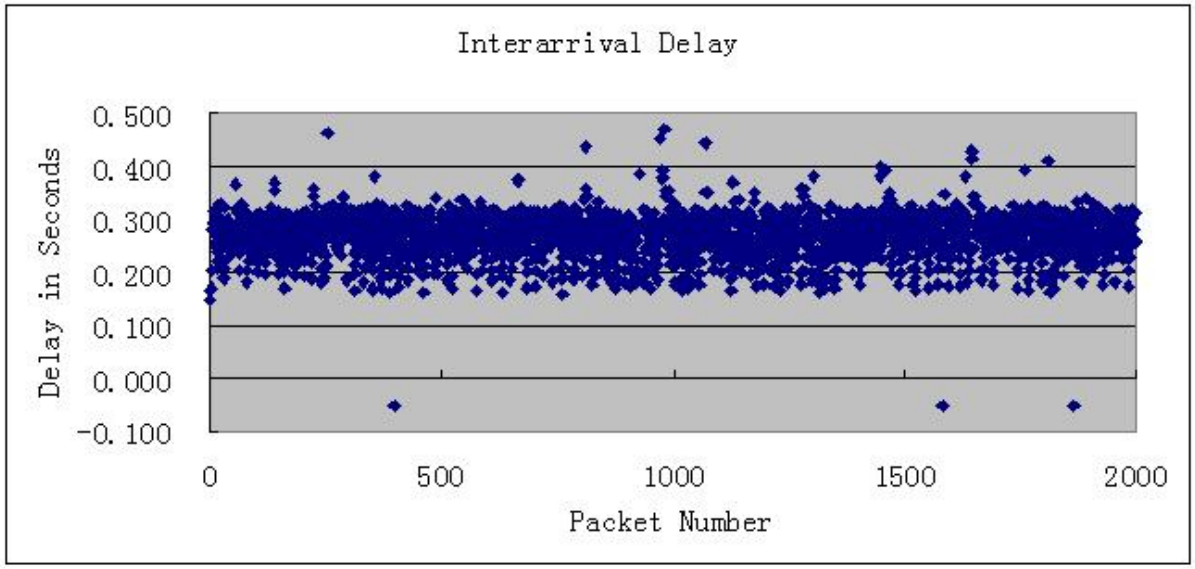


Figure 54: Scatter diagram of time delay – UDP traffic

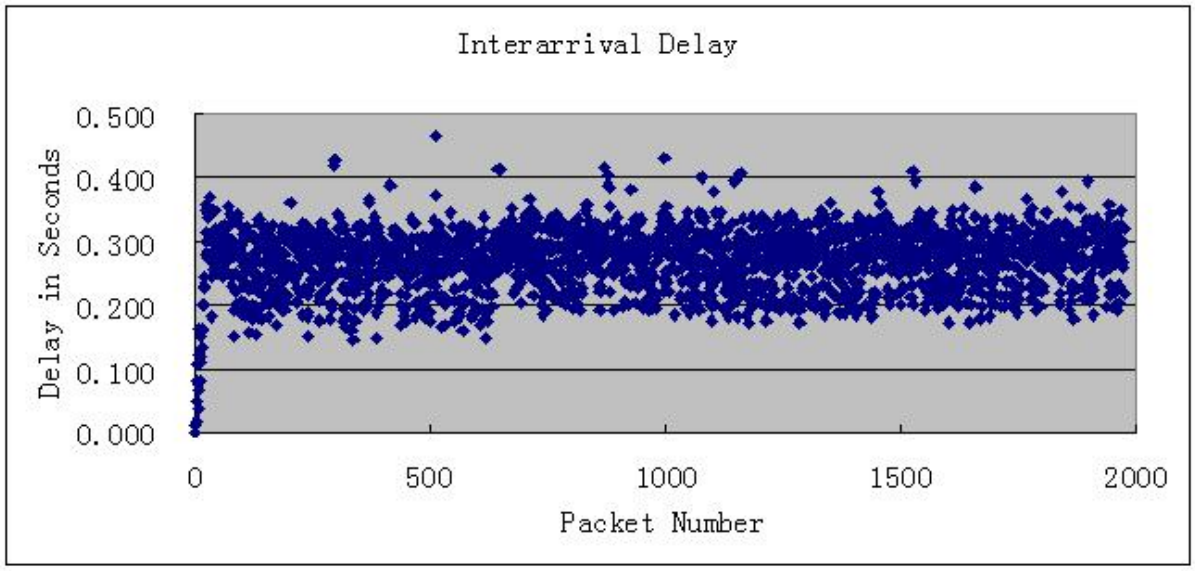


Figure 55: Scatter diagram of time delay – TCP traffic

In the TCP traffic example, the first several packets have shorter time delay (during connection establishment), but then the delay increases to higher values since TCP flow control limits the rate at which segments are sent. In contrast, UDP experiences a reduced delay, since UDP does not include any reliability mechanisms hence does not wait for acknowledgements. However packet loss occurs during transmitting UDP traffic. In Figure 54, we can see that there are three packets were lost (the spots with values below zero) among the total 2000 packets. We selected a group of twelve values of time delay for UDP traffic for analysis, see Table 5-4.

Table 5-4: Experimental data of time delay for UDP traffic

Sample number	Time delay (in seconds)
1	0.294
2	0.290
3	0.287
4	0.332
5	0.301
6	0.278
7	0.279
8	0.272
9	0.278
10	0.270
11	0.269
12	0.264
Average	0.284

And we can also set up a group of twelve items of value of time delay as a sample congregation:

$E_{\text{-TimeDelay}}$ = {0.294,0.290,0.287,0.332,0.301,0.278,0.279,0.272,0.278,0.270,0.269,0.264};

Then use formula 3.3 to calculate standard deviation:

$$S_{-TimeDelay} = \sqrt{\frac{\sum_{i=1}^{12}(S_i - 0.284)^2}{12}} \approx 0.019;$$

Therefore the result comes out: average time delay is 0.284 seconds, with standard deviation of 0.019.

6. Conclusions and future work

USB attached 3G modems are being used by mobile users to connect to the Internet by more and more people all over the world. As we know, the speed of a wireless network depends on several factors, which can reduce the actual network speed to below the maximum theoretical value (in this thesis 7.2 Mbps). Thus the most important question that customers ask is if the real network speed which they can achieve is the advertised "7.2 Mbps" or not.

According to the experimental data analysis in chapter 5, the final results we obtained seem to answer the above question as "not". The average throughput in blocking mode is 0.306 Mbps with a standard deviation of 0.020; and the average throughput in non-blocking mode is 0.450 Mbps with a standard deviation of 0.124, which is far from the announced "7.2 Mbps". It is true that the experimental results comes from the "worst" conditions during the peak traffic period, but represent the practical values and most likely close to real values that the customer will experience. We also collected data during the 3G network's peak hour around 18:00, when the average throughput was around 0.354 Mbps; and data during off-peak hours, when the average throughput was around 0.358 Mbps.

These uplink rates are indeed very low, so we also perform some measurements on both uplink and downlink via a speed-testing website: http://www.bredbandskollen.se. In this case, we got uplink rate of 0.35 Mbps, downlink rate of 2.15 Mbps, and time delay of 120ms. The download speed of a USB attached 3G network is much higher than the upload speed, and having less time delay.

There is a survey of iPhone 3G network performance [48], revealing network weaknesses: in this survey, participants in Australia reported the slowest average 3G download speeds of about 0.759 Mbps, and Australian carriers Optus and Virgin users reported the slowest speeds of about 0.390 Mbps on average; while users of U.S. carrier AT&T, which tied for third with Telstra, Telia and Softbank, reported average download speeds of roughly 0.990 Mbps. All of these results indicate that we would not achieve an advertised high speed (in this case 7.2 Mbps) when connecting to the Internet via a 3G network interface, but a much lower rate.

However, not achieving advertised high speed has not reduced people's enthusiasm for using 3G modems. Having a 3G USB modem enables customers to work on the move. Whether a 3G modem satisfies the customer is determined by customer themselves, depending on what service they use on the network, if they are concern more about mobility or bandwidth, etc.

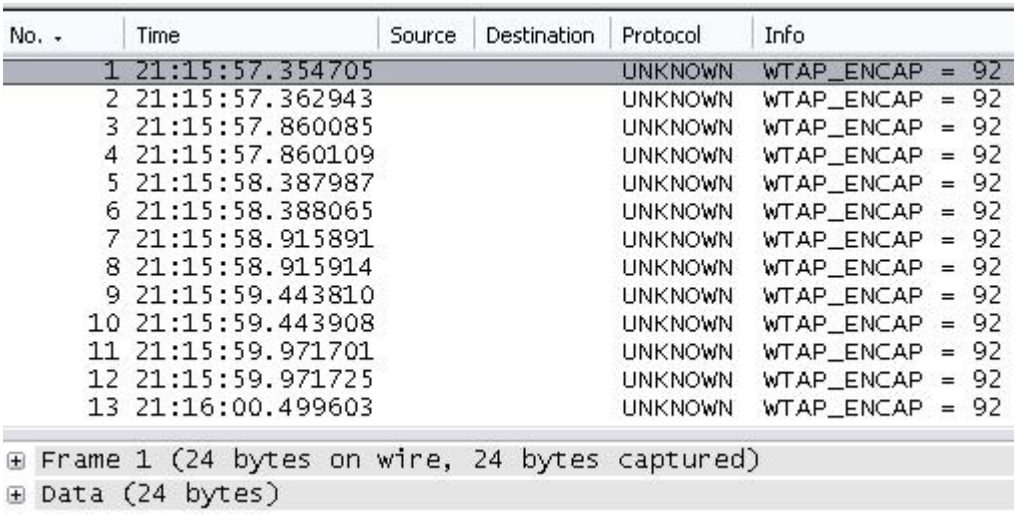


Figure 56: USB traffic captured by Wireshark on a Linux machine
(From an attached Pocket PC)

Future work: In this thesis, we attached the USB modem to a laptop running Windows system. For future analysis the USB modem can be attached to a computer running another operating system such as Linux. The problem will be how to use capture tools to monitor traffic in that case. Since Wireshark cannot recognize the USB attached network interface, another configuration will be needed. However, once a suitable configuration has been done on a Linux machine, then the traffic will be captured in a raw format, as shown in Figure 56. Additional changes have to be made in Wireshark to enable it to analysis traffic to and from a USB attached network interface. This way we will be able to go deep into the USB traffic that we discussed in section 5.1.3 to see the relationship between the traffic captured on the USB interface and on a 3G modem.

