HELSINKI UNIVERSITY OF TECHNOLOGY

Department of Electrical and Communications Engineering

Markus Ylikerälä

Real-time data transportation with fake tunneling

Master's thesis that has been given for review for the degree of Master of Science in Technology, Espoo, May 13th, 2003.

Supervisor:	Professor Teemupekka Virtanen
Instructor:	Group Manager Raimo Launonen

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	**Markus Ylikerälä**
Title:	**Real-time data transportation with fake tunneling**
Date:	**May 13, 2003**
Pages:	**73**
Department:	**Department of Electrical and Communications Engineering**
Professorship:	**T-110 Telecommunications Software**
Supervisor:	**Professor Teemupekka Virtanen, Helsinki University of Technology**
Instructor:	**Raimo Launonen, Group Manager, VTT Technical Research Centre of Finland**

Content: Telecommunication networks enable connecting of users at different places so that the users can simultaneously use the same multi-user virtual environment. This enables collaboration between the users of the virtual environment over the network because they can interact with each other and manipulate the same virtual objects in real time. In addition to the network and user device, a protocol is needed to transport real-time data between the users. Although transport protocols that are suitable for real-time interaction should be used, their use can be denied for security reasons. But, transport protocols that are not restricted for security reasons can have other characteristics that make them unsuitable for real-time interaction.

The subject of this Master's thesis was to research what kind of transport protocol should be used to achieve real-time interaction over the network. As a result of the research, the constraints of the current Internet transport protocols are avoided to enable collaboration of distinct users in the same multi-user virtual environment.

As the practical part of this Master's thesis, the author has developed the concept of fake tunneling used on the Internet Protocol (IP) based networks. Fake tunneling is designed according to the current Internet transport protocols and implemented according to the standard, de facto, interfaces. Fake tunneling was compared to the current transport protocols and was shown to be superior.

Keywords: **header, interface, protocol**

TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	**Markus Ylikerälä**
Työn nimi:	**Reaaliaikainen datan siirto lumetunneloinnilla**
Päivämäärä:	**13. toukokuuta 2003**
Sivumäärä:	**73**
Osasto:	**Sähkö- ja tietoliikennetekniikka**
Professuuri:	**T-110 Tietoliikenneohjelmistot**
Työn valvoja:	**Professori Teemupekka Virtanen, Teknillinen korkeakoulu**
Työn ohjaaja:	**Ryhmäpäällikkö Raimo Launonen, VTT Tietotekniikka**

Tiivistelmäteksti: Tietoliikenneverkot mahdollistavat eri käyttäjien yhdistämisen niin että käyttäjät voivat yhtäaikaisesti käyttää samaa monen käyttäjän virtuaaliympäristöä. Tämä mahdollistaa virtuaaliympäristön käyttäjien välisen yhteistyön verkon välityksellä, koska he voivat olla vuorovaikutuksessa keskenään ja käsitellä samoja virtuaaliesineitä reaaliaikaisesti. Verkon ja käyttäjän laitteen lisäksi tarvitaan jokin protokolla siirtämään reaaliaikaista dataa käyttäjien välillä. Vaikka reaaliaikaiseen vuorovaikutukseen sopivia siirtoprotokollia tulisi käyttää, niiden käyttö voi olla kiellettyä turvallisuussyiden perusteella. Toisaalta siirtoprotokollilla, joiden käyttöä ei ole rajoitettu turvallisuussyillä, voi olla ominaisuuksia, jotka tekevät niistä sopimattomia reaaliaikaiseen vuorovaikutukseen.

Tämän diplomityön aiheena oli tutkia minkälaista siirtoprotokollaa pitäisi käyttää reaaliaikaisen vuorovaikutuksen saavuttamiseksi verkon välityksellä. Tutkimuksen tuloksena osoitetaan, että nykyisten Internet- siirtoprotokollien rajoitukset voidaan välttää, jotta erilliset käyttäjät pystyvät yhteistyöhön samassa monen käyttäjän virtuaaliympäristössä.

Diplomityön käytännönsovelluksena tekijä on kehittänyt lumetunneloinnin käsitteen käytettäväksi Internet Protokollaan (IP) pohjautuvissa verkoissa. Lumetunnelointi on suunniteltu nykyisten Internet- siirtoprotokollien mukaisesti ja toteutettu de facto standardi rajapintoja noudattaen. Työssä Lumetunnelointia verrattiin nykyisiin siirtoprotokolliin ja se osoittautui ylivoimaiseksi.

Avainsanat: **otsake, rajapinta, protokolla**

PREFACE

This Master's thesis was done at VTT Technical Research Centre of Finland, VTT Information Technology.

The supervisor of this thesis was Professor Teemupekka Virtanen at Helsinki University of Technology and the instructor was Group Manager Raimo Launonen at VTT Technical Research Centre of Finland, VTT Information Technology.

I would like to thank Professor Teemupekka Virtanen for his extremely valuable instructions and Raimo Launonen for his views and support for this thesis.

I wish also to thank my colleagues who have helped me with this thesis.

Finally, I would like to thank my family for their support and my friends who have been interested in this thesis. Especially, I would like to thank Piia for her love and support that guided me during the process.

Espoo, May 13th 2003

Markus Ylikerälä

TABLE OF CONTENTS

ACRONYMS and DEFINITIONS

API	Application Programming Interface
ARP	Address Resolution Protocol, converts IP address to physical address
avatar	object or shape that represents a user in virtual reality
BSD	Berkeley Software Distribution, distribution of UNIX operating system
DHCP	Dynamic Host Controlling Protocol, maps physical address to IP address
DNS	Domain Name System maps domain name to IP address and vice versa
firewall	network component that can restrict data transportation between networks
fragmentation	packets that are to large to be transported are divided and denoted as fragmentation.
HTTP	Hypertext Transfer Protocol, markup language to describe web pages on the World Wide Web
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
Internet	all the networks in the world connected together using Internet Protocol
IP	Internet Protocol
IPv4	version 4 of the Internet Protocol
IPv6	version 6 of the Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
J2SDK	Java 2 Software Development Kit
JRE	Java Runtime Environment
LAN	Local Area Network, fast small scale networks

LTP	Lume Tunneling Protocol
MTU	Maximum Transfer Unit, maximum amount of data that a physical network can transport in a packet
MAC	Media Access Control
Napster	service for sharing files on the Internet
OS	Operating System
P2P	Peer-to-Peer, network technology where data can be transport between equal peers
PC	Personal Computer
platform	combination of the hardware and the operating system
QoS	Quality of Service, metrics that describes data transportation characteristics of a network
RFC	Request For Comments, name of the documents of IETF
router	connects networks and transports data between them
RTP	Real Time Transport Protocol
SETI@home	Search for Extraterrestial Intelligence at home by analyzing radio telescope data with computers
socket	abstraction to enable communication between processes over the network
switch	connects network components such as computers
TCP	Transmission Control Protocol
teleimmersion	realistic sense of being in another place
UDP	User Datagram Protocol
UML	Unified Modeling Language, notation for software development
UI	User Interface
VE	Virtual Environment
VTT	VTT Technical Research Centre of Finland

LIST OF FIGURES

1 Introduction

This Master's thesis has been done at VTT Technical Research Centre of Finland, VTT Information Technology, as a result of the T1P2P project [1] that the author was involved in. The T1P2P project was an internal research project. The purpose of the T1P2P project was to research grid-based technologies and to apply Peer to Peer (P2P) technology to data transportation. The growing interest in P2P technologies was the reason to get familiar with such a technology. The related View of the Future project [2] was used as the conceptual test bed for the T1P2P project. The View of the Future project is a project IST programme of the European Union. Related to the project a prototype for training of astronauts in a virtual environment (VE) was developed and different interaction methods were investigated.

Connecting different users with networks enables simultaneous use of the same multi-user virtual environment (VE) such as collaborative virtual environments and teleimmersion. Applications that can take advantage of teleimmersion are e.g. education, training, scenario simulations, art and entertainment [3]. As a result, the multi-user VE can be used for collaboration of the users at different places. The VE can represent a real environment but also be totally unrealistic. In the VE users can see each other as they are or possibly as avatars, talk to each other and handle the same objects. The users can use different kinds of devices, such as a desktop computer, laptop, handheld, mobile phone etc., connected to the same network such as the Internet. The network transports the control, spatial and visual, aural data between the users so that real-time interaction is possible. The User Interfaces (UI) that the hardware or software offers to the user can consist for example of text, two- and three-dimensional images, voice commands, haptic feedback, tracking, odor etc. In addition, virtual environments (VE) can be used as a UI. The development of Personal Computer (PC) technology and the prices of that technology have made it possible to use PC technology to produce VE. The benefit of the VE compared to the UI produced by the other devices is the more realistic sense of immersion that can be achieved. Although many people can use such a VE at the same time, at this moment, the viewpoint in a stereographic VE is correct only for a single user. But, with the remote use of the same shared multi-user virtual environment (VE), this problem can be solved. So networks are needed to transport data between users so that the real-time interaction would be possible. If the Internet is used as the network, the common resources of the network are shared with all the other users. In addition, the Internet was originally designed for data transportation in unreliable circumstances and not for the real-time interaction. If the resources of the network are limited, Quality of Service (QoS) can be used to classify different data transportation. As a result, data transportation of higher priority is treated as more urgent than data transportation of lower priority. Unfortunately, the Internet Protocol (IP) based networks do not generally guarantee QoS [4]. Data transportation on networks is controlled with protocols, which are used for sending and receiving the data between users. The restricted capabilities of the network, protocols and also characteristics of different kinds of data can set limitations to data transportation. This can affect usability of the multi-user VE and comprehensibility of the other users and as a result real-time interaction can be impossible.

The alternative data transportation protocols used on the Internet Protocol (IP) based networks, such as the Internet, are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). It depends on the application which protocol should be used. TCP is suitable for reliable data transportation and e.g. Hypertext Transfer Protocol (HTTP) used with World Wide Web (WWW) uses TCP. UDP is suitable to data transportation when some data can be lost e.g. Real Time Protocol (RTP) uses UDP. TCP is not suitable for real-time interaction because when data is lost, TCP tries to send it until it succeeds [5, 6]. But, meanwhile the value of data has expired because it no longer represents the situation of that moment but the situation in the past. In addition, TCP buffers data so that the network resources are better utilized but waiting for the data only causes interference to the interaction. Because UDP does not retransmit or buffer data, it is more suitable for real-time interaction than TCP [5]. But, unlike TCP, UDP can transport data out of the order so outdated data can occur. Presentation of outdated data in real-time interaction is not desirable and a situation like that should be handled properly. In addition to the restrictions of networks and protocols, a firewall used for security reasons can also set limitations to data transportation [7]. The firewall is used to separate the network of an organization or individual from the other networks like the Internet. All data between these networks is transported through the firewall that controls and can restrict data transportation. For example, all data transportation that uses UDP could be denied and only data transportation that uses TCP would be allowed. As a result, only TCP could be used although UDP would be more suitable for real-time interaction. Tunneling enables transporting of unsupported protocol with supported protocol. As a result, to circumvent restrictions of firewalls, tunneling could be used to transport UDP with TCP. But, the unsuitable characteristics of TCP for real-time data transportation remain.

At this moment, many software services on the Internet are based on the Client-Server architecture and the entities of this architecture are denoted as clients and servers. For example, WWW and email are based on this architecture. Some drawbacks of this architecture are that a server and the connection from the server to clients can have inadequate capacity for serving all the clients. Also, resources of the clients are usually under-utilized compared to the server and a malfunction of the server can affect all the clients. An alternative way to use the resources of the Internet and to offer services is the peer-to-peer (P2P) architecture and the entities of this architecture are denoted as peers. Napster [8] and SETI@HOME [9] are examples of this kind of architecture. Some benefits of this architecture are that data are distributed among peers. When data are available at many peers, not all of the peers have to use the same connection to get that data. As a result, the peers can get the data faster and the resources of the network are better utilized. Also, a malfunction of one peer does not have to affect other peers.

Transport protocols and P2P are related to the grid technology and were investigated at the T1P2P project. At VTT Technical Research Centre of Finland, VTT Information Technology, researching of transport protocols and P2P technology was seen as the potential trend of the future. The author has done all the work for designing and implementing the concept of Fake Tunneling that offers a P2P platform suitable for real-time data transportation.

2 Determination of the Statement

In this chapter the research problem and the subject of this Master's thesis are presented. Also, the evaluation criteria and scope of the research problem are considered. Finally, the overview of this Master's thesis is described.

2.1 Research Problem

The research problem given to the author is to develop a solution that would enable use of multi-user Virtual Environment (VE) over the network with different kinds of User Interfaces (UI) and user devices that are connected together so that peer-to-peer (P2P) aspects are taken into account. Especially a single-user VE used with View of the Future project ought to be connected. As a result, real-time interaction between the users should be possible.

The subject of this Master's thesis is to research what kind of transport protocol should be used so that real-time data transportation over the network could be achieved without the constraints of the current Internet standard transport protocols. The research for developing such a solution is the subject of this Master's thesis.

2.2 Evaluation Criteria

The most important criteria of this Master's thesis are going to be used for the evaluation of the solution that the author will develop according to the research problem.

Criterion 1:
The solution should be feasible and it should be able to be implemented in a reasonable time.

Criterion 2:
The solution should be capable to real-time data transportation so that use of multi-user VE is achieved between users at different places and that the users are capable to real-time interaction.

Criterion 3:
The design of the solution should be done according to the current Internet protocols. Also, the use of the solution must be possible with the current protocol implementations.

Criterion 4:
The platform should be executable on different kinds of operating systems (OS) and it should be able to be attached to the existing VE.

2.3 Scope

To utilize the current technology and to limit the topic of this Master's thesis the scope contains the following aspects, although, it should be noted that these aspects provide the framework for this Master's thesis but not for the solution to the research problem.

- Internet Protocol (IP) version 4 (IPv4) technology
- Connectionless networks, namely packet switched networks
- Personal Computer (PC) and Ethernet technologies

2.4 Overview

Chapter 3 considers constraints and requirements of real-time data transportation on connectionless networks. As a result, the chapter describes the main area of technology related to this thesis.

Chapter 4 considers the current protocols that are used with the Internet. Also, the chapter describes the current technology and the problem area of this thesis.

Chapter 5 describes the new technology denoted as Fake Tunneling as the solution of the author to the research problem. The chapter considers the improvement to the current technology that can be achieved with Fake Tunneling.

Chapter 6 considers different implementation aspects and also describes an implementation of Fake Tunneling.

Chapter 7 includes description of the tests and test results that are made to compare the new technology of the author to the current technology. The chapter also concerns evaluation of the test results and criteria of this Master's Thesis.

Finally, Chapter 8 makes conclusions and also describes the further development of Fake Tunneling.

3 Requirements of Real-time Data Transportation

Basically, the users needs set requirements to the service that they are willing to use. The service is used through the user interface (UI) that the software and hardware offer. The characteristics of the UI and also the data that the service uses affect the usability of the service. The format and interpretation of the data are dependent on that particular service. Data transportation between users can be enabled with networks, which consist of two or more network components that are connected together. As a result, also the limitations of the network can affect the usability [3]. For example, transportation of real-time data has different requirements to the characteristics of the network compared to transportation of non real-time data and Quality of Service (QoS) that the network offers can be used to affect the use of the network resources.

The objective of this chapter is to consider different requirements of real-time data transportation so that real-time interaction could be achieved. First, a definition of the term real- time is declared. In addition, usability is described and an example of a use of virtual environment (VE) is given. After that, concepts of networks and network components are considered. Also, different kinds of connections between network components are described. In addition, network limitations, QoS and techniques that can be used to achieve better transport performance are considered. Finally, two different kinds of network architectures are described.

3.1 Definition of Real -Time

There is a difference between the terms fast and real-time [10]. If some event is fast then it occurs rapidly. But if some event is real-time then it occurs at the certain time. The real-time system means the ability of the system to react to some event in a predictable time. This is different to react to some event as fast as possible. So one difference between the real-time system and the non real-time system is that the behavior of the real-time system is predictable compared to the non real-time system. This behavior can be achieved e.g. with harder control on hardware activities and scheduling tasks based on time. There are number of systems that benefit from or require real-time behavior. A few examples of real-time systems are virtual reality, telecommunication, robotics and military systems [10]. The drawback of the real-time systems compared to non real-time systems can be poorer throughput of tasks. But if this is the problem, while real-time behavior must be guaranteed, one solution is to achieve more efficient hardware. Also, using better algorithm in software can be used to improve performance.
[10]

3.2 Usability

The term usability can be used in many different circumstances. With Information Technology (IT) the usability can be defined e.g. according to Organization for Standardization (ISO) or the usability researcher Jakob Nielsen. The definition according ISO is the following. Usability is specific goals that a particular user can achieve in such a way that the achieving is effectiveness, efficiency and get satisfaction to the user [11]. The definition according to Jakob Nielsen is that usability is the measure of the quality of a user's experience when interacting with a product or a system [12]. The purpose of these definitions is to enable measurement of usability so that usability can be affected. The user uses a service through the User Interface (UI) that the software and hardware provides [13]. A UI can consist of text, two- and three-dimensional images, voice commands, haptic feedback, tracking, odor etc. Although the user needs not to be aware of a possible use of a network, the use of the network effects the usability in general because the service can use real-time data like graphics, speech and also location, velocity, acceleration etc. information that is transported over the network. The network causes lag perceived by the user [3]. Lag means the time it takes between related events. For example, the time it takes between speaking into the microphone and hearing it from the speakers or the time it takes between moving the head and seeing the updated graphics produced according to the heads position. If the lag is high enough, the use of the VE can be impossible [3].