References

[1] Tele2 advertisement, Tele2 AB, http://www.tele2.se/mobilt-bredband.html, last visited 2009-3-16

[2] Modem, Wikipedia, http://en.wikipedia.org/wiki/Modem , last visited 2009-2-27

[3] Wikipedia, Personal Computer Memory Card International Association, http://en.wikipedia.org/wiki/PCMCIA, last visited 2009-2-27

[4] PCMCIA, http://www.webopedia.com/TERM/P/PCMCIA.html, last visited 2009-2-26

[5] Tele2 advertisement, Tele2 AB, http://www.tele2.se/mobilt-bredband.html, last visited 2008-12-20

[6] 3G USB Modem, http://www.vodafonebusinessshop.co.uk/3GUsbModem.html , last visited 2008-12-20

[7] Qiang Fu, Building models of Wireless Local Area Network coverage, Master's Thesis, Royal Institute of Technology, 2007, http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/070124-Qiang_Fu-with-cover.pdf, last visited 2009-2-25

[8] Broadband penetration, OECD, http://www.oecd.org/document/54/0,3343,en_2649_34225_38690102_1_1_1_1,00.html, last visited 2009-02-28.

[9] http://www.3g.co.uk/PR/June2005/1591.htm, last visited 2009-2-27

[10] "1Q08 Mobile Forecast: Sweden, 2007-2010", IE Market Research Corp., January 2008

[11] ITU Internet Reports 2002: Internet for a Mobile Generation, International Communication Union, 2002

[12] GSM/3G Networks - November 2007, the Global mobile Suppliers Association, published on http://www.gsacom.com, last visited 2009-2-27

[13] Third Generation networks, http://www.itu.int/osg/spu/imt-2000/technology.html#Cellular%20Standards%20for%20the%20Third%20Generation, last visited 2009-02-27

[14] 3G, http://www.webopedia.com/TERM/3/3G.html, last visited 2009-2-27

[15] Wikipedia, 3G, http://en.wikipedia.org/wiki/3G, last visited 2009-2-27

[16] ITU Internet Reports 2002: Internet for a Mobile Generation, adapted from European Information Technology Observatory 2002, International Communication Union, 2002

[17] 3G network coverage of Sweden, TeliaSonera Sverige, http://www.telia.se/privat/produkter_tjanster/mobilt/tackningskartor/tackningskartahomerun/tackningskarta-data-internet.page, last visited 2009-3-16

[18] The Telia Trend Report 2007 (in Swedish) can be downloaded from TeliaSonera Press Archive, http://www.teliasonera.se

[19] News from http://www.3g.co.uk/PR/April2007/4555.htm, TeliaSonera Launches Turbo 3G in Sweden, 13th April 2007, last visited 2009-2-27

[20] Xiao Lin, Internet Protocol Network Performance Analysis and Evaluation Norm, 20th July 2004, http://www.ctiforum.com/forum/2004/07/forum04_0736.htm, last visited 2009-2-27

[21] http://sourceforge.net/projects/usbsnoop, last visited 2009-2-2

[22] Wireshark, http://www.wireshark.org, last visited 2009-2-2

[23] Sniffer Pro, http://www.packet-sniffer.net/sniffer-pro.htm, last visited 2009-2-2

[24] http://www.pcausa.com/Utilities/UsbSnoop/default.htm, last visited 2009-2-2

[25] Wireshark, http://en.wikipedia.org/wiki/Wireshark, last visited 2009-2-2

[26] About Wireshark, http://www.wireshark.org/about.html, last visited 2009-2-13

[27] Universal Serial Bus, http://en.wikipedia.org/wiki/USB#USB_2.0, last visited 2009-2-13

[28] Wikipedia, Throughput, http://en.wikipedia.org/wiki/Throughput#Peak_measured_throughput, last visited 2009-02-24

[29] Wikipedia, Network delay, http://en.wikipedia.org/wiki/Network_delay, last visited 2009-02-24

[30] Tele2, http://www.tele2.se/mobilt-bredband-maxi.html, last visited 2009-02-24

[31] Wikipedia, Standard deviation, http://en.wikipedia.org/wiki/Standard_Deviation, last visited 2009-02-24

[32] How to determine sample size, http://www.clearmarket.cn/method/samplesize.html, last visited 2009-02-24

[33] Wikipedia, Transmission Control Protocol, http://en.wikipedia.org/wiki/ACK_(TCP), last visited 2009-02-24

[34] John Nagle, Congestion Control in IP/TCP Internetworks, IETF, RFC 896, 1st June 1984

[35] TCP Algorithm, http://cities.lk.net/tcp.html, last visited 2009-02-25

[36] Steve Makofsky, Pocket PC Network Programming, published ISBN 0-321-13352-8, 2003

[37] Traffic Analysis for 20-ke5-GW, http://mrtg1.lan.kth.se/rtrdata/ke5.gw.kth.se_20.html, last visited 2009-02-25

[38] Ming Li, 3G Update, VP, Southeast Asia and Pacific Qualcomm International, 24th February 2005,

[39] Simple random Sample, http://www.coventry.ac.uk/ec/~nhunt/meths/random.html, last visited 2009-02-26

[40] Simple random Sample, http://en.wikipedia.org/wiki/Simple_random_sample, last visited 2009-2-26

[41] Universal Serial Bus, Wikipedia, http://en.wikipedia.org/wiki/USB, last visited 2009-2-27

[42] Xiao Hu, Lianchao Zhang, Properties and Application of USB2.0 Controller CY7C68013, http://www2.minitos.com/article/sort010/info-3051.html, last visited 2009-2-27

[43] USB Background, Total Phase, http://www.totalphase.com/support/kb/10047, last visited, 2009-2-27

[44] USB Isochronous Transfer, MSDN, http://msdn.microsoft.com/en-us/library/ms790482.aspx, last visited 2009-2-27

[45] _URB_HEADER, MSDN, http://msdn.microsoft.com/en-us/library/ms793351.aspx, last visited 2009-2-27

[46] _URB_BULK_OR_INTERRUPT_TRANSFER, MSDN, http://msdn.microsoft.com/en-us/library/ms793345.aspx, last visited 2009-2-27

[47] Point-to-Point Protocol, Wikipedia, http://en.wikipedia.org/wiki/Point-to-Point_Protocol, last visited 2009-2-27

[48] Brian X. Chen, Wired.com's iPhone 3G Survey Reveals Network Weaknesses, August 25, 2008, http://blog.wired.com/gadgets/2008/08/global-iphone-3.html, last visited 2009-03-22

Appendix A: I/O Graphs

The IO Graphs on the USB attached 3G network interface are shown in the following figures.

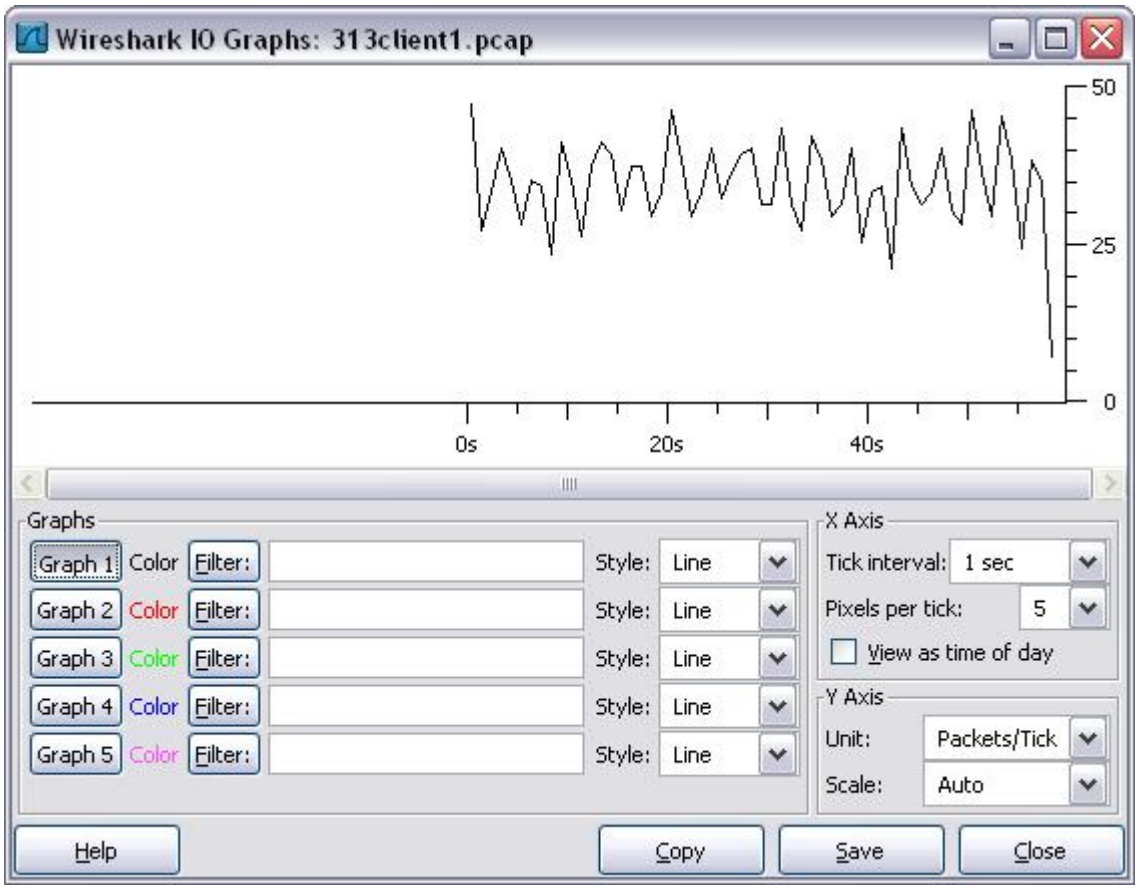


Figure 57: Sample 1 – IO Graph, 2009-3-13, 14:04

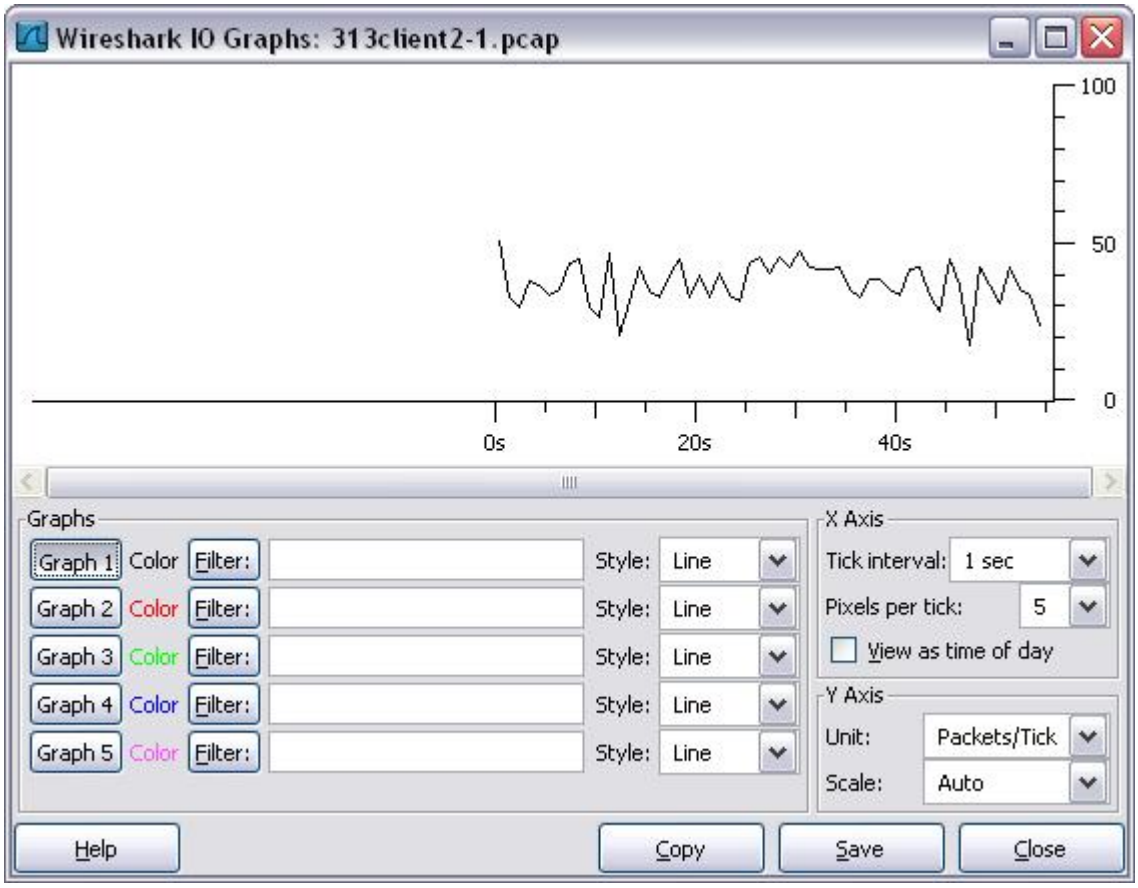


Figure 58: Sample 2 – IO Graph, 2009-3-13, 14:14

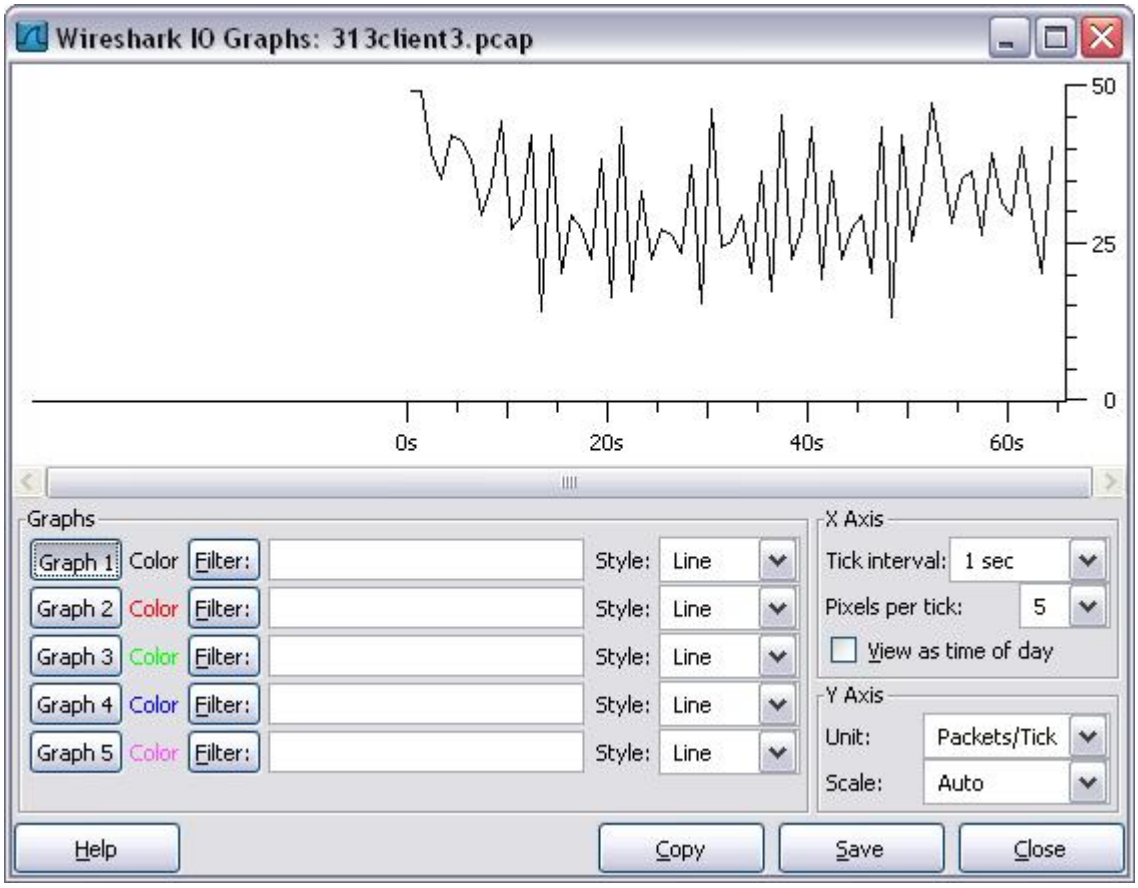


Figure 59: Sample 3 – IO Graph, 2009-3-13, 14:23

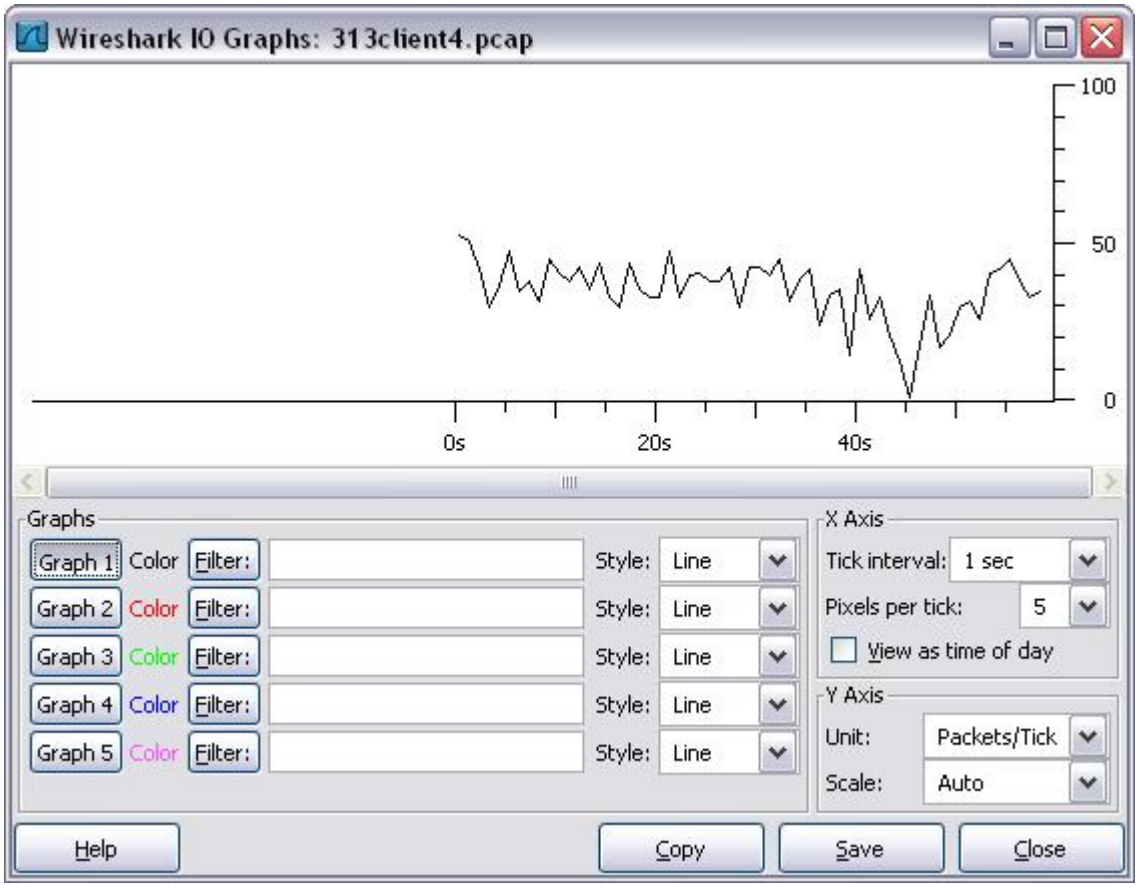


Figure 60: Sample 4 – IO Graph, 2009-3-13, 14:32

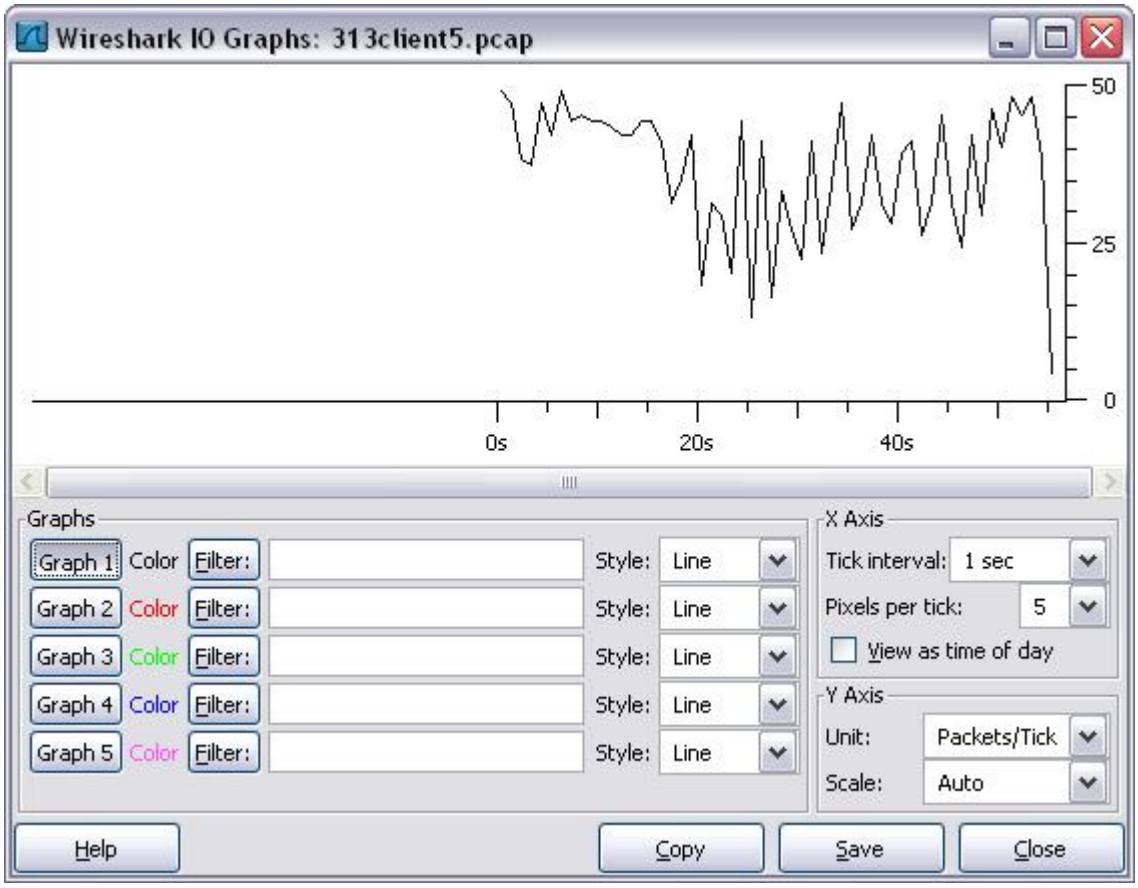