Although, the human can sense by sight, hearing, smell, taste and feeling, not all of them are needed in the use of a virtual environment (VE). Usually at least visual and aural methods are considered because they are suitable in real-time interaction and adequate technology is available [3]. Also, tracking can be used to enable producing of correct viewpoint, handling of objects etc. In Figure 3.1 the author is in an interaction situation in a VE.
[13]

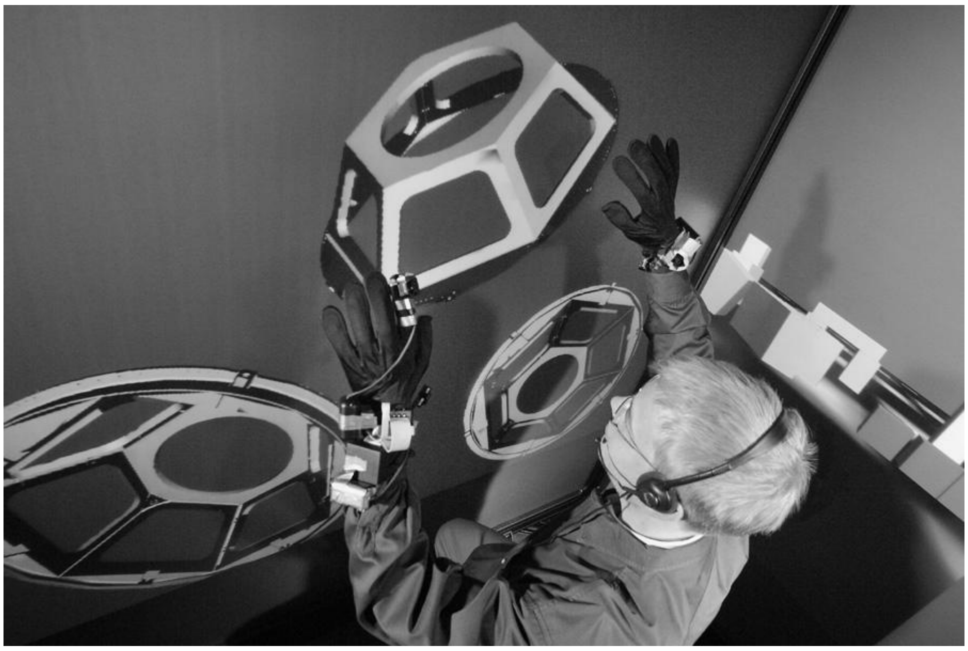


Figure 3.1 Interaction in Virtual Environment

3.3 Concept of Networks

Networks enable connecting of users at different places so that the users can simultaneously use the same multi-user virtual environment. Indeed, networks form connections between the users and enable data transportation between them. The network can consist of different kinds of hardware and software that form network components and connections between them. In addition, a use of software in networked environment can introduce unpredictable situations that should be handled properly. Finally, the network can provide different kinds of data transportation types and a choice between them can affect the network utilization. In addition, transport protocols and hardware can set limitations to their use.

3.3.1 Network Topologies and Components

Network components and connections between them can be described with a graph denoted as the network topology. In the topology, network components are described as nodes and the connections, which can be wired or wireless, between the nodes are described as links. Nodes are usually described as circles and links as arcs. If on connection data can flow only to the other direction, the direction of the data flow can be indicated with an arrow. It should be noted that not all of the network components and connections have to be described with the same graph. In Figure 3.2 two different kinds of network topologies are shown. On the left side of the figure is a star topology and on the right side is a fully connected topology. A route between nodes consists of those nodes, links and other nodes connected together. Data can be transported between the nodes only if there is a route between them. In Figure 3.3 there are to networks but no route between them.

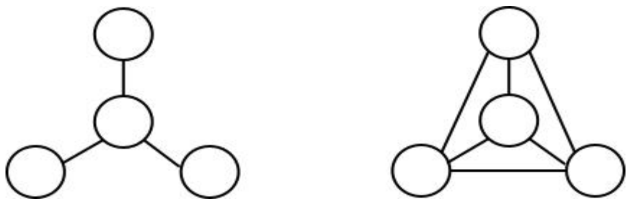


Figure 3.2: On the left a start topology and on the right a fully connected topology

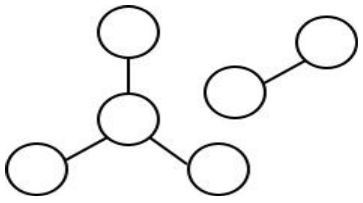


Figure 3.3: Two distinct networks without any route between them

Some important network components are computers, switches, routers and firewalls. Computers can be used to execute computer programs that can be used to both send data to another network component and also to receive data from another network component. Switches are used to connect network components. Routers, also called gateways, are used to connect networks and to transport data between them.

Firewalls are used to separate networks from each other to control and restrict the data transportation. The data transportation through the firewalls can be restricted by different kinds of rules e.g. based on protocol [7]. In addition, firewalls act as routers.

Network components and connections between network components can also be described with graphic symbols, as shown in Figure 3.4. For the clarity, explicit routers and switches are not shown. Arbitrary networks can be represented with clouds, network components such as firewalls with brick walls and computers with images of a conventional computer. Computers and firewalls are connected to networks with lines that represent connections, which can be wired or wireless. The private network is usually denoted as Intranet and the public network as the Internet. Also sender and receiver of data are declared.

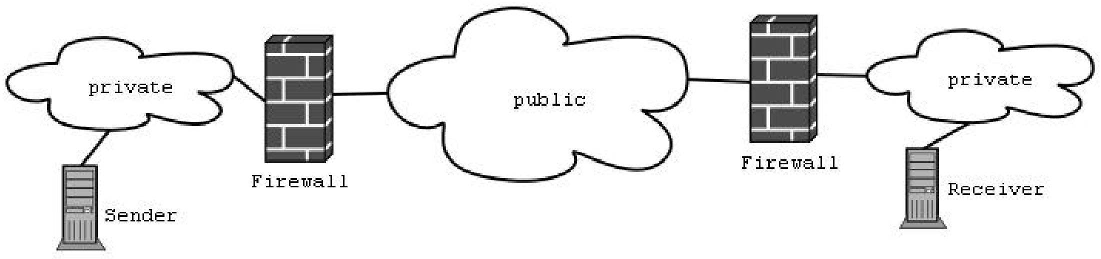


Figure 3.4: Representation of networks with graphic symbols

On the sender side data is divided into packets that are delivered separately across the network from the sender to the receiver. Each packet contains the unique address of the receiver that is used to deliver the packet to the receiver. The transport is connectionless because the sender shares the resources of the network with all the other users. So the delivering is based on best effort because no resources are allocated for a delivery but the delivery must compete with all the other deliveries that use the same route for the finite resources. When the packets arrive to the receiver the data is assembled from the packets. The transport is unreliable because some packets may be lost, be duplicated, arrive in different order than they were sent or delayed.
[5, 14]

3.3.2 Synchronization

Synchronization describes the relation of some event to other events or time. As mentioned before transportation of data across the network such as the Internet is unreliable. Compared communication between software in single computer to communication in networked and distributed systems, it is more complicated and error prone in the former than in the latter [15, 16]. The reasons for this are related to the use of memory and characteristics of the network.

3.3.3 Data transportation types

Data transportation between different network components can be classified into three types that are unicast, multicast and broadcast that have different kinds of characteristics and use. In Figure 3.5 different kinds of connection types are shown. The letters A, B, C, D and E denote computers so that A is always the sender. In addition a switch is denoted with the letter S and a router with the letter R. The propagation of data from the sender to the receivers is indicated with arrows.

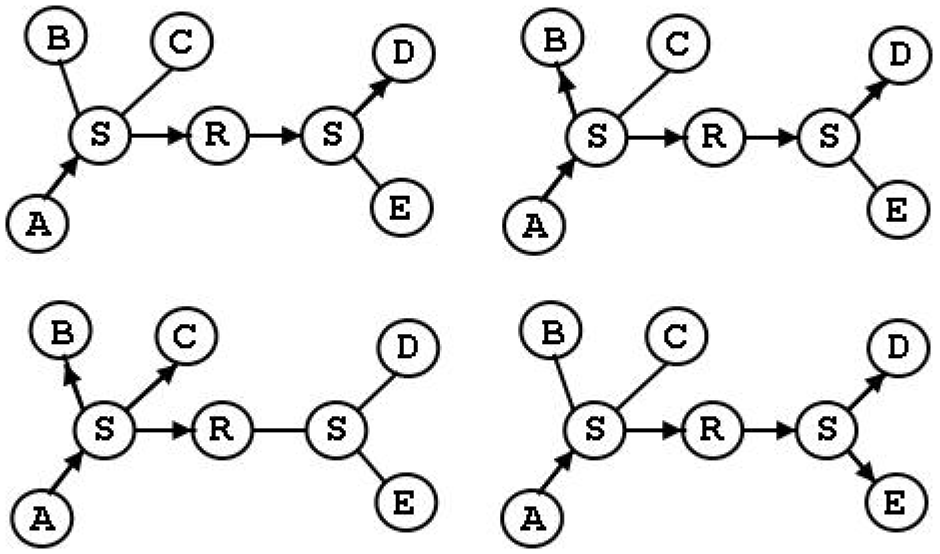


Figure 3.5: Different data transportation types

Unicast
Unicast can be used to sending data between two network components. In order to use unicast, the sender has to know the address of the receiver [14]. In Figure 3.5 an example of unicast data transportation is shown in the top-left when unicast data transportation is made between A and D.

Multicast
Multicast can be used to send data simultaneously to group of network components. In order to use multicast, the sender must know the address of the multicast group, which the receiver belongs [14]. In Figure 3.5 an example of multicast data transportation is shown in the top-right when multicast transportation is made between A, B and D. Routers must support multicast so that multicast can be used to send data to another networks.

Broadcast
Broadcast can be used to send data simultaneously to all network components of a network. Broadcast can be classified into local broadcast and global, also denoted as directed, broadcast. The former sends data only to the same network where the sender is but the latter can send data to any network. In order to use broadcast the sender must use a certain kind of well-known address [14]. In Figure 3.5 an example of local broadcast data transportation is shown in the bottom-left and an example of (global) broadcast data transportation in the bottom-right.
Choices between the data transportation types affect the utilization of the network resources. Basically, the choice depends on the service and the characteristics of the network which one of the data transportation types should be used. In addition characteristics of the transport protocol can restrict the use of data transportation types and as a result not all of them can be used [14]. In general, if the same data should be transported to multiple users, it is better to use multicast or broadcast than

unicast. But otherwise unicast can be better because it does not necessarily consume so much network resources. Finally, the solution that the author is developing should take able to support different kinds of data transportation types because real-time interaction can require transporting of different kinds of data. As a result, the most suitable data transportation type could be used.

3.3.4 Ethernet technology

With Ethernet technology a computer can be connected to the network with one or multiple network card. Each network card has a physical address that refers to that particular network card. In Ethernet technology this is a unique Ethernet address that is also called a Media Access Control (MAC) address or hardware address. The physical address can be used to achieve different kinds of connection types. In wired connections the network card is attached to the wire that is used as connecting media. The wire can be made e.g. of copper or fiber. In wireless connections the network card is capable to use radio frequencies as the connecting media.
[5, 14]

3.4 Network limitations

Restrictions of both hardware and also software that is used on the network can set several limitations to the data transportation. Hardware limitations that are concerned in general are bandwidth, throughput and latency.

Bandwidth
Bandwidth tells the maximum amount of data that could be transported in a certain time. Physical characteristics of the hardware set limitations that affect the bandwidth. Adding more transmission media and using media that allows better data propagation can increase the bandwidth [5.]

Throughput
Throughput tells the amount of data that is correctly transported in a certain time. Throughput can be maximized e.g. by minimizing corruption of data on transmission and increasing the speed of the network [5.]

Latency
Latency means delay and tells the time it takes to transport data from a sender to a receiver [3]. Because latency means the time between related events it is similar to lag. Round-trip time is latency that it takes to transport data from a sender to a receiver and back to the sender. For example, latency can consist of protocol processing, transmission of data between network components, limitations of bandwidth and routing [3]. Latency can be minimized e.g. by improving protocol software, OS or the network interface but the if the speed of light [17] sets the upper limit for latency [5].

3.5 Network Quality of Service (QoS)

The term QoS is used to describe characteristics of data transport in a network. QoS is needed when the resources of the network are limited compared to the users' needs. If the resources of the network were feasible for all users, QoS would be useless [4]. QoS should be able to be affected when a network is used for real-time data transportation. Examples of network QoS parameters are rate, loss, delay and jitter of data. Rate measures the amount of the data that is sent, loss measures how much of the data is lost and delay measures the time that it takes to transport data from a sender to a receiver. Finally, jitter measures the variation of delay. QoS parameters are dependent on characteristics of different components between the sender and the receiver.
[3, 4, 5, 14]

With QoS, data and data transportation can be classified and treated differently so that the resources of the network can be utilized for different kinds of services. Transportation of real-time data has different kinds of demands compared to transportation of non real-time data. For example, real-time interaction data such as speech is usually sensitive to jitter and delay but can tolerate some data loss because they can be omitted or interpolated [3]. But, non real-time data such as email [18] does not in general demand a small delay or jitter even if a small delay was desired but none of the data should be lost or corrupted. Finally, requirements of data transportation are dependent on the particular service. Once some QoS is guaranteed, it should not be affected by other data transportation. Compared to the Internet, affecting QoS is usually easier in private networks such as Local Area Networks (LAN) of an organization or an individual because in general one authority controls the private network. But there are approaches that can provide QoS also on the Internet such as Integrated Services (IntServ) and Differentiated Services (DiffServ) of the Internet Engineering Task Force (IETF) [19].
[3, 4, 5, 19]

Because the Internet does not in general provide QoS compression or streaming of data can be used [3, 20]. The limited resources of a network can be utilized better with compression of data. So when data transported over the network, proper compression can be used. This can reduce the size of the data to be transported so that the transportation of compressed data can be done faster compared to the transportation of the non-compressed data. When compressed data is received it has to be decompressed before use. Although, compression can make the transportation of data faster, it causes delay before the transportation and in addition decompression causes delay after the transportation. So, if fast data transportation is desired, the speed up gained from the data transportation should be more than the delay caused by the compression and decompression phases. Another way to achieve performance is streaming. Waiting for the end of the data transportation of the whole data before its use is not suitable for real-time data and streaming should be used instead. Streaming means that presentation of data is started as soon as some data arrive and presented while the rest of the data is still being transported. So, the benefits of streaming are that a receiver does not have to wait the completion of the whole data transportation before beginning to use that data. Instead, the receiver can use the data at the same time as the sender produces and sends data. Finally, streaming should suite well for interaction purposes.

3.6 Network architectures

Network architectures provides an abstraction to the communication of network components. The chosen network architecture also affects to the utilization of network resources. At this moment, many software services on the Internet are based on the Client-Server architecture but an alternative way to use resources of the Internet and offering services is the peer-to-peer (P2P) architecture.

3.6.1 Client-Server Architecture

The entities of the Client-Server architecture [14] are denoted as clients and servers and communication between them is well defined. The communication between a client and a server is following. The server waits until some client contacts it. So, the client always starts the communication. The client contacts the server by sending data called request. Then the server handles the request and finally sends data called response to the client. This whole client-server functionality is shown in Figure 3.6. It should be noted that the Client can retrieve data from the Server without user interaction and, as a result, network utilization can be increased and the delay perceived by the user decreased.

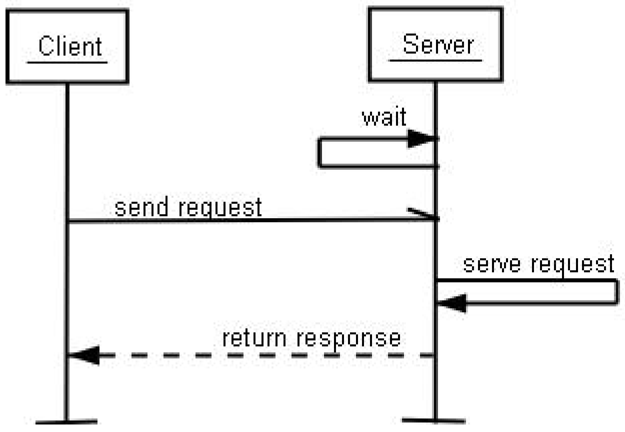


Figure 3.6: Client-server functionality

For example World Wide Web (WWW) [21, 22] is based on the Client-Server architecture. There the WWW browser is the client and the beginning of the WWW address contains domain name that identifies the server. WWW address can be typed on the text field of the browser or hyperlink can contain it. The request is send by pressing return or clicking hyperlink. After that the client can notice if the server is functioning or not. If the server is not functioning some error message is usually shown. Otherwise, the request is served by mapping the rest of the WWW address e.g. to a page or a file that is sent to the client, which finally shows it. If the server is functioning but the page cannot be found, some error message denoting this situation is usually shown.

To consider benefits and drawbacks of client-server architecture, an example of a client-server network is given in Figure 3.7. The server is denoted with the letter S and the clients with the letter C. The benefits of the client-server architecture are that the central authority eases the administration of the whole system. Using of the same central resource can be used to control the clients and the same services are available to all clients. The drawbacks of the client-server architecture come from the central role of the server. Malfunction of the server paralyzes the whole system. Also, connection from the server to the clients and also the server itself can have inadequate capacity for serving all the clients. This can be shown as slow data transportation or unavailability of the server. In addition, resources of the clients are usually under-utilized compared to the server.

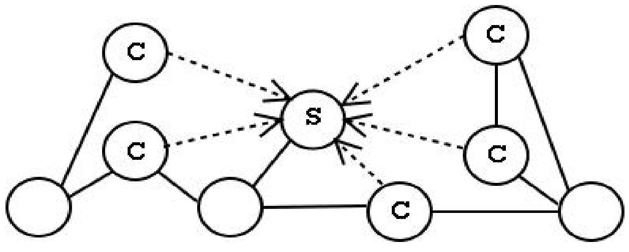


Figure 3.7: Benefits and drawbacks of client-server networks

3.6.2 P2P Architecture

The entities of the Peer-to-Peer (P2P) architecture [23] are denoted as peers that are equal and each peer can contact another peer. A peer can offer services to other peers and also use services of other peers. This can also be shown in Figure 3.6 in which instead of a separate client and server both are denoted as peers. Both of the peers can contain the functionality of both client and server. A peer is not dependent on the specified peer but the same service can be requested from another peer.

To consider benefits and drawbacks of client-server architecture, an example client-server network is given in Figure 3.8. Peers are denoted with the letter P. The benefits of the P2P architecture is that services and data are distributed among peers. When services are available at many peers, the system is more scalable. A malfunction of a certain peer that offers some services that other peers are interested in, does not prevent other peers to get that service. The service can always be gotten from some other peer. Also, not all peers have to use the same connection to get that data. As a result, the peers can get the data faster and the resources of the network are better utilized. Also, a malfunction of one peer does not usually affect other peers. The drawbacks of the P2P architecture are that because availability of a peer is unpredictable, the functionality cannot be based on the certain peer. Also, administration can be difficult when there are no common management.