Figure 61: Sample 5 – IO Graph, 2009-3-13, 14:43

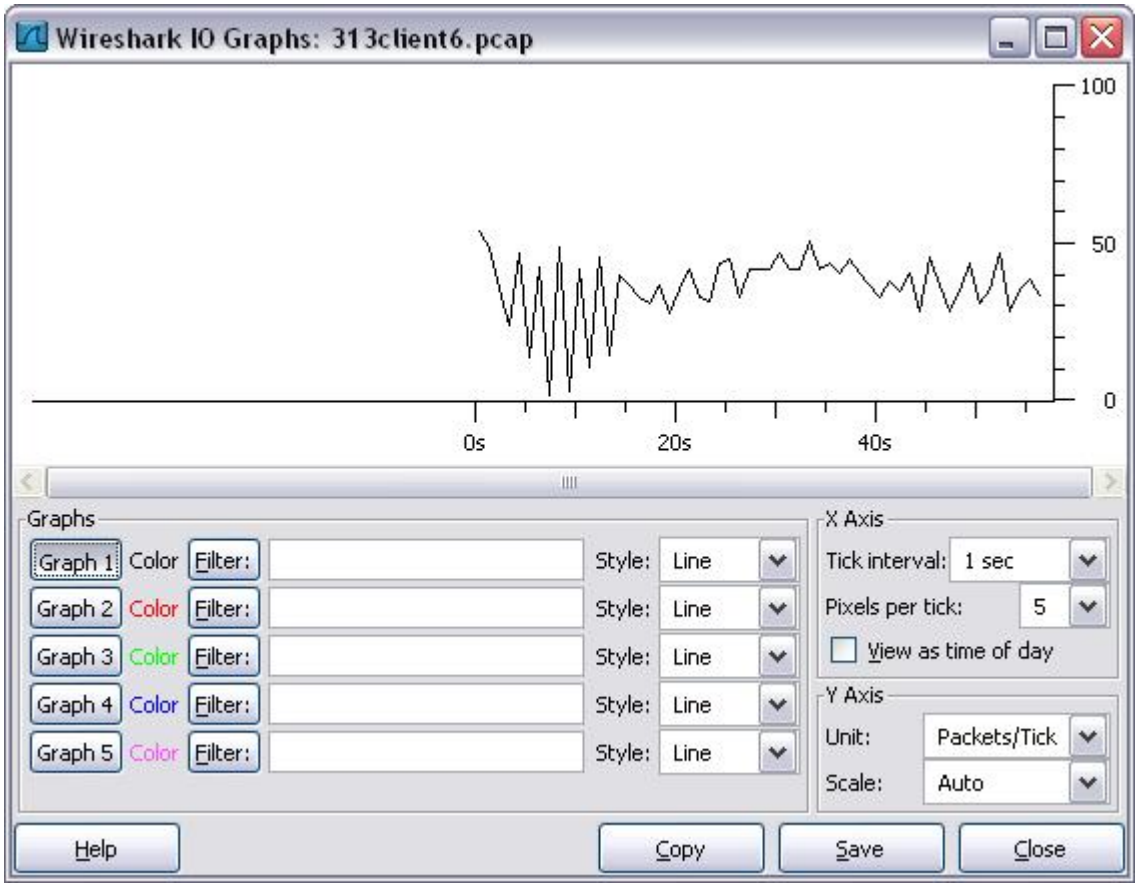


Figure 62: Sample 6 – IO Graph, 2009-3-13, 14:52

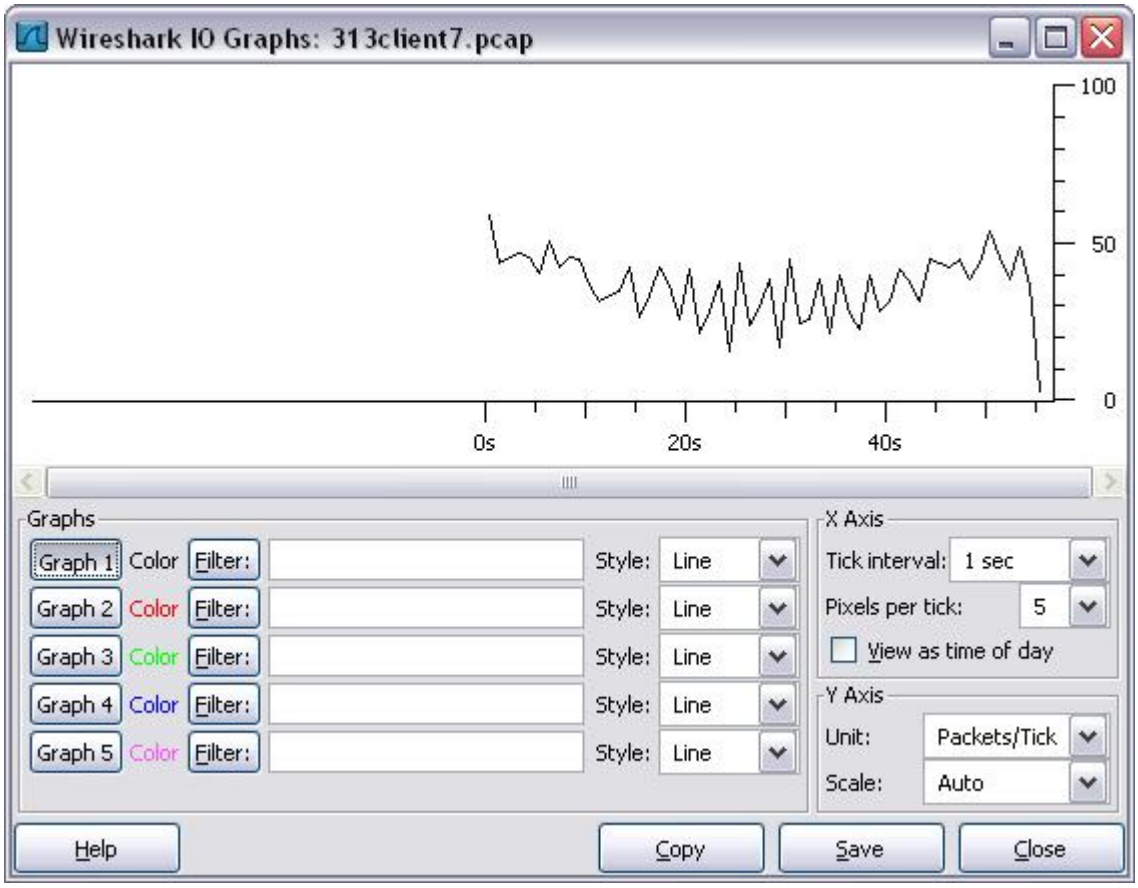


Figure 63: Sample 7 – IO Graph, 2009-3-13, 15:02

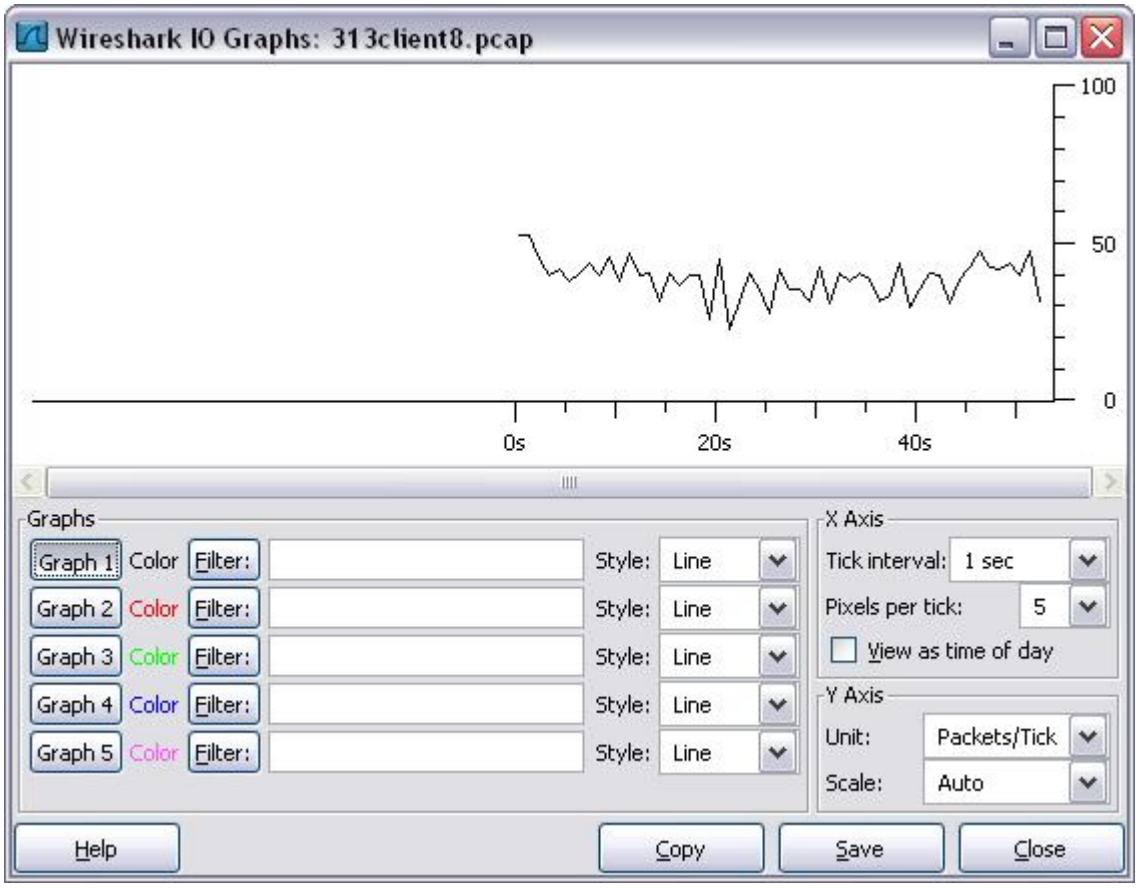


Figure 64: Sample 8 – IO Graph, 2009-3-13, 15:12

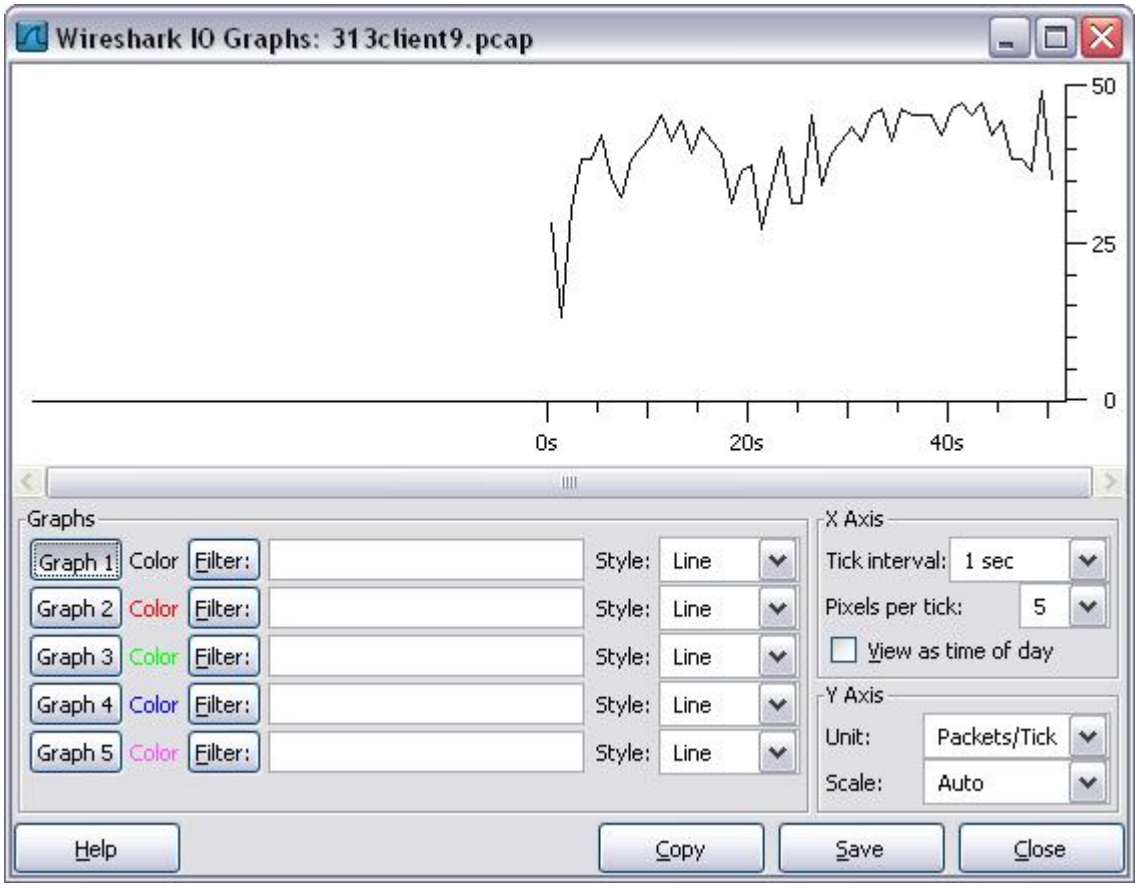


Figure 65: Sample 9 – IO Graph, 2009-3-13, 15:21

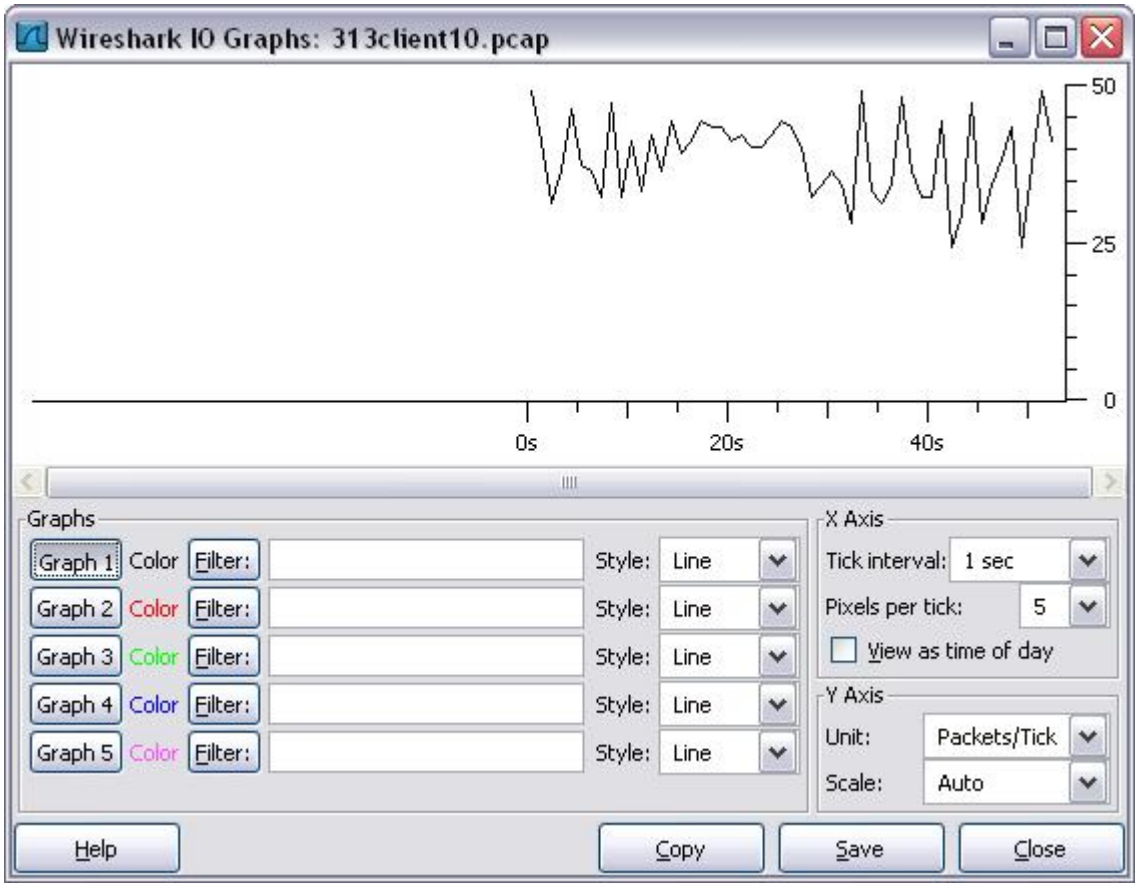


Figure 66: Sample 10 – IO Graph, 2009-3-13, 15:32

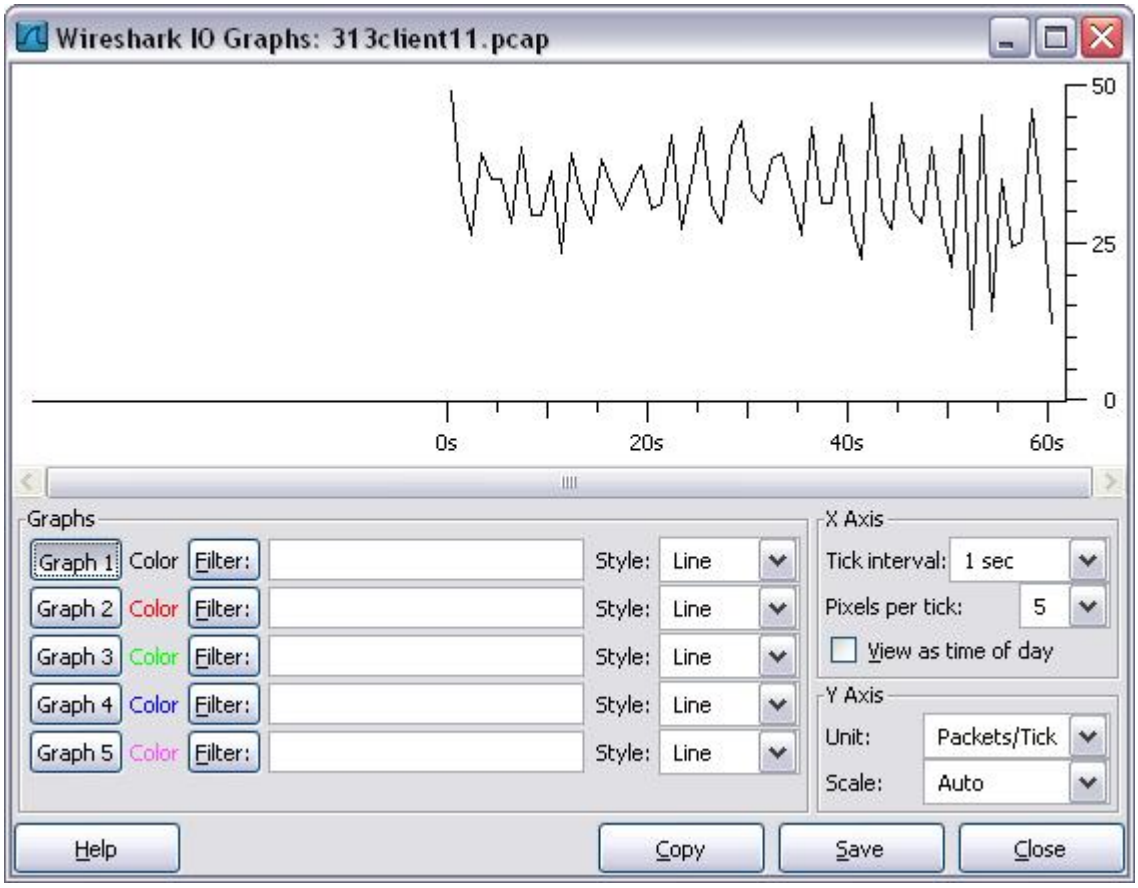


Figure 67: Sample 11 – IO Graph, 2009-3-13, 15:41

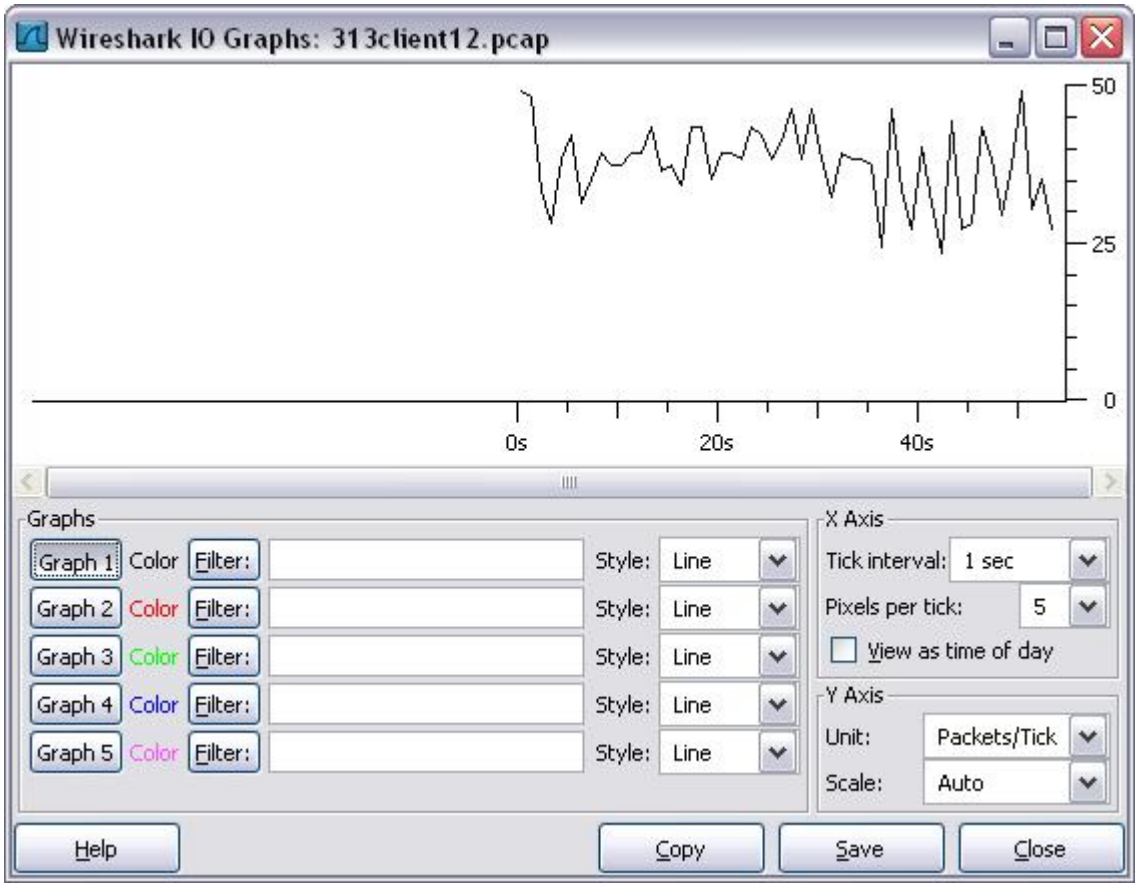


Figure 68: Sample 12 – IO Graph, 2009-3-13, 15:53

Appendix B: Scatter plot of time delay

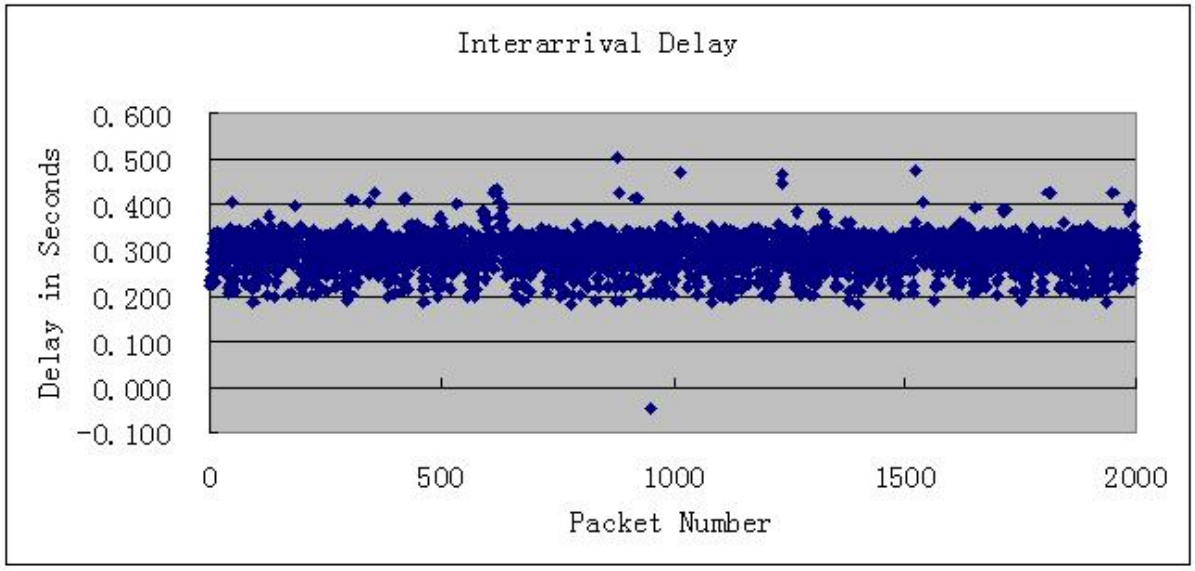


Figure 69: Sample1 – Interarrival Time Delay, 2009-3-13, 14:04