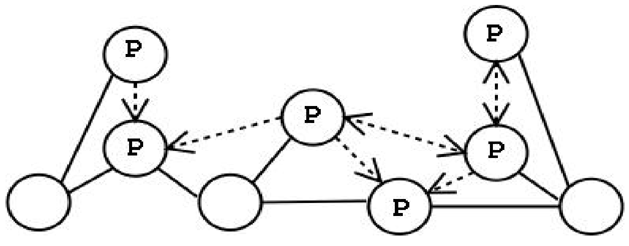


Figure 3.8: Benefits and drawbacks of P2P networks

The use of P2P technology enables development of new kinds of communication mechanisms between different network components. Especially when there is no central authority and the individual peers can be connected and disconnected in an unpredictable manner, the P2P architecture can be more scalable and robust. The solution that the author is developing should take the benefits and drawbacks of the both Client-Server and P2P architecture into account.

3.7 Summary

In this chapter a definition of the term real-time, usability and also concepts of networks and different kinds of data transportation types were considered. In addition, network limitations, QoS and two different kinds of network architectures were introduced. The usability can be seen as the most important aspect in the human computer interaction. The networks were described as a method to connect users so that data between them can be transported. The networks can be represented as topologies, which consist of network components such as computers and connections between the network components. The use of different data transportation types, namely unicast, multicast and broadcast, should depend on the particular service and a choice between them affect utilization of the network resources. But, restrictions of transport protocols and network characteristics can affect their use. QoS should be able to be affected if the resources of the network are insufficient. Compared to the non real-time data transportation, real-time data transportation can in general tolerate some data loss but is sensitive to jitter and delay. Because the current Internet cannot guarantee QoS in general, compression or streaming can be used instead. The use of P2P architecture enables an alternative to the Client-Server architecture and can be more scalable and robust when individual peers connected and disconnected the network in an unpredictable manner.

4 Current Transport Protocols

Transport protocols enable data transportation between different network components and can be implemented as computer programs. The fundamental protocols of the Internet are Internet Protocol (IP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP. The Internet Engineering Task Force (IETF) [19] controls the technical aspects and standardization of the Internet. Among other things, IETF also offers descriptions of protocols as documents, which are called Request For Comments (RFC).

The objective of this chapter is to consider IP-based networks. First IP addressing schema and protocols are considered. After that IP, TCP and UDP are described and also the limitations of these protocols in real-time data transportation are considered when real-time interaction should be achieved. Finally, a technique called tunneling is declared.

4.1 Network Addressing

A computer can be attached into network with one or multiple network card and each network card in a certain network is denoted as a network interface [14]. On IP-based networks the IP address refers to a certain network interface although it is usually meant to denote to a certain computer the network card is attached. Also, other network components such as a router can have an IP address. The IP address consists of a network id, which identifies a network and a host id, which identifies the network component attached to the network [14]. Because the same computer can have several network cards it can also have several IP addresses. An example of this is a router. IP address is used as an abstraction to provide the consistent address space in the Internet regardless of the network hardware.

Mapping an IP address to a physical address, such as the Ethernet address, can be done e.g. with Address Resolution Protocol (ARP) [24]. Mapping physical address to IP address can be e.g. with Dynamic Host Controlling Protocol DHCP [25]. IP address is represented as numbers but humans usually prefer meaningful names. The mapping between these domain names and IP addresses can be done e.g. with Domain Name System DNS [26]. In Figure 4.1 examples of domain name, IP address and physical address are given. In this case Ethernet address is used as the physical address. In a certain network, each one of these different addresses can be used to denote to the same computer. Ultimately the addresses are represented as bits.

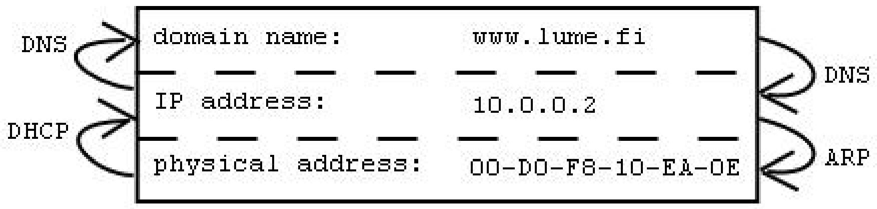


Figure 4.1: Different address representations and mapping between them

4.2 Concept of Protocols

Protocols denote the logical behavior of parties and can be described as a protocol stack [5, 14]. Each layer on the protocol stack relays on services of the layer below and adds some additional service. Logically, layers on the same level interact with each other and data is virtually transported between layers on the same level. But, physically the data flows through the stacks. On the senders side the data flows from the top to the bottom of the stack and on the receivers side the data flows from the bottom to the top of the stack. As data is moved between layers, on the sender side headers are attached to it. The resulting frame is transported over the network and on the receiver's side the headers are removed. As a result, the layers on the same level actually see the same data. Although there are many conceptual models of the TCP/IP protocol stack, in Figure 4.2 and 4.3 the TCP/IP protocol stack coherent to the terminology of this Master's thesis as introduced in [14] is shown. An alternative TCP/IP protocol stack introduced in [5] uses terminology of Open Systems Interconnection (OSI) Reference Model. It should be noted that basically either of them could have been selected.
[27, 28, 29, 14, 5]

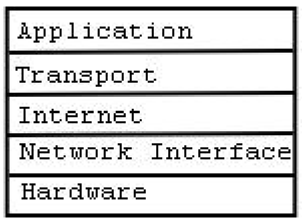


Figure 4.2: Conceptual TCP/IP protocol stack

In Figure 4.3 the protocol stacks of the sender and receiver are shown. Also, packet construction and deconstruction are represented. Directions of data flow are indicated with arrows. Logical connection is shown with the dashed line and the actual flow of the data is shown with solid lines. The physical connection can be wired or wireless. Attaching and removing of headers are shown between layers. In this figure TCP is used as the transport protocol but also UDP could be used instead of TCP.

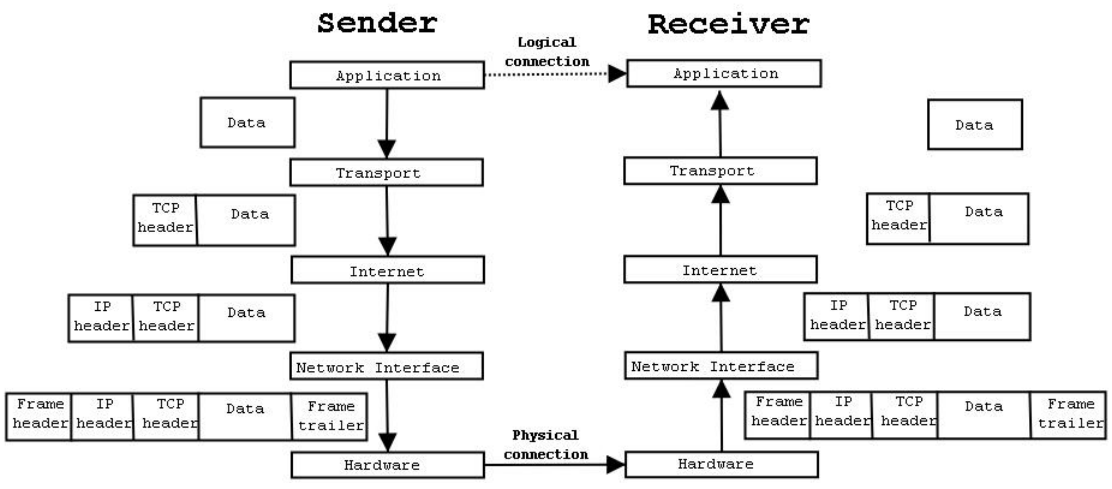


Figure 4.3: Logical and physical connections of TCP/IP protocol stack

4.3 Network Protocols

Although there are several data transportation protocols, it should depend on the application what kind of protocols should be used. In principle the transport and Internet layers of the TCP/IP protocol stack provides three different protocols that can be used to transport data. But, although applications can use IP directly, usually either TCP or UDP is used between application and IP. The purpose of the use of TCP or UDP is the additional services they provide [10] such as port numbers, reliable or unreliable data transportation and checksum calculation over the data portion of the packet. Each protocol defines the uniform format of the header. The purpose of the format is to organize the data into a well-known form so that protocols at different network components can construct, deconstruct and interpret packets uniformly. It should be noted that although a byte is usually thought to consist of eight bits, the number of bits in a byte can also be machine dependent and that is the reason why octet can be used to refer to exactly eight bits [14]. But, in this Master's thesis the byte is defined to contain exactly eight bits so the term byte (B) is used instead of octet. The most fundamental protocols on IP-based networks, such as the Internet, are IP, TCP and UDP [14]. Each description of these protocols contains a figure that declares a format of the protocol header, which consists of several fields and interpretation of them varies. Also the sizes of the fields vary and this sets the limit to the number of different values that a field can represent. In the figure the vertical sequence of numbers from zero to thirty-one denotes the bits of a row, so a row consist of 32-bits, and the horizontal sequence of numbers denotes the number of the rows. It should be noted that the numbering is started from zero as stated in the RFC that describes the protocols, nevertheless, the justifications of reasons for that are not relevant.

4.3.1 Internet Protocol (IP)

Internet Protocol (IP) [29] resides on the Internet layer and is the core of the Internet because it provides wrapping for IP-addresses and many other protocols rely on it. IP was designed to deliver data in unreliable conditions when some of the connections and network nodes on the route between the sender and the receiver could be damaged. The service that IP provides is connectionless and unreliable data transportation. There are two different version of IP namely version 4 (IPv4) and version 6 (IPv6). An IP packet also called datagram consists of an IP header and data area. The data area follows the header but the IP does not specify the format of the data area. So arbitrary data can be transported. In Figure 4.4 the format of the IPv4 header is shown.
[29, 14]

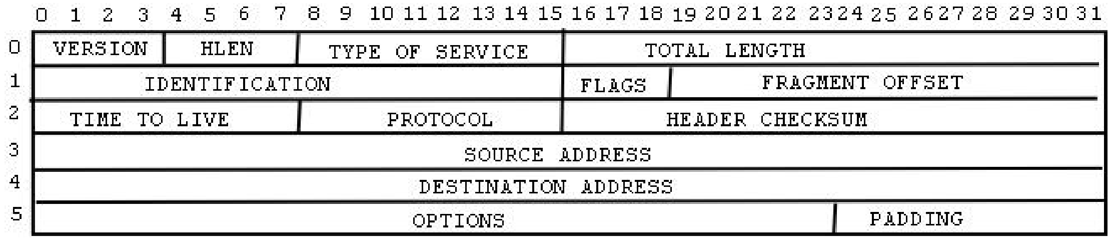


Figure 4.4: Format of IPv4 header

The VERSION field describes the content of the IP header so the rest of the fields are interpreted according to this field and can contain e.g. identification of IPv4 or IPv6. The HLEN field defines the length of the IP header measured in 32-bit multiples. If OPTIONS and PADDING fields are not used, the HLEN field is equal to 5, which is also the minimum size of the IP header. The TOTAL LENGTH field defines the length of the whole IP packet. The TYPE OF SERVICE field specifies how the network should handle the IP packet. This can be taken as a hint because there are no guarantee that the network takes these into account. The IDENTIFICATION field gives the unique identification number for the IP packet and in addition with the FLAGS and FRAGMENT OFFSET fields controls fragmentation and assembly of IP packets. The value of TIME TO LIVE field is decreased on the route from the sender to the receiver and the IP packet is discarded when it reaches zero. The PROTOCOL field defines the format of the IP data area and can contain e.g. identification of TCP or UDP. The HEADER CHECKSUM field contains the computed checksum over the IP header. The SOURCE ADDRESS and DESTINATION ADDRESS fields contain the unique IP address of the sender and the receiver in the same network. The OPTIONS field can contain additional options and then the PADDING field fills the rest of the header with zero so that the size of the header is multiple of 32-bits because of the HLEN field.
[29, 14]

4.3.2 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) [30] resides on the transport layer and the service that TCP provides is connection oriented and reliable data transportation. Compared to the IP TCP provides reliable data transportation, ports and checksum calculation over the data area of the TCP packet. TCP can be used for unicast connections. In Figure 4.5 the format of the TCP header is shown.
[30, 14]

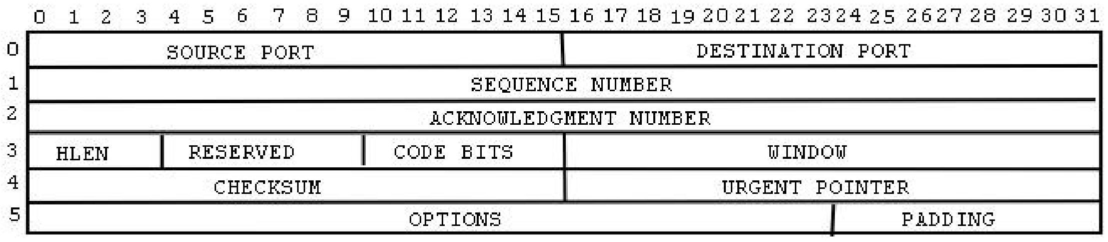


Figure 4.5: Format of TCP header

The SOURCE PORT and DESTINATION PORT fields define the endpoints of the sender and the receiver. Although the size of a port field sets the maximum number of endpoints, TCP uses the abstraction of connection, which is a pair that consists of the IP-address and the port of both the sender and the receiver. As a result, the same port can be shared between different connections. The SEQUENCE NUMBER and ACKNOWLEDGMENT NUMBER fields and also bits in the CODE BITS field are used to establish and close the connection. They are also used during the data transportation to enable reliable connection, so that the receiver sends an acknowledgment when it has received data from the sender. If the sender does not get the acknowledgment in a certain time the data is retransmitted. Usually the sender sends multiple packets instead of one before waiting of acknowledgment so that the resources of the

network are better utilized and the sliding window mechanism is utilized. Also the WINDOW field s used as a part of this mechanism to control the number of packets that can be sent before the sender must get acknowledgment from the receiver. The HLEN field defines the length of the TCP header measured in 32-bit multiples. If OPTIONS and PADDING fields are not used the HLEN field is equal to 5, which is also the minimum size of the TCP header. The CHECKSUM field contains the computed checksum over the TCP packet. The URGENT POINTER field and a bit in the CODE BITS field are used to indicate that processing of that particular TCP datagram is urgent. The OPTIONS field can contain additional options and then the PADDING field fills the rest of the header with zero so that the size of the header is multiple of 32-bits because of the HLEN field
[30, 14]

The reliable data transportation that TCP offers is important to many services, such as email [18], because data should not be lost. But, reliable data transportation is not suitable for real-time data transportation because it causes jitter if data need to be retransmitted [5]. In addition, TCP usually buffers data so that more data is transported at once. This enables better utilization of network resources because fewer packets need to send so also less header data need to be sent. But, this characteristic causes delay to data transportation and is not suitable for real-time data transportation [6].
[14, 5, 3]

4.3.3 User Datagram Protocol (UDP)

User Datagram Protocol (UDP) [31] resides on the transport layer and the service that UDP provides is connectionless and unreliable data transportation. Although UDP provides the same data transportation service than IP, the additional services that UDP offers are ports and checksum calculation over the data area of the UDP packet. If reliable transport is needed it must be handle at the application level. UDP can be used for unicast, multicast and broadcast connections. In Figure 4.6 the format of the UDP header is shown.
[31, 14]

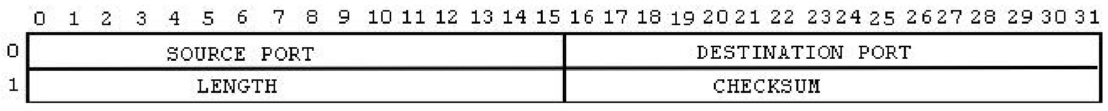


Figure 4.6: Format of UDP header

The SOURCE PORT DESTINATION PORT fields define the endpoints of the sender and the receiver. In principle, the use of the SOURCE PORT is optional. Because UDP does not use the abstraction of connection, the size of a port field sets the limit to the number of simultaneous data transportation. The LENGTH field indicates the size of the whole UDP packet. The CHECKSUM field contains the computed checksum over the UDP packet but its use is optional.
[31, 14]

The unreliable data transportation that UDP offers is suitable to real-time data transportation because data is not retransmitted so additional jitter is not caused [10]. In addition, UDP does not buffer data so delay that buffering can cause is avoided.

4.4 Tunneling

Tunneling is a technique to transport a packet of some protocol as a data portion of some other protocol [5, 14]. As a result, the reason for tunneling is that without it the packet of the former protocol could not otherwise be transported because the network does not support its use. But, because the latter protocol is supported data transportation can be achieved and the data of the unsupported protocol can be delivered from the sender to the receiver. For example, if the network does not support transportation of UDP packets but supports transportation of TCP packets, with tunneling UDP packets are transported in the data portion of TCP packets so that they can be delivered from the sender to the receiver.

4.5 Summary

In this chapter different characteristics of IP-based networks and transport protocols were considered. The most important concept was the IP address that enables the use of the TCP/IP protocol stack regardless of the hardware. In addition, concept of protocol stack and also packet construction and deconstruction were represented. The most important protocols described were IP, TCP and UDP. The choice between TCP and UDP should depend on the application because these offer different kinds of services with different kinds of characteristics. Also the unsuitable real-time data transportation properties of TCP compared to UDP were considered. Finally, a technique called tunneling, which can be used if some protocol such as UDP is not supported, was described.

5 Improvement of Current Transport Protocols

Fake tunneling is the concept that the author is developing as the solution to the research problem stated in Chapter 2. Chapter 3 has described the requirements for the design. Chapter 4 has described the current technology and the general constraints of it. The purpose of the author is to improve the current technology with the concept of Fake Tunneling that the author has invented.

The objective of this chapter is to describe the development process of Fake Tunneling. First, the motivation for such a concept is given. After that, the definition of Fake Tunneling is given. Finally, the design of Fake Tunneling is described. Basically, the objective in development of Fake Tunneling is to investigate in which way the characteristics of current transport protocols could be combined to achieve a new protocol that would be more suitable for real-time data transportation so that a solution to the given problem could be developed. In addition, the solution should take the general constraints and the requirements of the current technology into account.