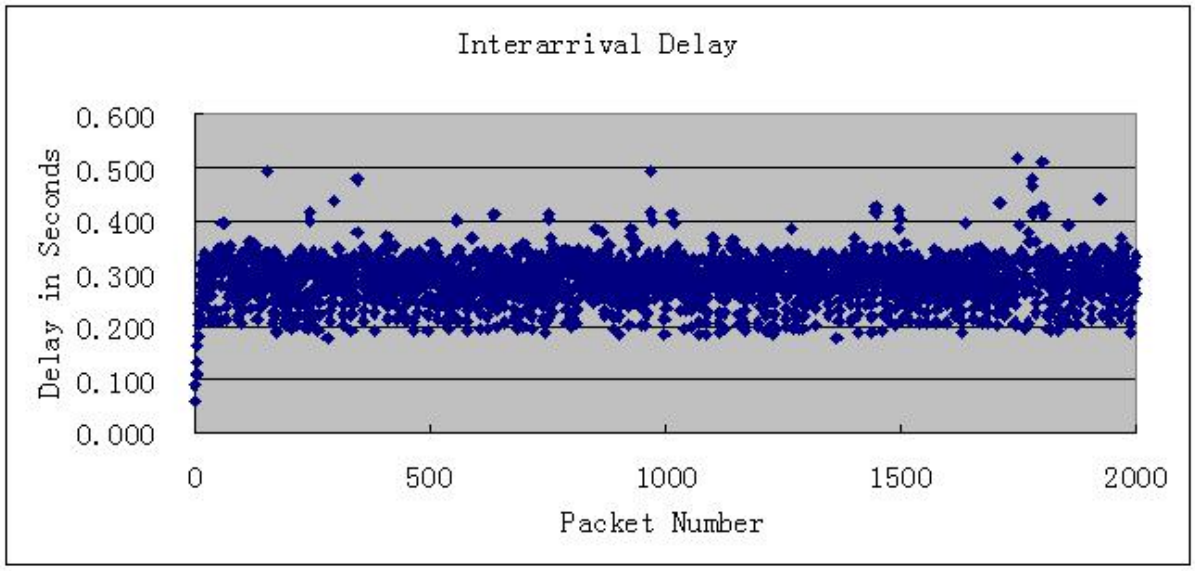


Figure 70: Sample 2 – Interarrival Time Delay, 2009-3-13, 14:14

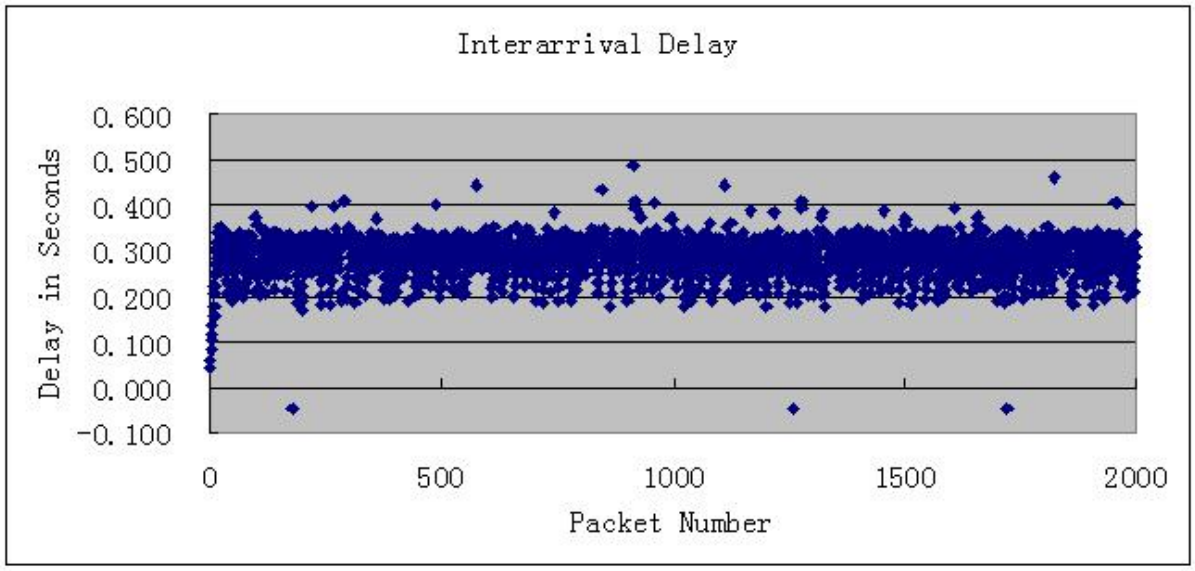


Figure 71: Sample 3 – Interarrival Time Delay, 2009-3-13, 14:23

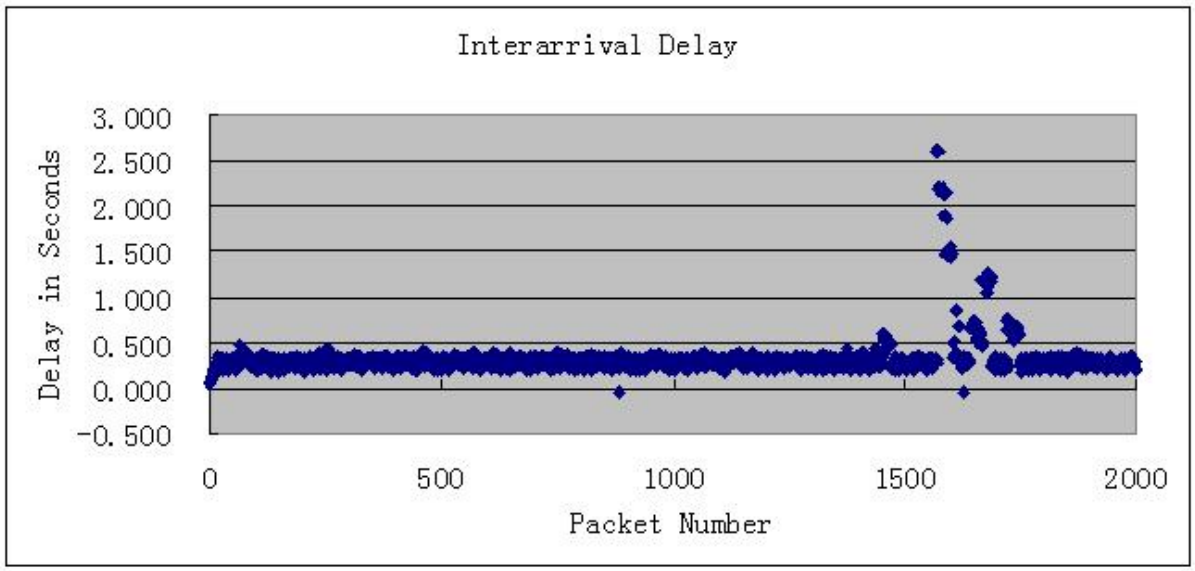


Figure 72: Sample 4 – Interarrival Time Delay, 2009-3-13, 14:32

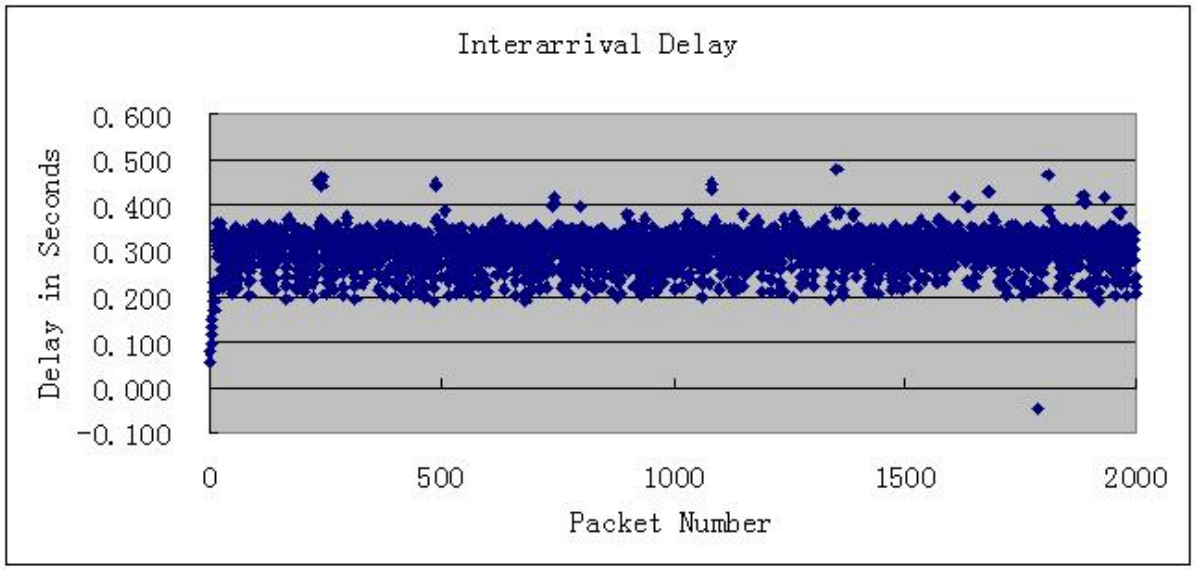


Figure 73: Sample 5 – Interarrival Time Delay, 2009-3-13, 14:43

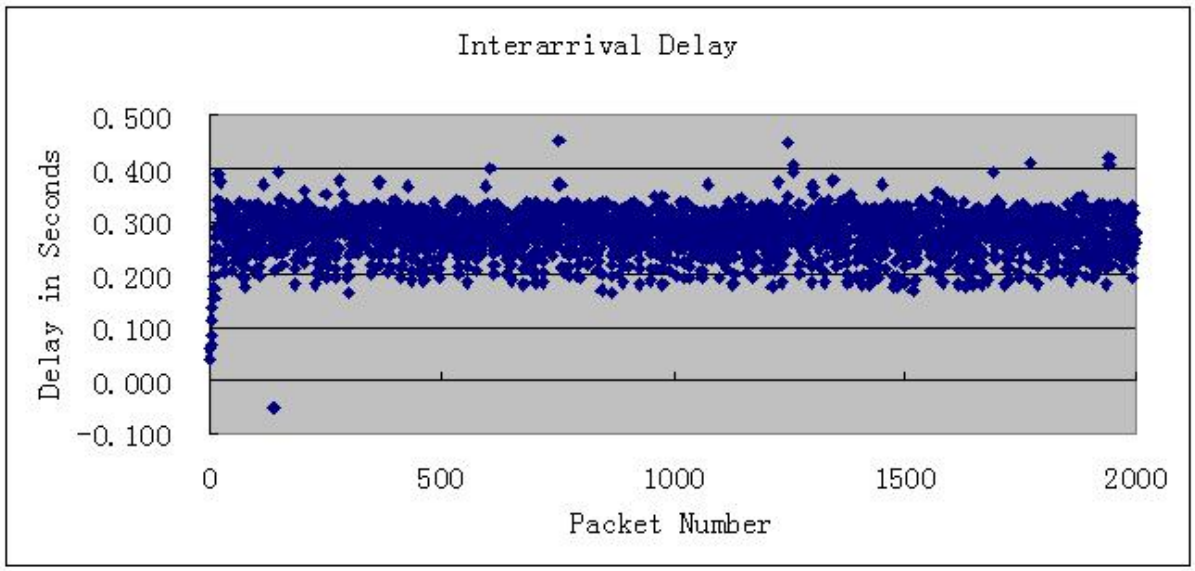


Figure 74: Sample 6 – Interarrival Time Delay, 2009-3-13, 14:52

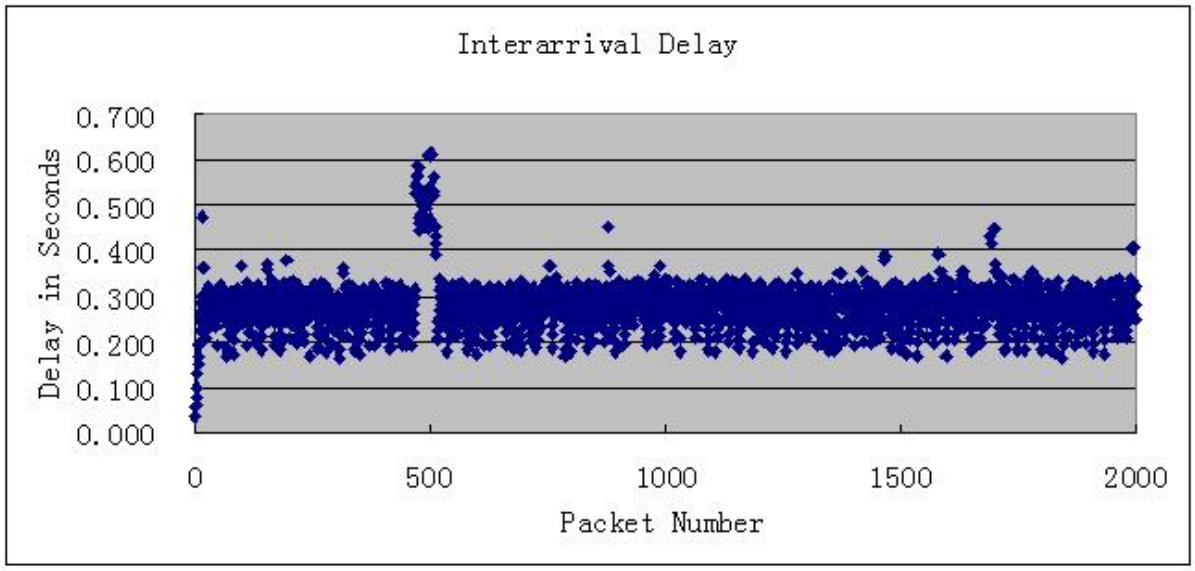


Figure 75: Sample 7 – Interarrival Time Delay, 2009-3-13, 15:02

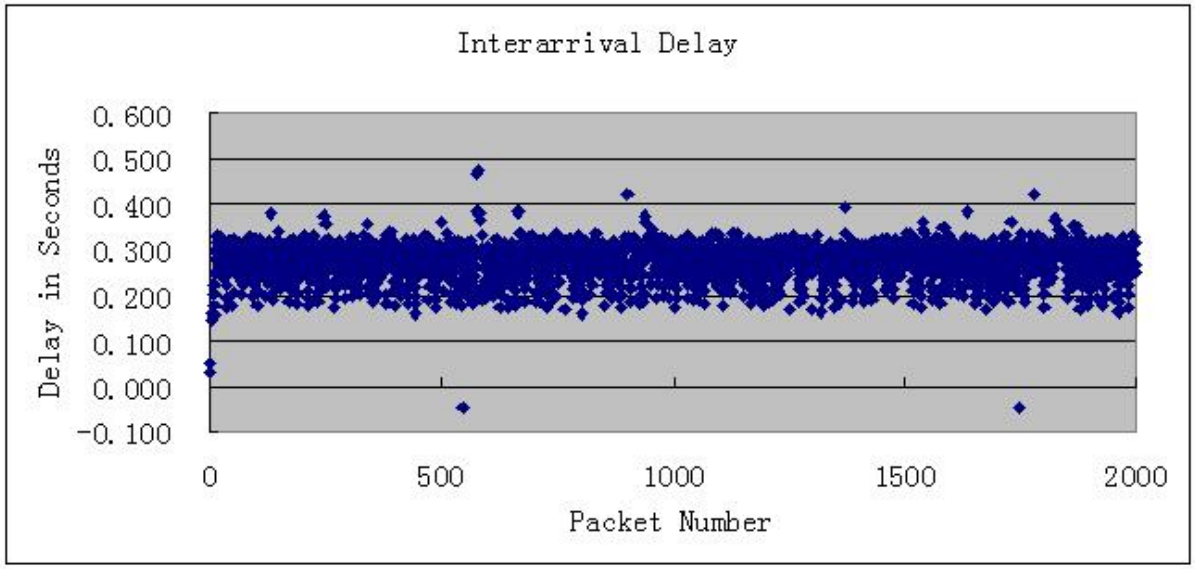


Figure 76: Sample 8 – Interarrival Time Delay, 2009-3-13, 15:12

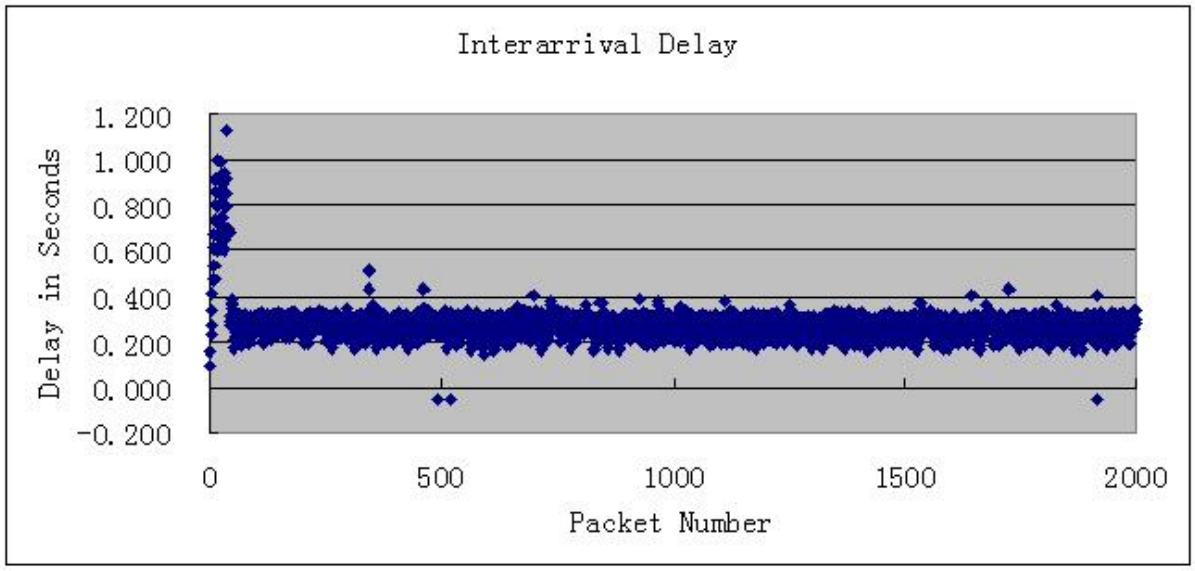