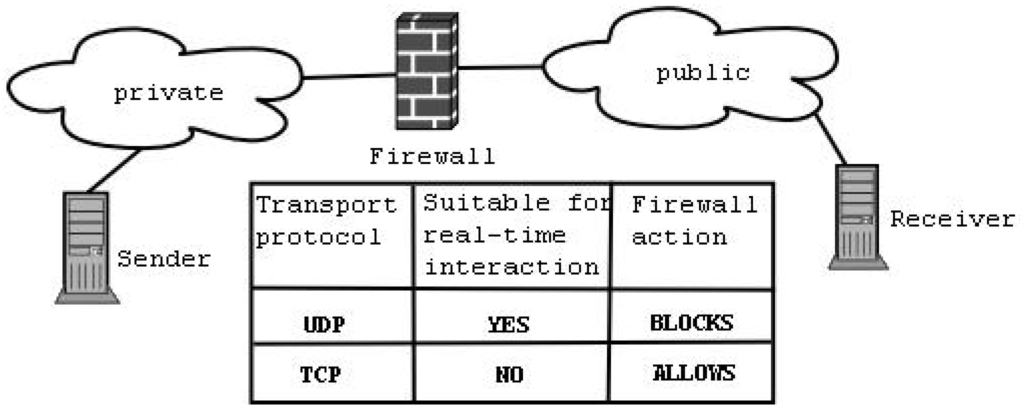
5.1 Motivation

Compared to TCP the characteristics of UDP are more suitable to real-time data transportation [5]. First, data is not buffered but is sent immediately. Secondly, acknowledgments or retransmission are not used so there are no additional delay or jitter. Finally, UDP can supports multicast and broadcast that enable transporting the same data to multiple users at the same time. As a result, UDP should be used instead of TCP in real-time interaction.

To secure the network of an organization or an individual, it can be separated with the firewall from the other networks such as the Internet. The own network to be protected can be considered as private and secure while the other networks can be considered as public and insecure. Although the purpose is to block malicious data transportation from the public to the private network, this can set limitations also to the opposite direction, namely from the private to the public network. For example, all data transportation that uses UDP can be denied and only data transportation that uses TCP is allowed. As a result, the use of UDP and also all the protocols that use it, such as Real Time Protocol (RTP) [32], is impossible if the receiver is beyond the firewall [7].

To circumvent restrictions of firewalls, tunneling UDP with TCP would be possible. But, the characteristics of TCP, namely the buffering and reliability achieved with acknowledgment and retransmission, are not suitable for real-time data transportation [5, 6]. Although the buffering can be neglected and the window size of the TCP header can be changed, the design of TCP remains. That makes TCP unacceptable choice to real-time interaction. If Quality of Service (QoS) could be guaranteed, TCP could be suitable enough but that is not the case in the current Internet, which cannot guarantee QoS in general [4].

In Figure 5.1 is shown the conceptual picture of the problems that UDP and TCP cause when the sender should achieve real-time data transportation beyond the firewall to the receiver. As shown in the figure, the sender is attached to the private network and the receiver is attached to the public network. In addition, the networks are separated from each other with a firewall. Because the firewall is the only path between the sender and the receiver, all the data are passed through it. As a result, the firewall can block all data transportation that is considered malicious. The problems arise when data transportation that uses UDP is blocked. On the other hand, the benefits of UDP cannot be used because of the firewall and on the other hand, the drawbacks of TCP make it unsuitable although TCP could otherwise be used.



Transport protocol	Suitable for real-time interaction	Firewall action
UDP	**YES**	**BLOCKS**
TCP	**NO**	**ALLOWS**

Figure 5.1: Problems with both UDP and TCP

5.2 Definition of Fake Tunneling

So, the restrictions of both UDP and TCP, namely the use of UDP with firewalls and the lack of real-time data transportation properties of TCP, make them unsuitable to real-time data transportation through firewalls when real-time interaction should be achieved [5, 6, 7]. The solution to these restrictions is the concept of Fake Tunneling that the author is developing. So that the definition of Fake Tunneling to be used on the IP-based networks was meaningful, the constraints of the current technology and also network components and existing applications are considered. In addition, the fundamental solutions to these constraints are introduced. First, the whole point of developing Fake Tunneling is the service that can be achieved. This service is the real-time data transportation so that real-time interaction between the users of a multi-user virtual environment (VE) can be achieved. Because TCP cannot but UDP can provide this service the characteristics of UDP makes it more appealing choice compared to TCP. Based on the characteristics of UDP and TCP, the data transportation characteristics of the new protocol should be closer to UDP than TCP. The important conclusion is that the new protocol should achieve data transportation similar to UDP. Secondly, in spite of the fact that in general any transportation protocol can be used in the private network, the public network allows only the use of certain protocols. So, compared to the private network, the public network sets the stricter limitation to the transportation protocols that can be used. In addition, the firewall can further limit the use of transport protocols. Based on these constraints the form of the new protocol must be identical to the supported transport protocol namely TCP because in general firewalls block rather UDP than TCP data transportation. The important conclusion of making the new protocol to look like TCP is that none of the network components can distinguish Fake Tunneling from TCP. Also, the following conclusions are achieved and named as the fundamental principles of Fake Tunneling:

<u>The fundamental principles of Fake Tunneling:</u>

1. UDP like data transportation
2. TCP like form

As a result, because the properties of pure TCP or pure UDP are not adequate enough, Fake tunneling combines the benefits of both TCP and UDP to achieve data transportation that is suitable for real-time interaction. This is denoted with the conceptual definition of Fake Tunneling given in Formula (1). The formula states that Fake Tunneling consists of a combination of UDP and TCP but instead of using the current protocols, a completely new protocol is to be created. The term Fake Tunneling denotes the fact that although in principle UDP is tunneled with TCP in reality this is not the case as will be shown. But instead of using the term virtual the term fake is chosen because only the user but not the observer of Fake Tunneling can say that the tunneling is not real.

Fake Tunneling = combination of (UDP + TCP) **(1)**

Although the fundamental principles of Fake Tunneling are adequate it should be noted that the application uses the current transportation protocols through the certain interfaces that the implementation of the protocol offers. Further, the implementation of the protocol uses the network through the certain interface that the operating system (OS) provides. As a result there are at least two interfaces between the application and the network. Although the former interface is dependent on the implementation, the properties of UDP and TCP set limits to the characteristics that the interface offers. The latter interface is dependent on the OS although the same kind of abstraction is often used among OS. In addition the latter interface is harder to affect than the former interface. Although the interfaces are not arbitrary, they are not uniform either. Based on these constraints the new protocol is dependent on the applications and the OS. As a result, the new protocol should minimize dependencies of the different applications and OS but take advantage of the interfaces that the current protocols use.

In Figure 5.2 the idea of Fake Tunneling is described as conceptual protocol stacks. In the resulting combination UDP offers services to the application layer and TCP offers services to the Internet layer. As a result, the interfaces that the two main transportation protocols of the TCP/IP stack provide are used to utilize Fake Tunneling. Thus, the application only interacts with UDP and the network only interacts with TCP.

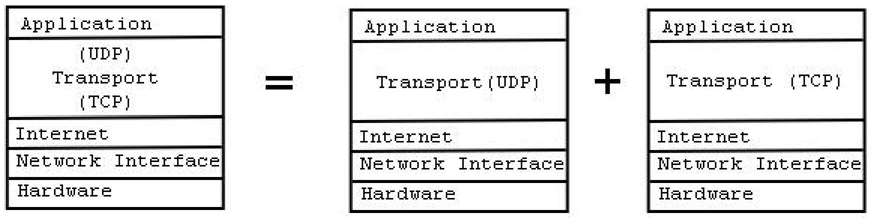
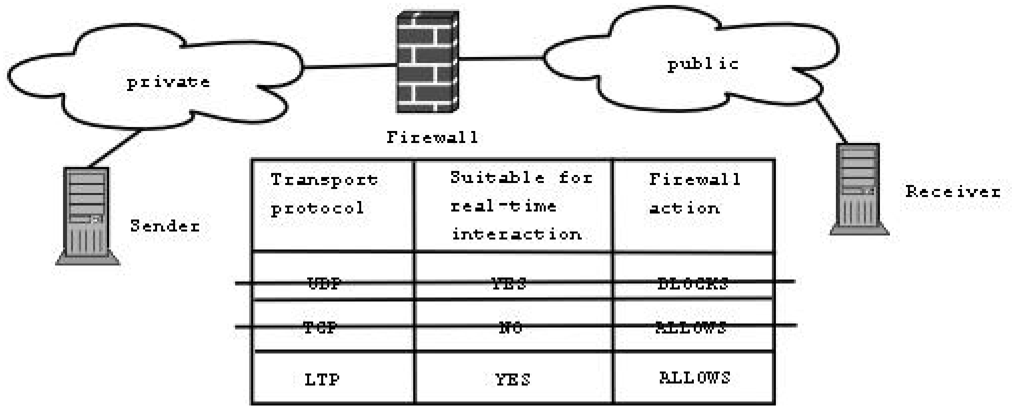


Figure 5.2: Conceptual definition of Fake Tunneling

As a result of the definition of Fake Tunneling, the solution is given in Figure 5.3 to the problems of the current transport protocols shown in Figure 5.1. Because Fake Tunneling is used, both UDP and TCP have become obsolete. In Figure 5.3, LTP denotes the protocol that is used with Fake Tunneling. LTP is the acronym for Lume Tunneling Protocol used with Fake Tunneling. Lume is a Finnish word for the English word fake. Obviously, FTP would be an unsuitable acronym because on IP-technology FTP denotes File Transfer Protocol. LTP combines the benefits of both UDP and TCP so that LTP achieves real-time data transportation and the firewall does not block the data transportation.



Transport protocol	Suitable for real-time interaction	Firewall action
~~UDP~~	~~YES~~	~~BLOCKS~~
~~TCP~~	~~NO~~	~~ALLOWS~~
LTP	YES	ALLOWS

Figure 5.3: Solution to the problems of both UDP and TCP

5.3 Design Principles

The most important design choices include the tools and also Application Programming Interfaces (API) of Fake Tunneling that are going to be used. It should be noted that, some or all of these could be replaced with another technology without affecting the concept of Fake Tunneling. In addition, implementation issues like use of another OS can require some modifications. Also, some of them, such as the socket abstraction is widely accepted and used that replacing it could be deprecated. Nevertheless these design principles here provides feasible methods for the design and implementation.

- The design is done according to the current Internet protocols. As a result, there is no reason why the use of Fake Tunneling would not be possible with current protocol implementations. The significant consequence is that the use of Fake Tunneling is possible without requiring that both the sender and the receiver use it but either one of them can use some TCP protocol implementation.

- The architecture of Fake Tunneling is designed with Unified Modeling Language (UML) [33]. Although some other modeling method could also be chosen, wide acceptance and suitability of it in modeling of software development make it a reasonable choice.

- With the use of the socket abstraction, namely Berkeley Software Distribution (BSD) socket and its derivations, a uniform interface between LTP and the network is achieved. Although not all of the OS support use of the socket, the most popular OS of this moment do. So, there is no reason to try something different although any interface between LTP and the network can be used. As a result, the socket is used as the interface to the network and the Socket API that the OS offers is utilized.

- The TCP/IP protocol stack does not define a certain interface to be used between the application and protocol but the OS usually provides it as a part of the Socket API. With the use of the UDP API a uniform interface between the application and LTP is achieved. It should be noted that because any UDP API can be used, a use of a specific UDP API is not relevant in the design phase. As a result, a generic UDP API is used in the design phase and a more specific one will be used in the implementation phase.

It should be noted that because the Socket API that the OS provides offers the interface to the use of the network, it also offers use of the UDP API related concept. As a result the Socket API and UDP API are related to each other. In addition some software library can be used between the application and the OS, as a result, the application uses the Socket API through the software library [14]. In this design the aspects of packet processing between the application and LTP relates to the UDP API and aspects of packet transporting over the network to the Socket API.

5.4 Architecture Description

In Figure 5.4 the conceptual architecture of Fake Tunneling is shown as a logical view. Fake Tunneling components communicate through the following standard APIs. First, the application and LTP communicate through the UDP API. Secondly, LTP and the physical network communicate through the Socket API. As a result, the implementation of LTP can be done independently of the application and the physical network.

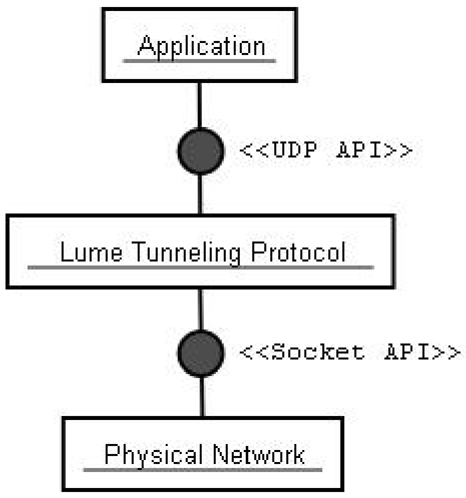


Figure 5.4: Conceptual architecture of Fake Tunneling

The details of the UDP API and the Socket API are considered later and instead conceptual descriptions are used. The reasons for that are that a detailed description of the APIs is not relevant yet. Also, the details of both the UDP API and the Socket API can be considered as implementation specific issues that should be omitted in the design phase of the development. Because the OS usually provides the Socket API and the UDP API as part of the Socket API, they are dependent on the OS. Although, some common abstraction layer can be specified between the application and the protocol layers. In spite of the fact that the interfaces can be OS specific, the basic principles can be thought to be quite similar among different OS.

5.4.1 Connecting entities

In Figure 5.5 the connection between entities, such as sender and a receiver, is shown with the conceptual architecture of Fake Tunneling. It should be noted that neither the sender nor the receiver has to know the details or anything about the physical network or how they are actually connected. In the same way they do not have to know the details or anything about LTP. All that the entity has to know is what services the UDP API offers are how they are used. On the other hand, LTP has to know what services both the UDP API and the Socket API provides and how they are used.

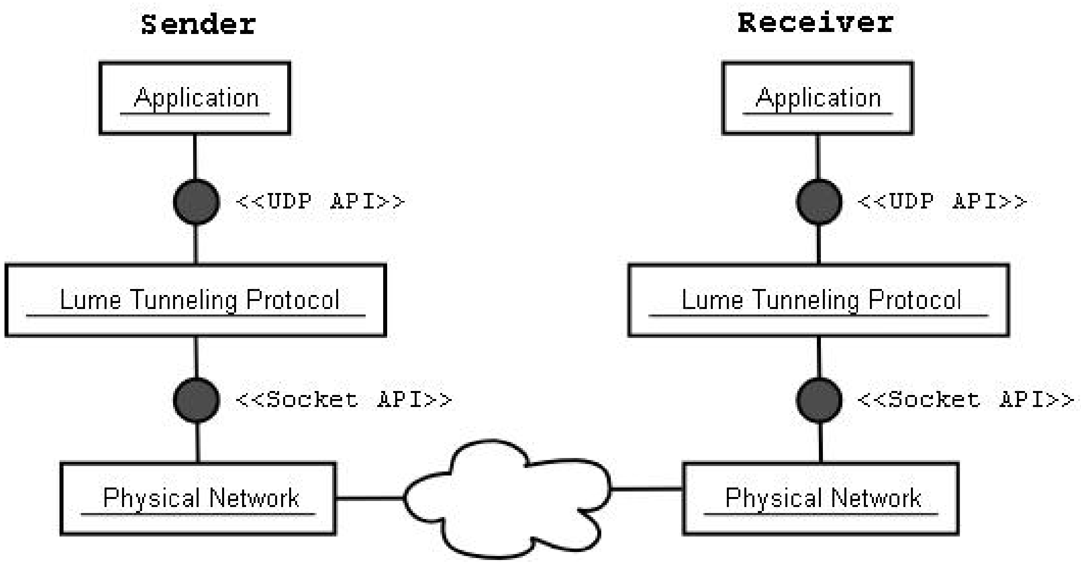


Figure 5.5: Connection of parties with conceptual architecture of Fake Tunneling

5.4.2 Fake Tunneling Interfaces

In Figure 5.4 and Figure 5.5 are described the interfaces of Fake Tunneling. In addition, in Figure 4.6 a format of the UDP header and in Figure 4.4 a format of the IP header are shown. The UDP API and the use of it are declared based on these formats. Instead, the formats are irrelevant to the use of the Socket API because it is used to send and receive arbitrary sequence of bytes and the interpretation of the bytes is left to LTP.

The UDP API contains methods to set and get values of the IP and UDP header fields and also methods to set and get data of the UDP packet data portion. The services of the UDP API that the entity uses are identifying of itself with a source IP address and a source port. In addition, the entity uses UDP API services that enable setting data to other entities and getting data from other entities based on the destination port and the destination IP-address of the other entity. The definition of the source IP address and source port can be implicit or explicit. In addition, the definition of the source IP address is based on the network configuration. If the entity does not declare the source port explicitly, it can be defined implicitly and as a result any available port can be given. The implicit source port declaration is usually sufficient for the sender because the receiver does not usually care from which port the packets are sent. But, usually the receiver declares its source port explicitly because the sender needs to know to which port the packets should be sent. To summarize, usually the UDP API provides at least the following conceptual services for the sender and the receiver.

The conceptual services of the UDP API:

- setData
- getData
- setIPAddress
- getIPAddress
- setPort
- getPort

Because LTP is used on the other side of the UDP API, it also has to manage the conceptual services that the UDP API offers. In addition, LTP has to use the Socket API. As a result, LTP must be able to send data to the physical network and also receive data from the physical network. Indeed, the Socket API contains services to send and receive arbitrary sequence of bytes that are considered as packets. So as a result, the Socket API is used to send packets to the network and receive packets from the network. In addition, Socket API is used when LTP takes care of the implicit IP-address and port declaration and also of the implicit filling of the fields of the headers that the entity does not declare explicitly. To summarize, the Socket API should provide at least the following conceptual services for LTP.

The conceptual services of the Socket API:

- sendPacket
- receivePacket

5.5 Lume Tunneling Protocol (LTP)

LTP interacts with the application through the UDP API and with the network through the Socket API as shown in Figure 5.4. This specifies the way data is passed through the interfaces to and from LTP. In addition, the specifications of LTP data processing are shown In Formula (1) and Figure 5.2. So, Fake Tunneling states that application uses UDP and network uses TCP and that there is a mapping between UDP and TCP. As a result, LTP takes care of this mapping as shown in Figure 5.6.

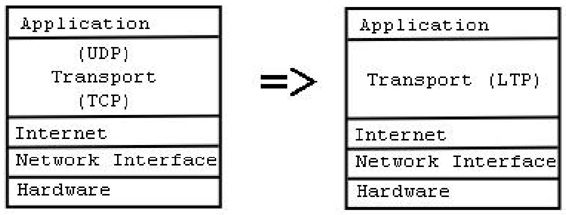


Figure 5.6: Fake Tunneling protocol stack

5.5.1 LTP Datagram Basics

It should be noted that in general the details of the packets are not the primary interest of the entities, such as the sender and the receiver. Although, it is an important part of the mechanism that enables transmitting data between the entities. Nevertheless, because the entities use the UDP API, they can also thought that they are actually using the UDP header shown in Figure 4.6 and also the packet shown in Figure 5.7. Indeed, from the application point of view this is the correct assumption. The figure shows the virtual packet of Fake Tunneling when IP is used as the underlying transport protocol.