Figure 77: Sample 9 – Interarrival Time Delay, 2009-3-13, 15:21

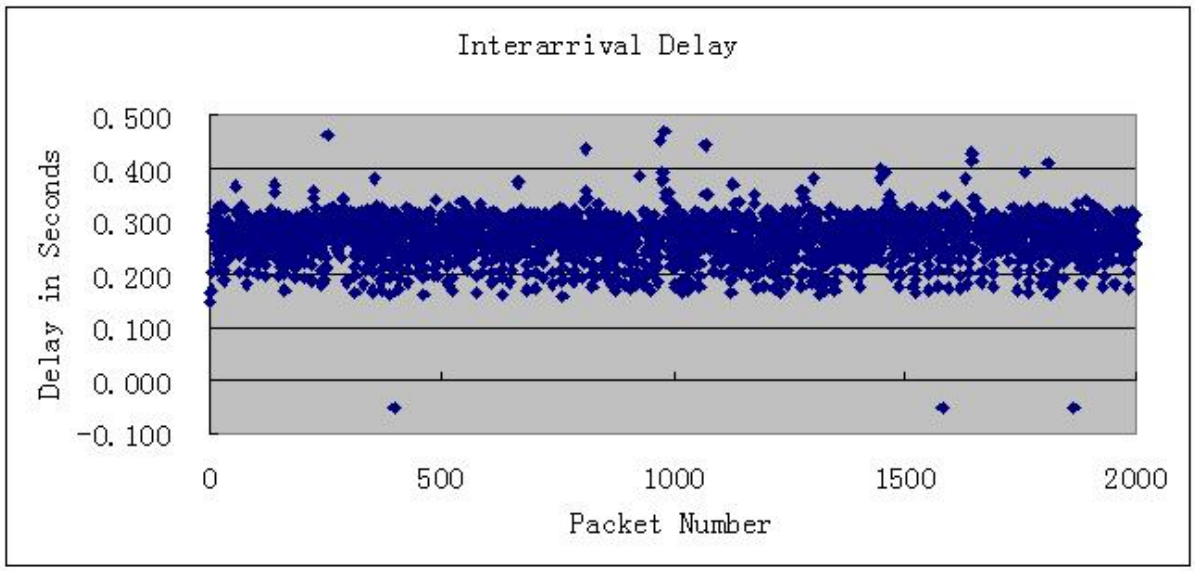


Figure 78: Sample 10 – Interarrival Time Delay, 2009-3-13, 15:32

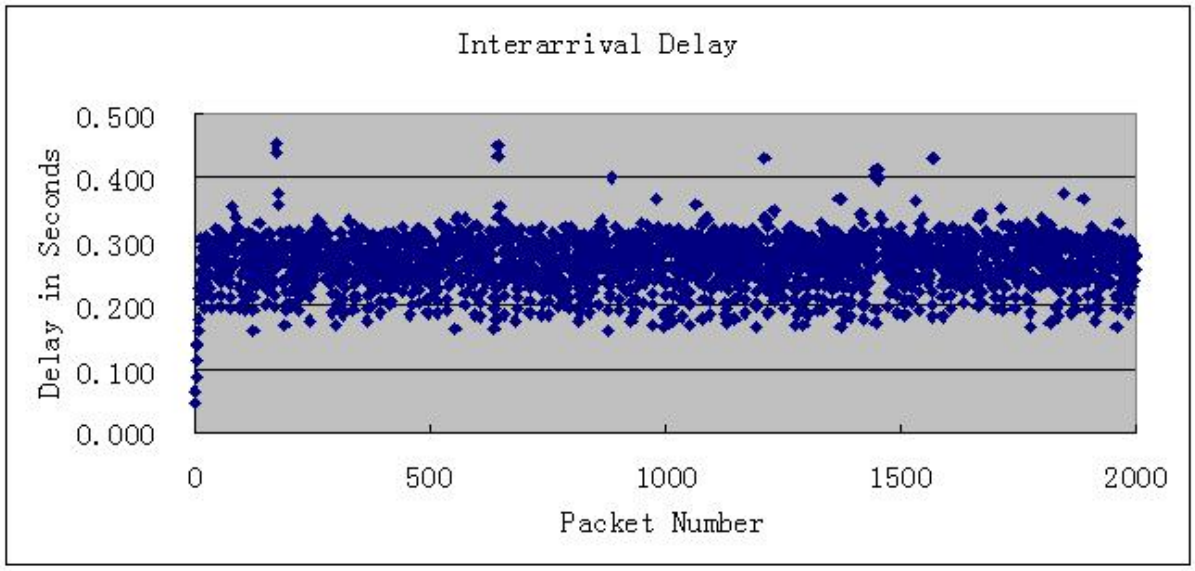


Figure 79: Sample 11 – Interarrival Time Delay, 2009-3-13, 15:41

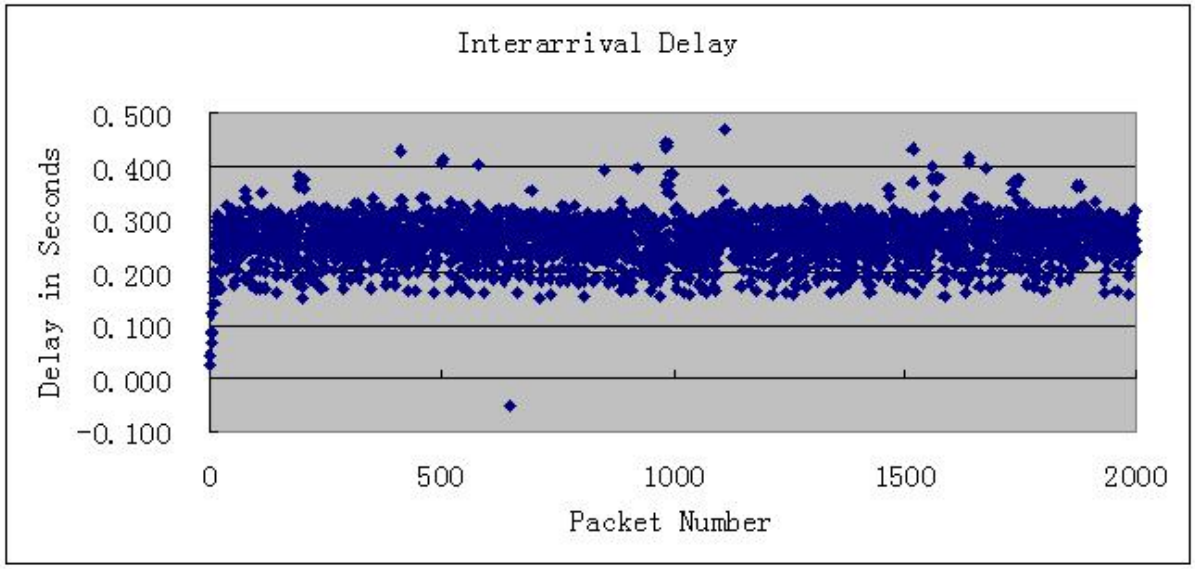


Figure 80: Sample 12 – Interarrival Time Delay, 2009-3-13, 15:53

TRITA-ICT-EX-2009:2

www.kth.se