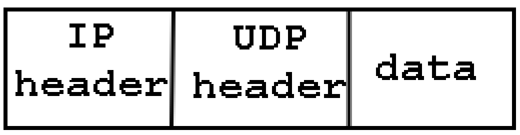


Figure 5.7: Virtual packet of Fake Tunneling

But, Fake Tunneling does not use the UDP but TCP header in the packet. The TPC packet consists of a TCP header, which is shown in Figure 4.5, and a data portion. When LTP sends data, it fills the fields of TCP header and the data portion with values retrieved from UDP API. Correspondingly, when LTP receives data it uses the UDP API to set values retrieved from the TCP header and the data portion. As a result, instead of using the virtual packet shown in Figure 5.7, the packet shown in Figure 5.8 is used in reality. The figure shows the real packet of Fake Tunneling when IP is used as the underlying transport protocol. It should be noted that although in general and also in this design the IP protocol is used as the underlying transport layer, there is no reason why any other protocol could also be used. But, because IP is used, LTP also fills the fields of the IP header shown in Figure 4.4 in addition to the fields of the TCP header.

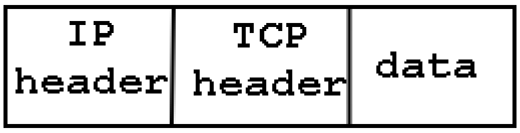


Figure 5.8: Real packet of Fake Tunneling

So, LTP takes care of the mapping between the UDP API and the Fake Tunneling packet and the conceptual services of the UDP API are used. The purpose of this mapping is to enable the use of the TCP header instead of the UDP header and in Figure 5.9 the actual mapping is shown. The related fields are connected with lines that have dots at the ends.

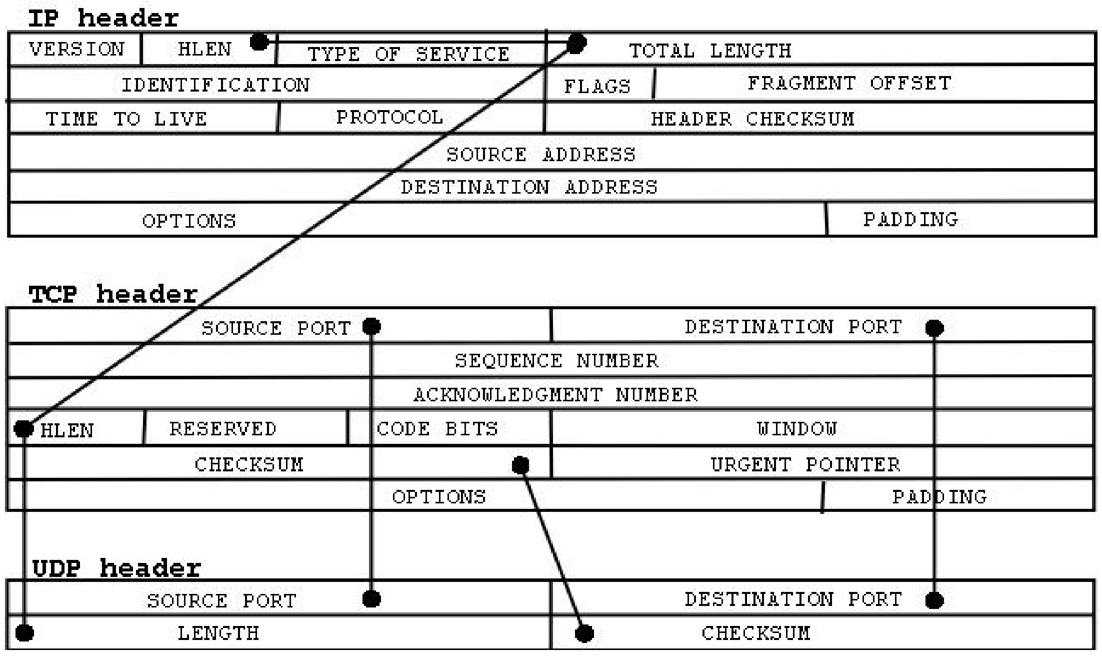


Figure 5.9: Mapping between fields of UDP and Fake Tunneling headers

Because the TCP header contains fields for all the data that can be represented with the UDP header, the mapping between the headers is quite straightforward. In addition, the IP header is used as usual. The SOURCE PORT and the DESTINATION PORT fields of the UDP header have direct counterparts in the TCP header. In addition, the LENGTH field of the UDP header contains the sum of that consist of the length of the UDP header and of the data. So, the UDP header length portion is omitted and the data length portion is included into the TOTAL LENGTH field of the IP header that also includes HLEN fields of both the TCP and IP header. Finally, the CHECKSUM field of the TCP header is similar to the CHECKSUM field of the UDP header and is calculated in the same way.

When the sender sends a packet to the receiver, the conceptual services of the UDP API are exploited in the following way. First, the sender uses the service setData to set the data portion of the packet shown in Figure 5.8 and the receiver uses the service getData to get the data portion of the packet. Secondly, the sender uses the service setIPAddress to set the IP-address of the receiver into the DESTINATION ADDRESS field of the IP-header and the receiver uses the service getIPAddress to get the IP-address of the sender from the SOURCE ADDRESS field. Finally, the sender uses the service setPort to set the port number of the receiver into the DESTINATION PORT field of the TCP header and the receiver uses the service getPort to get the port number of the sender from the SOURCE PORT field. Because LTP can implicitly fill the rest of the fields, this described the use of the packet and the services that the UDP API provides is sufficient. Especially, LTP must make sure that the PROTOCOL field of the IP-header corresponds to the value that indicates that the data portion of the IP datagram is TCP not UDP. As a result, LTP has provided the mapping between UDP and TCP. As a result, the design of the first and second item of the fundamental principles of Fake Tunneling has been declared and can now been implemented.

5.5.2 LTP Datagram Advanced

Because the UDP header is very simple containing just a few fields [31], the connection between the application and LTP was quite straightforward. Although the current implementations of the TCP and IP layers can be exploited, LTP itself can also offer the necessary services of these portions. Integrating the TCP and IP layers can take advantage of the data locality in memory [15, 16]. Also, the values of the specific fields can remain constant so they need to be set only once and are then omitted. As a result, unnecessary data manipulation is avoided that often results increased execution speed. Based on the conceptual definition of Fake Tunneling shown in Figure 5.2 and the Fake Tunneling protocol stack shown in Figure 5.6 this integration is described in Figure 5.10.
[3, 14]

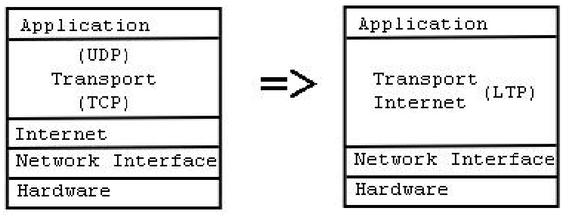


Figure 5.10: Integration of Fake Tunneling layers

When the IP header is used without options, the following fields of the IP header are considered: TOTAL LENGTH, IDENTIFICATION, HEADER CHECKSUM and DESTINATION ADDRESS. The rest of the fields need to be set only once. In Figure 5.11 the IP header of this kind of use is shown. In addition, when the TCP header is used without options, the following fields of the TCP header are considered: DESTINATION PORT, SEQUENCE NUMBER, ACKNOWLEDGEMENT NUMBER and CHECKSUM. The rest of the fields need to be set only once. In Figure 5.12 the TCP header of this kind of use is shown. Another benefit of having many constants fields affects the fact that calculation of checksums can be faster in general.

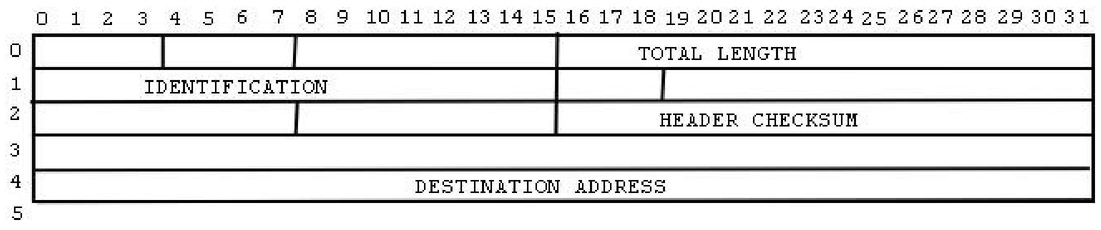


Figure 5.11: Smart use of IP header with Fake Tunneling

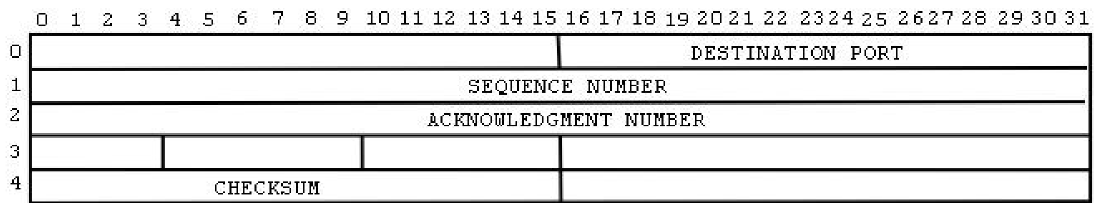


Figure 5.12: Smart use of TCP header with Fake Tunneling

If packets are sent only to a certain destination, the DESTINATION ADDRESS and DESTINATION PORT fields can be omitted because then they need to be set only once. In addition, the use of the fields IDENTIFICATION, SEQUENCE NUMBER and ACKNOWLEDGEMENT NUMBER can possible be omitted if a) the security restriction allow that and b) the both parties of the data transportation agree. So, the former is affected by firewalls and the latter is affected of the implementation details of the entities. If fixed size packets are used also the field TOTAL LENGTH has to be set only once. Also, if the TOTAL LENGTH, IDENTIFICATION and DESTINATION ADDRESS fields can be omitted also the CHECKSUM field of the IP header can be omitted. In addition, if the DESTINATION PORT, SEQUENCE NUMBER and ACKNOWLEDGEMENT NUMBER fields can be omitted the CHECKSUM field of the TCP header is the only one that is accessed. If all but the CHECKSUM field of the TCP header can be omitted as described, the optimum accessing of the IP and TCP headers fields is reached. It should be noted that Fake Tunneling could exploit all these features although it is not mandatory

5.5.3 Checksum Calculation Speed Up

Now as the constant and volatile fields of the Fake Tunneling design have been declared, the checksum calculation speed up is considered. The formal algorithm to the checksum calculation is the following. Treat the input as chunks of 16-bit integers using one's complement arithmetic and take one's complement of the result [29, 30] According to the algorithm, the IP checksum is calculated so that input is IP header. TCP checksum is calculated with the algorithm so that input is TCP Pseudo header, TCP header and data portion. The TCP pseudo header is shown in Figure 5.13. But instead of using such a formal algorithm, which involves copying of memory, execution of loops and accessing all of the fields, the checksum calculation is done in the following way. First, the sum of the constant fields is calculated only once because they always produce the same value. As a result the constant fields have to be accessed only once. Secondly, the pseudo header is omitted so that memory copying can be avoided. Instead the use of the pseudo header is integrated to the checksum calculation as the following phase describes. Thirdly, the common fields of the checksums are exploited at the same time to minimize atomic operations and memory traffic.

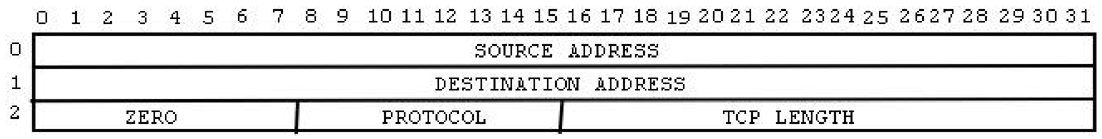


Figure 5.13: TCP pseudo header

5.5.4 Connection Establishment and Closing

When a sender sends packets, the receiver must be ready to receive those packets or they are lost. TCP defines the establishment of connection opening defined as the three-way handshake. During the three-way handshake three packets are transmitted between the entities namely the sender and the receiver. Although, TCP normally uses a similar mechanism to close connection than to open it, it also defines connection reset. The connection reset is simpler that connection closing. It should be noted that UDP does not provide any mechanism to open or close the connection because UDP does not use such an abstraction. As a result, if such a mechanism is needed with UDP, it must be done at the application layer. On the other hand, because UDP does not define connection opening, the application cannot according to UDP require such a mechanism. But, on the other hand TCP requires such a mechanism so according to the conceptual definition of Fake Tunneling shown in Figure 5.2 such a mechanism is required. Based on the latter argument it is strongly suggested that LTP would use connection opening. Also, some firewalls and OS can check that the connection is actually established. Based on these arguments the use of connection opening is mandatory. It should be noted that the application can also benefit from the connection-opening behavior of TCP. This is useful if the application wants to make sure that a route between the sender and the receiver is established before any payload data is sent. Instead of connection closing, connection reset can be used because of its simplicity. It should be noted that establishment of the connection has to be done only once so it does not produce additional overhead after to data transportation. Use of connection opening mechanism enables the use of LTP with the current implementations of TCP. As a result, use of Fake Tunneling is possible without knowing the implementation details of the other entity beforehand. This also enables solving of the implementation details of the other entity at run time. This could be done for example with a use of some certain identifier. Another possibility to figure out the implementation details of the other party would be to first omit the use of the IDENTIFICATION, SEQUENCE NUMBER and ACKNOWLEDGEMENT NUMBER fields and then inspect how the other entity reacts. Because the conceptual services of the UDP API allows setting of a different receiver each time a packet is sent, LTP has to take care that the connection is not reopened before it is closed.

5.5.5 Data Transportation Types

TCP is used for unicast connection only but UDP can also be used to multicast and broadcast connections. Because Fake Tunneling supports UDP interface to the application it can be suggested that LTP should be able to offer also multicast and broadcast connections.

5.5.6 Combination of Positive and Negative Acknowledgments

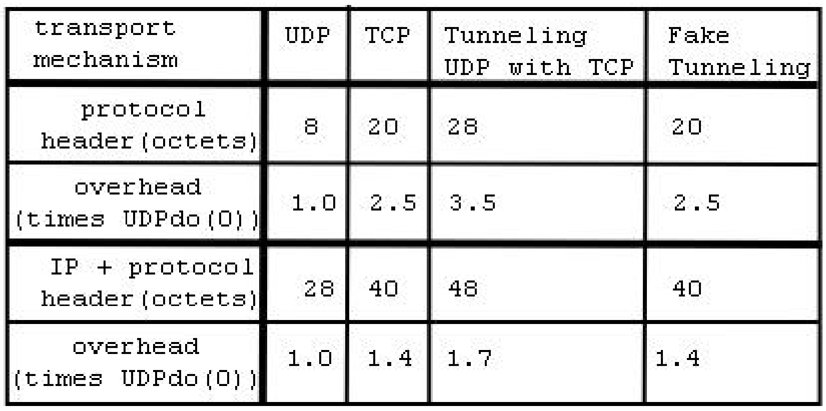
Instead of using the positive acknowledgment, like TCP does, LTP can take advantage of the negative acknowledgment. When the positive acknowledgment is used, the receiver acknowledges the sender if a packet is received. The negative acknowledgment works in the opposite way. When the negative acknowledgment is used, the receiver acknowledges the sender if a packet is not received. The benefits of the negative acknowledgment compared to the positive acknowledgment are that the sender will not be overloaded from the acknowledgments. In general, this has a great importance especially when multicast or broadcast is used so Fake Tunneling can exploit the combination of positive and negative acknowledgments.
[3]

5.6 Comparison of Theoretical Characteristics

In addition to the use of TCP or Fake Tunneling, another solution would be tunneling UDP with TCP to enable transmission of UDP packets beyond firewalls. So the theoretical characteristics of these different kinds of data transportation mechanism are compared. First, protocol header data overhead and after that protocol header access overhead is compared. The purpose of these comparisons is to give a uniform view of the theoretical characteristics so that the theoretical differences between the different data transportation mechanisms can be compared.

5.6.1 Protocol Header Data Overhead

The protocol causes overhead to data transportation because not all of the transported data is payload data but includes also header data. This is denoted as protocol header data overhead and the results of the comparison between different data transportation mechanisms are shown in Figure 5.14. In the figure the field *transport mechanism* specifies the different data transportation mechanisms and includes UDP, TCP, tunneling UDP with TCP and Fake Tunneling. The field *protocol header* indicates the size of the protocol header of each transport mechanism measured in bytes. The field *IP + protocol header* corresponds to the field *protocol header* but also the size of the IP header is included. The field *overhead* indicates the header data overhead that is introduced because part of the transported data contains header fields. The term UDPdo(N)), in which the substring 'do' denotes data overhead, indicates the header data overhead of UDP with a payload data size N, especially, UDPdo(0) is equal to 1.0. The header data overhead of a transport mechanism is calculated with Formula (2), which returns a multiple of the UDP header data overhead compared to UDPdo(0). The number eight in the nominator of the formula indicates the protocol header size of the UDP transport mechanism in bytes. Although firewalls can block UDP data transportation, the UDP header data overhead is used as a reference because it causes header data overhead least of all. The comparison is made without the use of the protocol options fields of the headers. The protocol option field of the TCP header is not included because although the size of the option field can vary, it can be considered as a constant because its effect is same for all other but UDP, which does not use such a field. In addition, the protocol option field of the IP header is not included because the effect of it is same for all the data transportation mechanisms.



transport mechanism	UDP	TCP	Tunneling UDP with TCP	Fake Tunneling
protocol header (octets)	8	20	28	20
overhead (times UDPdo(0))	1.0	2.5	3.5	2.5
IP + protocol header (octets)	28	40	48	40
overhead (times UDPdo(0))	1.0	1.4	1.7	1.4

Figure 5.14: Comparison of protocol header data overhead

$$overhead = \frac{IP + (UDP \vee TCP)}{IP + 8}, IP = (0 \vee 20) \wedge UDP = 8 \wedge TCP = 20 \qquad \textbf{(2)}$$

Based on the second item of the fundamental principles of Fake Tunneling and the Fake Tunneling packet shown in Figure 5.8 it should be obvious that the comparison of protocol header data overhead gives equal values to both LTP and TCP. Compared to the tunneling UDP with TCP the mapping shown in Figure 5.9 declares a more efficient way that is used with LTP. The reasons for that are that the UDP header does not need to be included explicitly to the data portion of the TCP packet but implicitly to the TCP and IP headers.

The header data overhead as a function of packet size in bytes is shown in Figure 5.15 when IP is used as the underlying transport protocol and the UDPdo(N) is used as the reference value. LTP is equal to TCP and UDP+TCP indicates the use of tunneling UDP with TCP. Note that because in the figure the reference value is UDPdo(N), the overhead of UDP is constant. According to the figure, the protocol header data overhead of the other data transportation mechanisms compared to UDP decreases as the size of the packet increases and finally is in practice equal to UDP.

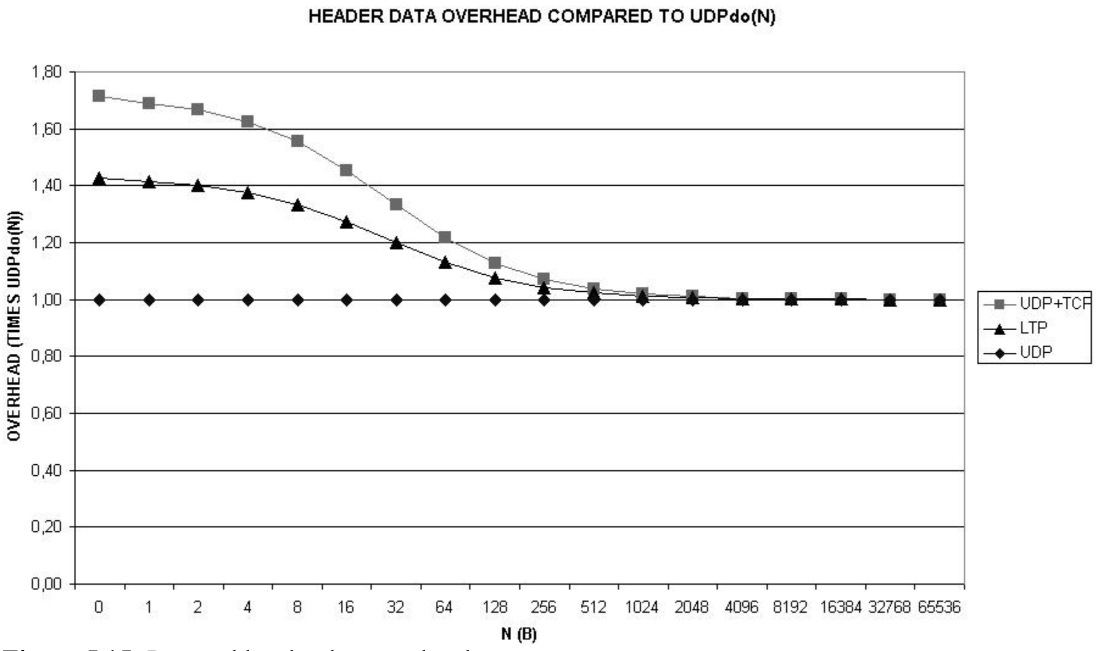
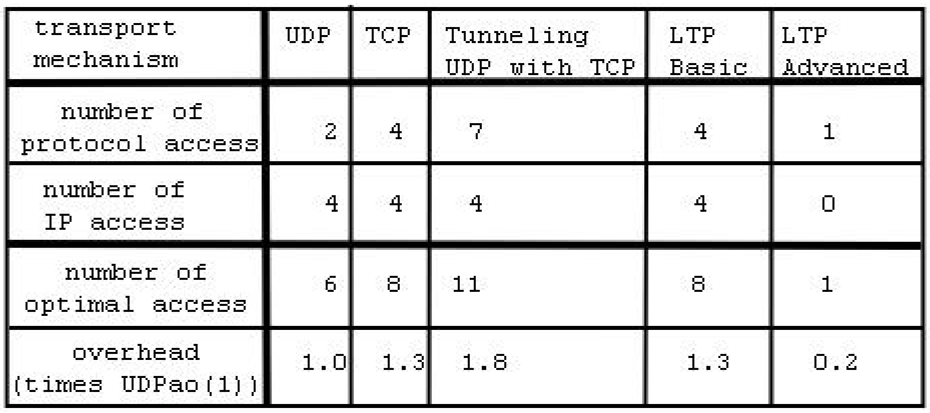


Figure 5.15: Protocol header data overhead

5.6.2 Protocol Header Access Overhead

Values of the fields of a protocol header need to be set and get. This causes field-accessing overhead and is denoted as protocol header access overhead. The results of the comparison between different data transportation mechanisms are shown in Figure 5.16. Because UDP header has fields least of all, it is used as the reference. Also this comparison is made without the use of protocol options fields of the headers based on the same arguments. In the figure the field *transport mechanism* specifies the different data transportation mechanisms and includes UDP, TCP, tunneling UDP with TCP also two extreme use of LTP with Fake Tunneling, namely LTP Basic and LTP Advanced. The field *number of protocol access* indicates the number of the individual fields of the transport mechanism that at least have to be accessed. The field *number of IP access* corresponds to the field *number of protocol access*. Because it relates to the IP header, the values are same for all data transportation mechanisms. The field *overhead* indicates protocol header access overhead that is introduced because some of the header fields have to be accessed each time a packet is sent or received. The term UDPao(N), in which the substring 'ao' denotes accessing overhead, indicates the protocol header access overhead of UDP with the number of packet N, especially, UDPao(1) is equal to 1.0. The protocol header access overhead of a transport mechanism is compared to the UDPao(1) and calculated with Formula (3), which returns a multiple of the UDP protocol header access overhead. The number six in the nominator indicates the number of optimal access of the UDP transport mechanism.



transport mechanism	UDP	TCP	Tunneling UDP with TCP	LTP Basic	LTP Advanced
number of protocol access	2	4	7	4	1
number of IP access	4	4	4	4	0
number of optimal access	6	8	11	8	1
overhead (times UDPao(1))	1.0	1.3	1.8	1.3	0.2

Figure 5.16: Comparison of protocol header access overhead

$$overhead = \frac{numberOfOptimalAccess}{6} \qquad \textbf{(3)}$$

Because the values indicates number of the fields that at least have to be accessed, the values that are gotten are optimal and at least in the case of LTP Advanced optimum. As a result, the values for the protocol header access overhead of UDP and TCP can be worse. Obviously, the protocol header access overhead of TCP and LTP Basic are the same because same kind of packet handling can be used but LTP Basic is optimal. In addition, as the number of packets increases the effect of the overhead increases. The protocol header access overhead as a function of packet size in bytes is shown in Figure 5.17 when IP is used as the underlying transport protocol and the UDPao(N) is used as the reference value. LTP Basics is equal to TCP and UDP+TCP indicates the use of tunneling UDP with TCP. Because the factor of protocol header access overhead is constant as the size of the packet increases the effect is similar to all of the transport mechanism.

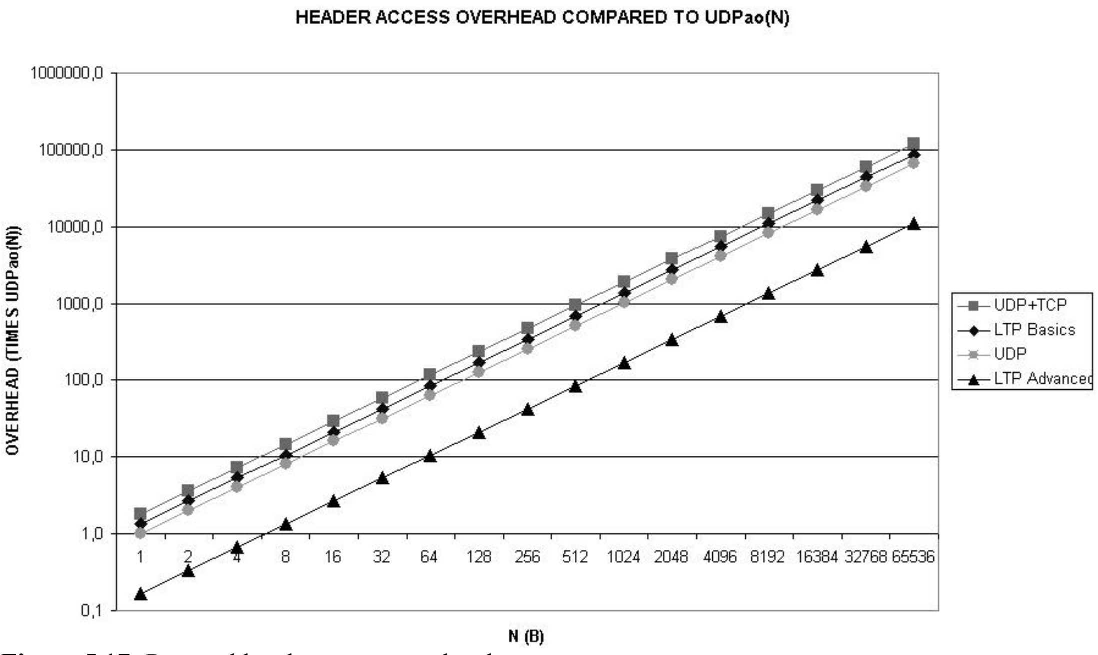


Figure 5.17: Protocol header access overhead

5.7 Summary

In this chapter concept of Fake Tunneling was introduced and design of it was described. As a result, the characteristics of UDP are used to provide the transport protocol functionality and the form of TCP is are used to allow data transportation beyond firewalls. These were defined as the fundamental principles of Fake Tunneling. Also Lume Tunneling Protocol (LTP) that Fake Tunneling uses was described. It was denoted that the architecture of Fake Tunneling exploits the UDP API and the Socket API. As a result, the conceptual services of the interfaces that Fake Tunneling uses were declared so that LTP can communicate through the interfaces with the application and the network. It was also shown how entities can be connected with Fake Tunneling. In addition, it was shown how LTP handles the mapping between UDP API and TCP packet. As a result, the design of the fundamental principles of Fake Tunneling has been declared and can now been implemented. Finally, the theoretical characteristics of different data transportation mechanisms were compared and the results were analyzed.

6 Implementation of Fake Tunneling

So far, the detailed implementation of Fake Tunneling has been omitted. For now on, different implementation issues are considered. It should be noted that Fake Tunneling does not demand any specific implementation of Lume Transport Protocol (LTP) because that is an implementation specific issue. As a result, the purpose of the described implementation is to give a concrete example of Fake Tunneling and enable practical comparison against the current protocol implementations.

In this chapter, implementation of Fake Tunneling that the author has developed and designed in the Chapter 5 is described. First, the Application Programming Interfaces (API) that are used to implement the Fake Tunneling interfaces are declared. After that the implementation of LTP that Fake Tunneling uses is described. In addition some different implementation branches are considered.

6.1 Implementation of Interfaces

LTP interacts with the application and network through the UDP and Socket interfaces. In Figure 5.2 the conceptual architecture of Fake Tunneling is shown. The detailed description of the interfaces of Fake Tunneling has been able to be omitted until this moment because the conceptual services of the interfaces were used instead of the specific ones. But, now the specific interfaces that are used in this implementation of Fake Tunneling are declared in details. Although the specific interfaces are now declared, the consideration of the too strict implementation issues is postponed for a while. The obvious reason for that is that separating the implementation from the interface is a good software development practice because it enables replacement of the implementation with another if needed without affecting the rest of the system.

6.1.1 UDP API

The choice of the UDP API can be considered to be distinct from the choice of the implementation language so any language can be used. So, although any UDP API could be used, the UDP API part of the already defined Java UDP API [34] is chosen because it defines a well-documented and clear interface. Although the substitution mechanism is also described it does not have to mean that it have to be used because not all the UDP API provide such a mechanism. The purpose of the substitution mechanism declaration is to give a concrete example how Fake Tunneling can be exploited also with the current applications. But, replacing the Java UDP API is always possible.

The UDP API part of the Java API denoted for now on as the UDP API offers the conceptual services of the UDP API to the use of the network with UDP from the application. This is available in the java.net package and the most relevant part of it is described as a class diagram shown in Figure 6.1. The DatagramSocket contains methods to send DatagramPackets to the network and receive DatagramPackets from the network. Although these offer the conceptual services of the Socket API, they are treated as a part of the UDP API that offers an abstraction to the use of the Socket API through LTP. The DatagramSocket also contains optional connect method that is used to send and receive DatagramPackets only from the specified receiver. In addition, the Datagram packet contains the methods that offer the conceptual services of the UDP API. These methods are used to set and get the application specific data and also to set and get IP address and port of the sender and the receiver.

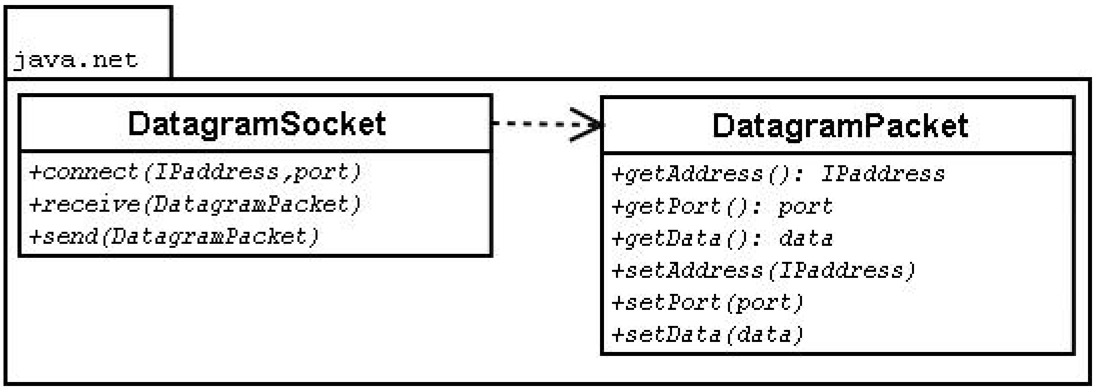


Figure 6.1: Interface between application and UDP

The UDP API defines a default transport protocol implementation of UDP, namely PlainDatagramSocketImpl that is used by default. This is described as a class diagram shown in Figure 6.2. As a result, the DatagramSocket, which is also shown also in Figure 6.1, uses the UDP implementation through the DatagramSocketImpl.

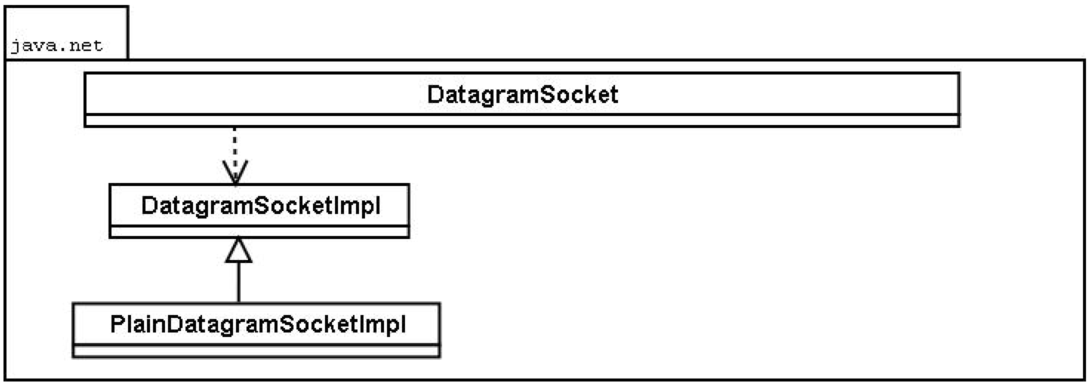


Figure 6.2: Default use of UDP API

Although a Fake Tunneling implementation can take advantage of the UDP API interface, the UDP API does not offer an implicit implementation of LTP. As a result, an explicit implementation of LTP is needed. The purpose of this implementation is to enable a mechanism to the entity, such as a sender and a receiver, to send and receive data over the network. But, instead of re-implementing the whole UDP portion of the Java API, the Java API itself provides methods to substitute the default transport protocol implementation. As a result, Fake Tunneling implementations could exploit one of these substitutions to enable the use of Fake Tunneling instead of UDP with the current applications.

One way to exploit the substitution mechanism is passing of the LumeDatagram, which is the substituting implementation to UDP provided in fi.vtt.tte.fi package, to the constructor of the DatagramSocket so that the default transport protocol implementation of UDP is replaced with the LumeDatagram. This is shown in as a class diagram in Figure 6.3. In Figure 6.4 is shown a sequence diagram related to the use of constructor parameter. Although the LumeDatagram does not implement UDP but LTP, the DatagramSocket does not notice the difference because the LumeDatagram implements the functionality that the DatagramSocket uses.

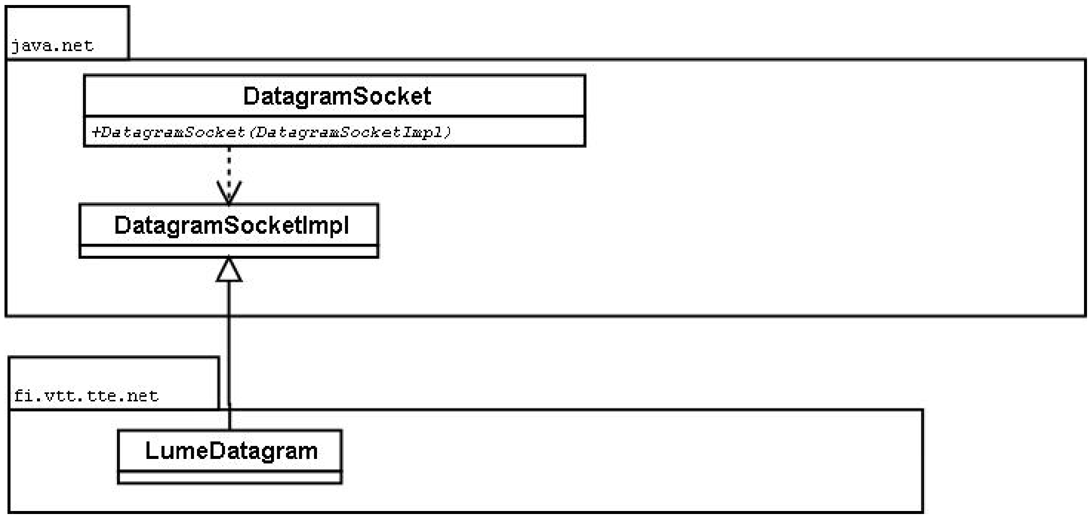


Figure 6.3: Use of constructor parameter of UDP API

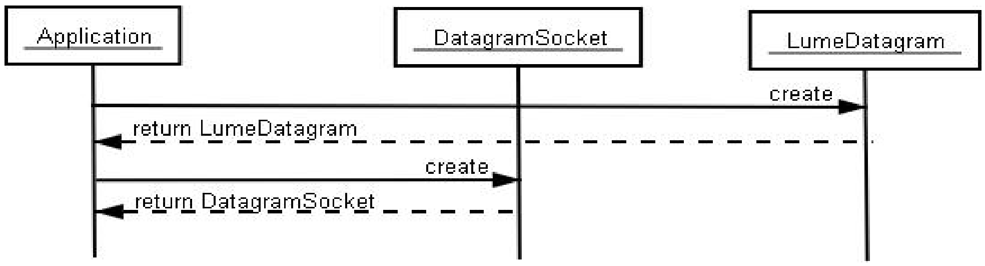


Figure 6.4: Use of constructor parameter of UDP API

The other way is shown in Figure 6.5. The DatagramSocket gets reference to the LumeDatagramFactory that in turn is used to create the LumeDatagram. Finally, the LumeDatagram replaces the default transport protocol implementation of UDP. In Figure 6.6 is shown a sequence diagram related to the use of the factory substitution.

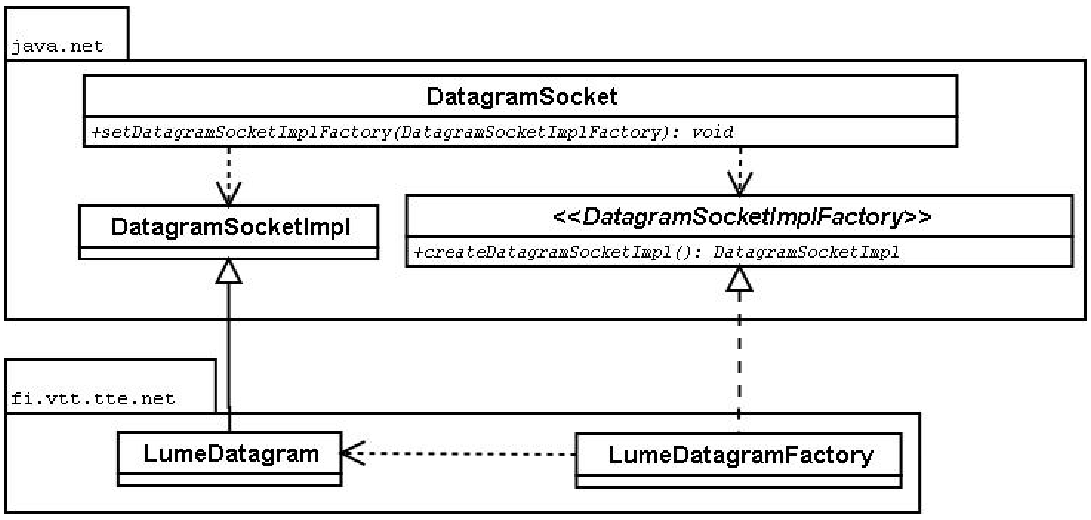

java.net
DatagramSocket
+setDatagramSocketImplFactory(DatagramSocketImplFactory): void
DatagramSocketImpl
<<DatagramSocketImplFactory>>
+createDatagramSocketImpl(): DatagramSocketImpl
fi.vtt.tte.net
LumeDatagram
LumeDatagramFactory

Figure 6.5: Use of factory of UDP API

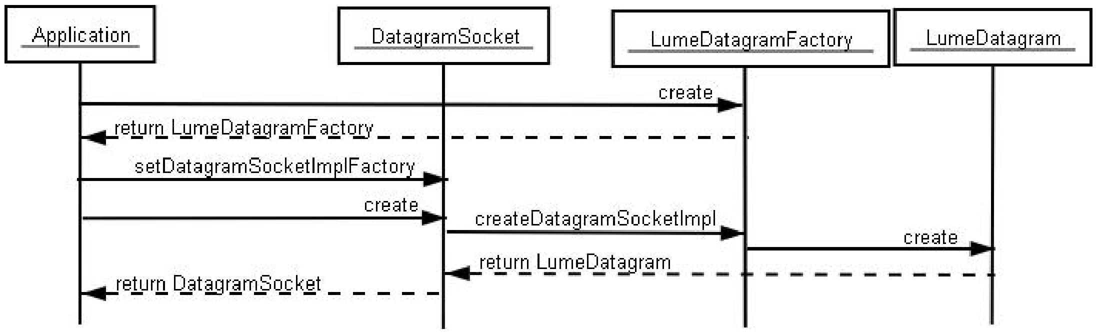


Figure 6.6: Use of factory of UDP API

Although, the Java API that is used can affect the choice which substitution to choose, nevertheless, the Java API has defined a standardized and clean way to the use of Fake Tunneling. So in addition to the interface between the application and UDP, Figure 6.1 also shows the interface between the application and LTP. As a result, the substitution mechanism that replaces UDP with LTP can also be achieved with the current applications and should be definitely used with the new implementations of Fake Tunneling if they are implemented with the API that supports such or a corresponding mechanism.

After the DatagramSocket has been created, the application can use it to send and receive DatagramPacket through the UDP API interface. In Figure 6.7 sending of data is shown. Correspondingly, in Figure 6.8 receiving of data is shown. The sending and receiving of data can be done independently of each other.

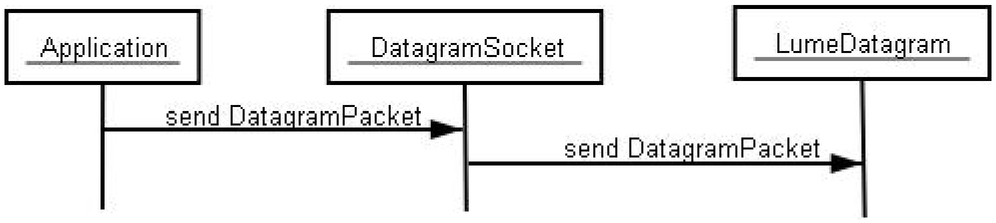


Figure 6.7: Sending data with Fake Tunneling

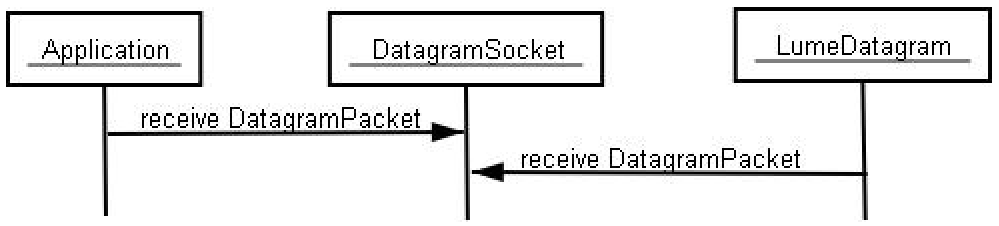


Figure 6.8: Receiving data with Fake Tunneling

6.1.2 Socket API

The socket is an abstraction that is used to enable use of network namely sending data to the network and receiving data from the network. Similar abstraction is e.g. file that enables writing data to the file system and reading data from the file system. The main difference is that file has to be identified with a name but the socket does not.

The concepts of the UDP API are related to the Socket API and are used with the Socket API or through some software library that enables the use of the UDP API part of the Socket API. Usually the operating system (OS) provides the Socket API and as a result the use of the specific Socket API makes the implementation dependent on the OS although the dependencies can be minimal. It should be noted that any OS that provides the socket abstraction to the use of network could be exploited. The specific Socket API that is chosen is the GNU Socket API [35] because it is well documented and easily available. GNU Socket API offers the conceptual services of the Socket API to the use of the network and is utilized with LTP. In addition to sending and receiving data, the GNU Socket API also contains methods to retrieve information about the sender and network. The most relevant part of this API is the following.

- s = socket(type, protocol)
- send(s, packet)
- recv(s, packet)

The socket function returns a reference to the variable that is used to access the network. The argument type of the socket function declares if reliable or unreliable socket is requested. It is also used to declare the use of raw sockets that can be exploited to bypass the default protocol implementations of the OS. This feature is exploited with Fake Tunneling. The argument protocol of the socket function declares the protocol that the socket uses, for example, UDP or TCP, but it can also represent all IP protocols. The send function is used to send packets to the network through the specific socket. Finally, the recv function is used to receive packets from the network through the specific socket.

6.2 Implementation of LTP

In addition to the use of the Fake Tunneling interfaces, LTP also constructs and deconstructs the LTP packets shown in Figure 5.8. The formats of IP and TCP headers are shown in Figure 4.4 and Figure 4.5. It should be noted that although on the Internet IP packets are used as an abstraction, the physical network transports bytes. Although, the Socket API provides methods to send bytes to the network and also receive bytes from the network, the interpretation of these bytes depends on LTP.

A sequence diagram of data sending is shown in Figure 6.9. The application uses send method of the class DatagramSocket to pass the class DatagramPacket to the underlying UDP API. After that, LTP retrieves the values of the DatagramPacket and constructs a packet. Finally, LTP passes the packet to the Socket API that sends it to the underlying network. In Figure 6.10 a sequence diagram of data receiving is shown. The functionality is contrary to the data sending.

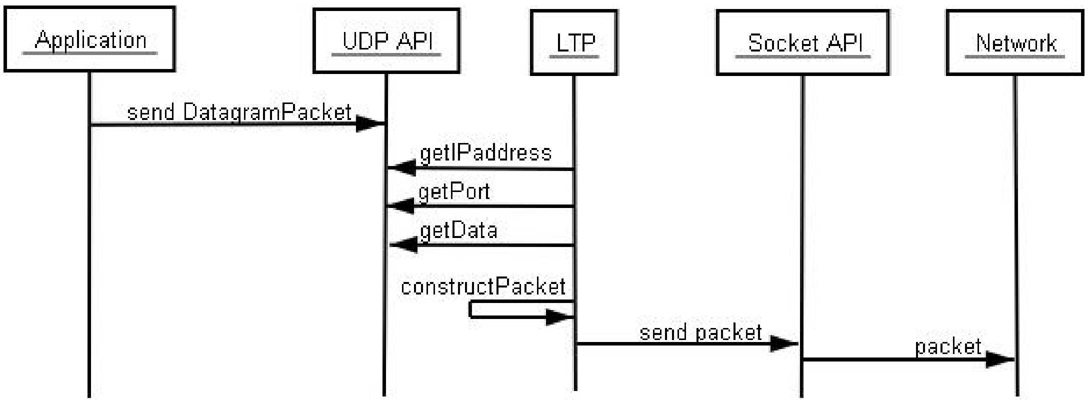


Figure 6.9: Data sending with Fake Tunneling

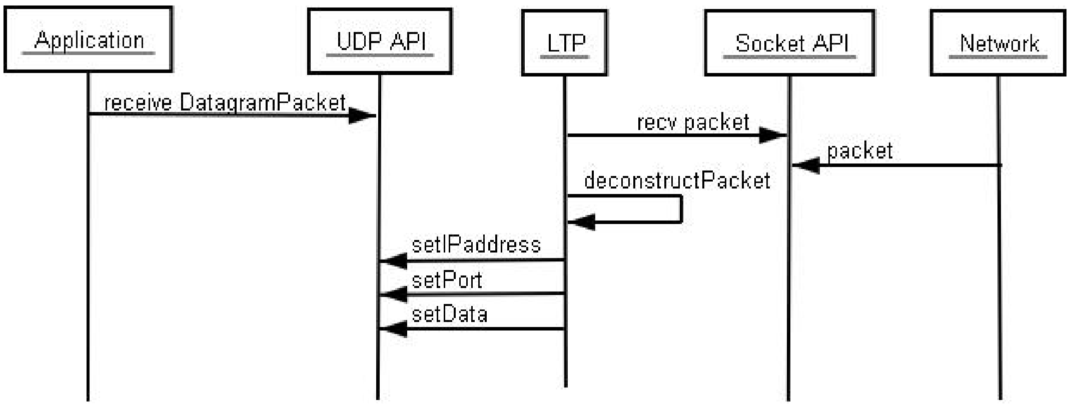


Figure 6.10: Data receiving with Fake Tunneling

When LTP constructs a LTP packet, the DatagramPacket, TCP and IP headers are used in the following way. The method getIPaddress of the DatagramPacket is used to set the DESTINATION ADDRESS field of the IP header. In addition, the method getPort of the DatagramPacket is used to set the DESTINATION PORT field of the TCP header. Finally, the method getData of the DatagramPacket is used to set the data portion of the LTP packet. The deconstructing of a LTP packet is done contrary to the constructing of a LTP packet. Also, when LTP deconstructs a LTP packet, the DatagramPacket, TCP and IP headers are used in the following way. The method setIPaddress of the DatagramPacket is used to get the SOURCE ADDRESS field of the IP header. In addition, the method setPort of the DatagramPacket is used to get the value of the SOURCE PORT field of the TCP header. Finally, the method setData of the DatagramPacket is used to get the data portion of the LTP packet. As a result, LTP has provided the mapping between UDP API and LTP packet. It should be noted that UDP packets are not transported to the network but the data that otherwise were used in conjunction with them are passed in the LTP packets.

In the design of Fake Tunneling it is suggested to omit some fields of the headers and indeed the checksum calculation is done according to that. Because in the implementation of Fake Tunneling that the author will done the use of the headers options are not exploited, the OPTIONS and as a result PADDING fields are not used. In addition to the fields that can be filled and retrieved with the UDP API, the rest of the fields must also handled. According to LTP Basics some of the fields can always hold constant values but the use of LTP Advances can affects which of the fields can further remain constant. LTP Basics defines that the VERSION, HLEN, TYPE OF SERVICE FLAGS, FRAGMENT OFFSET, TIME TO LIVE, PROTOCOL and SOURCE ADDRESS fields of the IP header can hold a constant value during data transportation as shown in Figure 5.11. In addition, the SOURCE PORT, HLEN, RESERVED, CODE BITS, WINDOW and URGENT POINTER fields can hold a constant value during data transportation as shown in Figure 5.12. In the implementation of Fake Tunneling that the author will do, the field options of LTP Advanced are not exploited so that the implementation is definitely compliant with the current TCP protocol implementation. But instead the connect method of the UDP API is used so that the DESTINATION ADDRESS and DESTINATION PORT need to be set only once and not each time a packet is sent. Now as the constant and volatile fields of the Fake Tunneling implementation of the author have been declared the checksum calculation speed up is done as suggested in the design of Fake Tunneling.

As suggested in the design of LTP, connection-opening mechanism is used. Although, the use of the connection reset is acceptable, connection closing is used instead because it is normal behavior of TCP connection. The choice of the connection closing is also meaningful because the current implementation of these mechanism are going to be exploited in addition to implementation of the author. Finally, LTP has to take care of that the connection is not reopened before it is closed. As a result, LTP must keep track on the state of the connections.

6.3 Implementation Aspects

LTP communicates with the application through the UDP API. In spite of the fact that in this Master's thesis the implementation of Fake Tunneling is done with Java programming language, any other programming language could also be used. The Java programming language is chosen because it enables independence of the application from the OS [34, 36]. But, the use of OS specific data means that this cannot directly be achieved although the socket abstraction is so widely accepted that the OS specific modification should be minimal. The Java programming language also offers a uniform Java API among Java programs so the same functionality is available for different Java programs. Finally, the Java programming language is chosen to give a uniform example of the implementation although the interface is separated from the implementation of the interface.

To summarize the use of the UDP API interface between the application and LTP, the application has to do the following phases to take advantage of the use of the Java UDP API with Fake Tunneling.

1. importing java.net and fi.vtt.tte.fi packages
2. substitution of UDP with LTP
3. use of Java UDP API

In addition to the UDP API, LTP communicates with the network through the Socket API. In spite of the fact that in this Master's thesis the implementation of Fake Tunneling GNU Socket API is chosen, the different Socket APIs are similar enough in general. As a result, the porting of this implementation also to the other OS is always possible. But, the chosen OS is GNU/Linux based on its great support for programming and raw sockets in the implementation of Fake Tunneling. Because the GNU Socket API is implemented with the C programming language also the part of LTP that interacts with the Socket API is implemented with the C programming language.

If the Socket API that the OS provides cannot directly be called from a Java program, all of the functionality needed cannot be implemented with the Java programming language. In such case the Java Native Interface (JNI) can be used to allow use of Java with another programming language [34]. Java denotes another programming languages as native. So, JNI interacts between the Java and native programming languages and as a result native side can be called from the Java side and vice versa. In Figure 6.11 connecting both the Java side and the native side with the JNI is shown so that the use of the Socket API is possible. If the OS provides the direct use of the Socket API from the Java side, JNI can be omitted and the functionality of the native side can be integrated directly to the Java side.

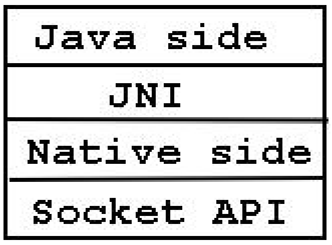


Figure 6.11: Connecting Java side to platform specific native side

On the Java side, Sun Microsystems Java 2 Software Development Kit (J2SDK) 1.4.1 on Linux, for now on J2SDK, is used because it is one of the latest Java distribution at this moment and enables both kinds of default transport layer substitutions. As the native language, the C programming language is used. JNI also supports connecting of Java and C programming languages. So, one part of LTP is implemented with Java and the other part with C and JNI is used. As a result of connecting the Java side and the native side with JNI, two modules are produced. The Java side is implemented as vtt.fi.tte.net package and constructed to the lumenet archive. The native side is implemented as lumenet library that is loaded at run time into the JVM. Because the native side is finally needed, the delay that the use of JNI causes cannot be avoided. But, the delay should be same for all the different protocols so it is considered as constant.

6.3.1 Branches

Depending on the OS that is used and the implementation specific details, the implementation of LTP can be divided into some branches. In Figure 6.12 a state diagram of the branches is shown. For clarity, the transition arrows and labels are shown only to the send method of the DatagramSocket of the UDP API. The receive method of Java UDP API is gotten by inverting the transition arrows but not all of the transition labels are then correct.

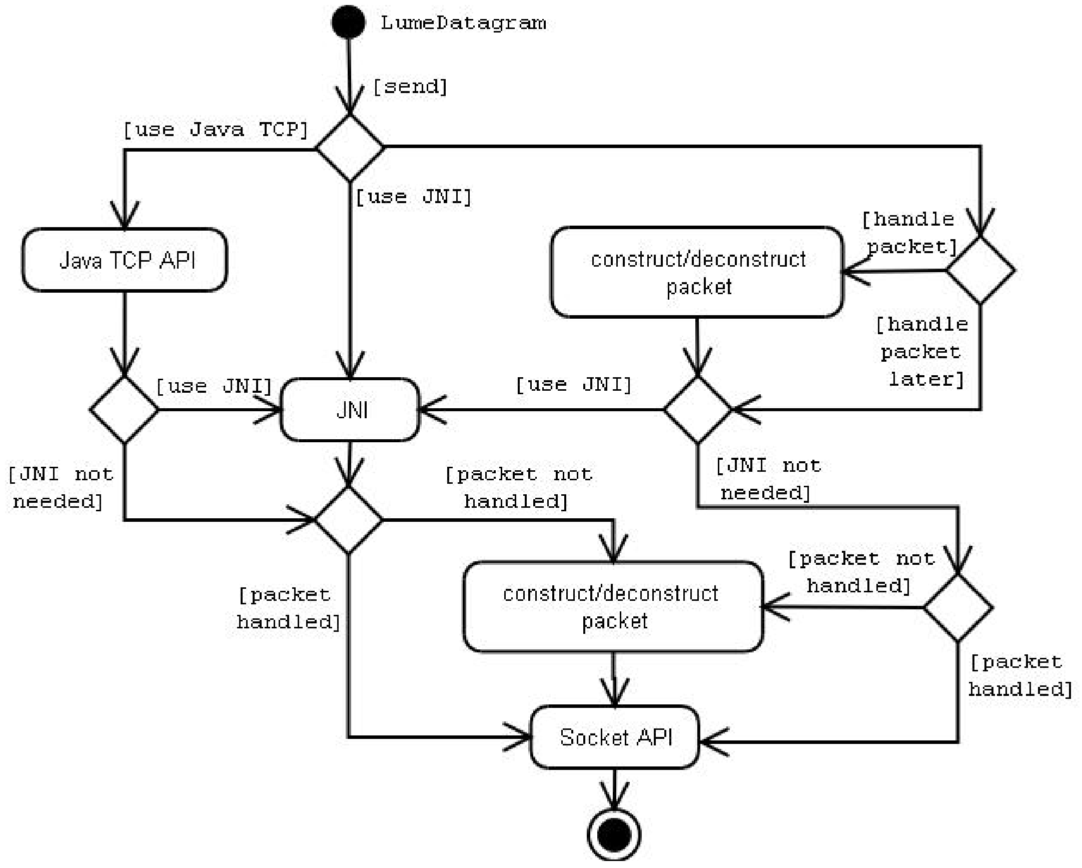


Figure 6.12: Possible branches of LTP implementation

<u>Summarizing the branches shown in Figure 6.12:</u>
1. LumeDatagram-Java TCP API-Socket API
2. LumeDatagram-Java TCP API-JNI-Socket API
3. LumeDatagram-JNI-construct packet-Socket API
4. LumeDatagram-construct packet-JNI-Socket API
5. LumeDatagram-construct packet-Socket API

Because the OS is usually implemented with a native language like C or C++, also the Socket API that the OS provides is used with a native language. There are also OS that are supposed to allow the direct use of Java [37] but the most popular OS of this moment do not support that. The branches 1 and 5 propose that JNI is not needed and based on that they are omitted. The branch 2 uses the Java implementation of TCP and implies that the full benefits of LTP are not exploited and based on that the branch is omitted. Finally, the branches 3 and 4 remains and are taken into more detailed consideration.

The C programming language is compiled to the platform dependent machine code that can be directly executed with the hardware. The Java programming language is compiled to the platform independent Java byte code that is at least at first interpreted with the Java Virtual Machine (JVM) that uses the hardware. As a result, the execution performance of C code is usually higher compared to Java because it is not directly executed with the hardware. But, the current JVM also compiles Java code during execution. Although, the compiling takes some time, the benefits can be greater because instead of interpreting, the code can be executed directly with the hardware. Although, the need for the high execution performance is dependent on the application and compared to C Java can take advantage of different kinds of runtime environments. Finally, there are translators that enable compiling of the Java programming language to the machine code instead of the Java byte code [38]. The current Java implementations of UDP and TCP are done according to branch 3. In addition, this can give better execution performance because the packets are constructed and deconstructed on the native side. Branch 4 let the programmer do as match of the implementation on the Java side because only the OS specific functionality must be implemented on the native side. As a result, it is possible to implement the constructing and deconstructing of packets on the Java side. This can also enable better utilization of the Java threads and class library. Although the branch 4 can provide great benefits to the development, the branch 3 is chosen instead. The purpose of this is to enable comparable tests to be performed. As a result, the branch 3 is implemented to make LTP according to the current Java implementations of UDP and TCP. The purpose of this is to enable comparable tests to be performed. Although, at this moment LTP is not implemented as a part of the GNU/Linux OS, the Scout OS and especially Scout In Linux Kernel (SILK) could improve the performance of Fake Tunneling because context switches would be avoided [5, 39].

6.4 Summary

In this chapter the implementation details of Fake Tunneling were considered. The implementation of Fake Tunneling uses Java and Socket APIs that are considered as de facto standards. These specific interfaces were also considered in detail. In addition, transport protocol substituting provided in the Java API was exploited to use the Lume Tunneling Protocol (LTP) of Fake Tunneling. It was also shown how LTP handles the mapping between UDP API and LTP packets. Although different implementation branches can be derived, not all of them exploit the full benefits of Fake Tunneling or cannot just be constructed with the most popular OS of this moment. As a result, the most suitable one of the different implementation branches was implemented.

7 Evaluation of Fake Tunneling

In Chapter 5 the design and in Chapter 6 the implementation of Fake Tunneling were considered. In this chapter the evaluation of the Fake Tunneling implementation that the author has done is going to be compared against the current technology described in Chapter 4. Also, evaluation of the criteria stated in Chapter 2 is considered. According to this, the results of this thesis and achievements that the author has made are presented and justified so that the results of the research and the quality of this thesis can be evaluated.

The objective of this chapter is to describe the evaluation of the Fake Tunneling implementation and also evaluation of the criteria. First the test bed of Fake Tunneling and test data are declared. After that the test cases are described and the test results of comparing Fake Tunneling against the current protocol implementations are declared. After that, the results of the tests are analyzed and synthesis is made. Finally, evaluation of the criteria is considered.

7.1 Fake Tunneling Test Bed

Test purpose of the test environment is to simulate the use of Fake Tunneling. The concept is that UDP cannot be used because of the firewall and TCP cannot be used because of its unsuitable characteristics to the real-time interaction as shown in Figure 5.1. In Figure 7.1 is shown the actual topology of the network that is used as the test bed for Fake Tunneling. Both the sender and the firewall are attached to the LAN of VTT and the receiver is directly attached to the firewall. The sender tries to send packets to the receiver. In order to be acceptable, the packets should pass though the firewall while sustaining suitability to the data transportation of real-time interaction. Because, comparing of LTP to both UDP and TCP is seen more attractive than comparing it just to TCP the firewall behavior will be disabled after the correctness of the concept is checked show that the result shown in Figure 5.3 could be compared against the current technology. So as the consequence, the firewall acts only as a router and UDP, TCP and LTP are compared to each other. In Figure 7.2, a more detailed description of the test bed components is given. The sender, firewall and receiver are Personal Computers (PC) and network is constructed with the 10/100 Ethernet technology.

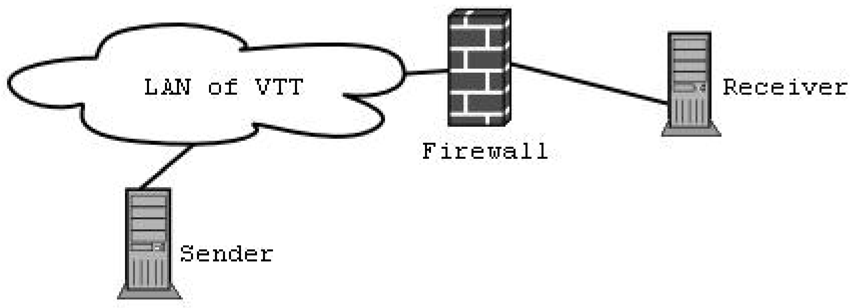
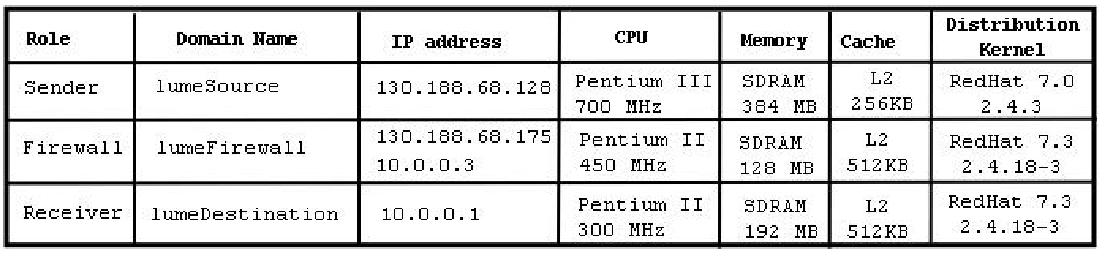


Figure 7.1: Test bed of Fake Tunneling



Role	Domain Name	IP address	CPU	Memory	Cache	Distribution Kernel
Sender	lumeSource	130.188.68.128	Pentium III 700 MHz	SDRAM 384 MB	L2 256KB	RedHat 7.0 2.4.3
Firewall	lumeFirewall	130.188.68.175 10.0.0.3	Pentium II 450 MHz	SDRAM 128 MB	L2 512KB	RedHat 7.3 2.4.18-3
Receiver	lumeDestination	10.0.0.1	Pentium II 300 MHz	SDRAM 192 MB	L2 512KB	RedHat 7.3 2.4.18-3

Figure 7.2: Detailed description of the test bed

Because the Ethernet is used as the underlying physical network, the translation mapping between IP address and physical address must be done. Because the OS already offers these service and they are utilized. Nevertheless, these can always be implemented if needed. The 10/100 Ethernet technology that is used sets limit to the MTU (Maximum Transfer Unit), namely 1500 B.

7.2 Measurements and Test Results

The test Fake Tunneling is going to be compared against the implementations of the current technology. As a result, LTP implementation that the author has done is going to be compared against the current protocol implementations of TCP and UDP included in J2SDK. In addition, similar test classes for all the protocols are implemented. The Java Runtime Environment (JRE) included into the J2SDK is used to execute all the tests cases.

7.2.1 Protocol Latency

The protocol latency consists of the time it takes for a packet between entering and exiting the protocol stack and is measured at the following way. At the sender side are measured the entering time from the application to the protocol stack and the exiting time from protocol to the network. At the receiver side are measured the entering time from the network to the protocol stack and the exiting time from the protocol stack to the application.

Measuring the time between the protocol stack and the network offers two obvious choices. Namely, measuring time in the implementation or in the network. Because the author implements Fake Tunneling, also the source code of the implementation will be available. But although Java implementation of the Java reference implementation is available, the availability of the native implementation of it is more restricted. Also, the usefulness of it is questionable because the change of the J2SDK could involve new investigation of the Java implementation. As a result, the network is monitored to measure the time when a packet exits or enters the protocol stack. The benefit of this is that comparable results are ensured because the times are measured on the same points for all of the protocols so this is also independent of the particular implementation. In Figure 7.3 are shown the measurement points of the protocol latency for the time of the moment when a packet is processed. In the application the measurement of time is part of the test class and on the network the measurement of time is done with Ethereal [40] network protocol analyzer.

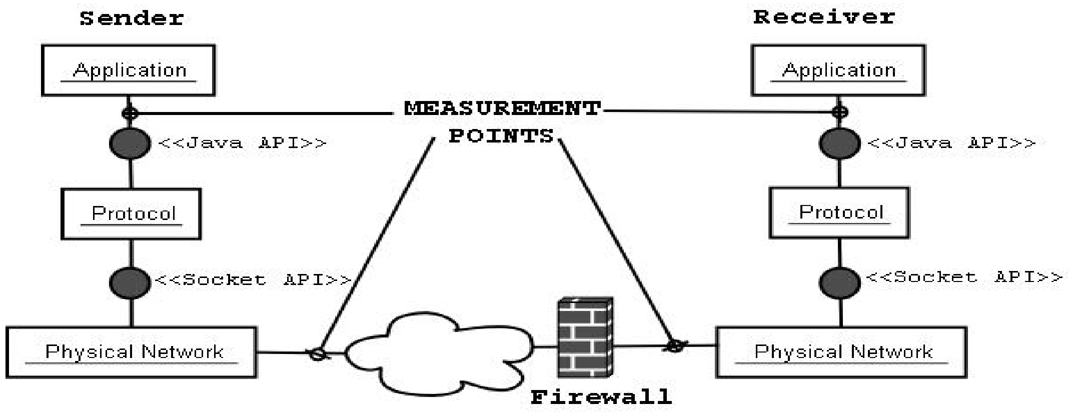


Figure 7.3: Measurement points of test cases

7.2.2 Test data

All of the test data is generated with the class java.util.Random of the J2SDK. The purpose of this is to generate different amounts of random data automatically. In addition, each protocol uses exactly the same data in corresponding test cases and the order of protocols is randomly chosen. In Figure 7.4 the payload and total packet sizes of test data for different protocols are shown although these turned to be proposal the justification of them is considered. The size of the payload data increases as a power of two. The maximum value that the TOTAL LENGTH field of the IP header can hold and the size of the IP and TCP headers set the maximum size for the payload data. This is the limit that can be used with LTP and TCP although UDP could in theory transport a little bit more because it does not contain so much header data as the other protocols.

Packet id	Payload (B)	UDP packet (B)	LTP packet(B)	TCP packet(B)
1	0	28	40	40
2	1	29	41	41
3	2	30	42	42
4	4	32	44	44
5	8	36	48	48
6	16	44	56	56
7	32	60	72	72
8	64	92	104	104
9	128	156	168	168
10	256	284	296	296
11	512	540	552	552
12	1024	1052	1064	1064
13	2048	2076	2088	2088
14	4096	4124	4136	4136
15	8192	8220	8232	8232
16	16384	16412	16424	16424
17	32768	32796	32808	32808
18	65495	65523	65535	65535

Figure 7.4: Proposed Test Packet sizes

Although payload sizes of packets as shown in Figure 7.4 were going to be used, it turned out that this would not be possible. The reason for that was that the UDP implementation of J2SDK could not handle packets which size was more than MTU of the network of the test bed, namely 1500 B. This limitation of UDP compared to the other protocols is shown in Figure 7.5. Instead of dividing the data into smaller packets the current UDP sender of J2SDK just sent them to the network. Obviously the network could not process larger packets than MTU so the data transportation failed. Instead, LTP and TCP divided the packets into smaller ones so that they could be transported over the network. To make the protocol implementations comparable to each other the limit of MTU is used to set the maximum size of the packet instead of the IP and TCP. As a result, the achieved payload and total packet sizes of test data for different protocols are shown in Figure 7.6.

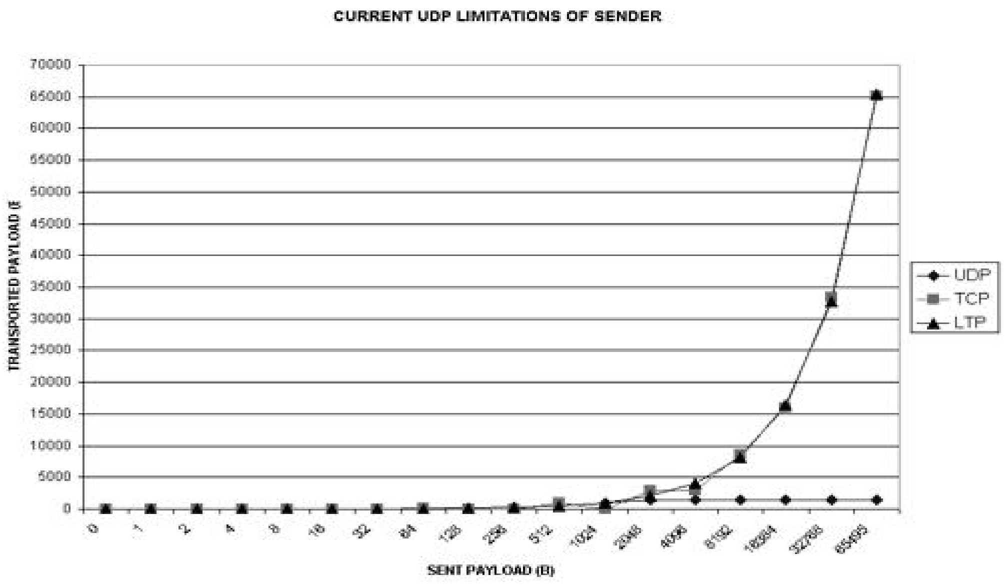


Figure 7.5: Limitations of current UDP sender

Packet id	Payload (B)	UDP packet (B)	LTP packet(B)	TCP packet(B)
1	0	28	40	40
2	1	29	41	41
3	2	30	42	42
4	4	32	44	44
5	8	36	48	48
6	16	44	56	56
7	32	60	72	72
8	64	92	104	104
9	128	156	168	168
10	256	284	296	296
11	512	540	552	552
12	1024	1052	1064	1064
13	1460	1488	1500	1500

Figure 7.6: Achieved Test Packet sizes

7.2.3 Test Cases

The test cases to be performed compare the real-time data transportation characteristics of the different protocols. As a result, protocol latency and jitter of the protocols of both the sender and the receiver are measured when the different number and size of packets are sent from the sender to the receiver. The purpose of these tests is to show the differences between the protocols so that a further improvement of the concept of Fake Tunneling can be performed if needed. It should be noted that all the other protocols but TCP could transport data if the size of the payload was zero.

Test Case 1

The first test case measures protocol latency of the sender as the function of payload in bytes. According to the settings, TCP should not send data immediately but can buffer it. In addition, UDP and LTP are not connected. In Figure 7.7 and Figure 7.8 the results of the test are shown. The former figure represents results of all the protocols and the latter figure UDP and LTP only so that also the differences between UDP and LTP can be seen.

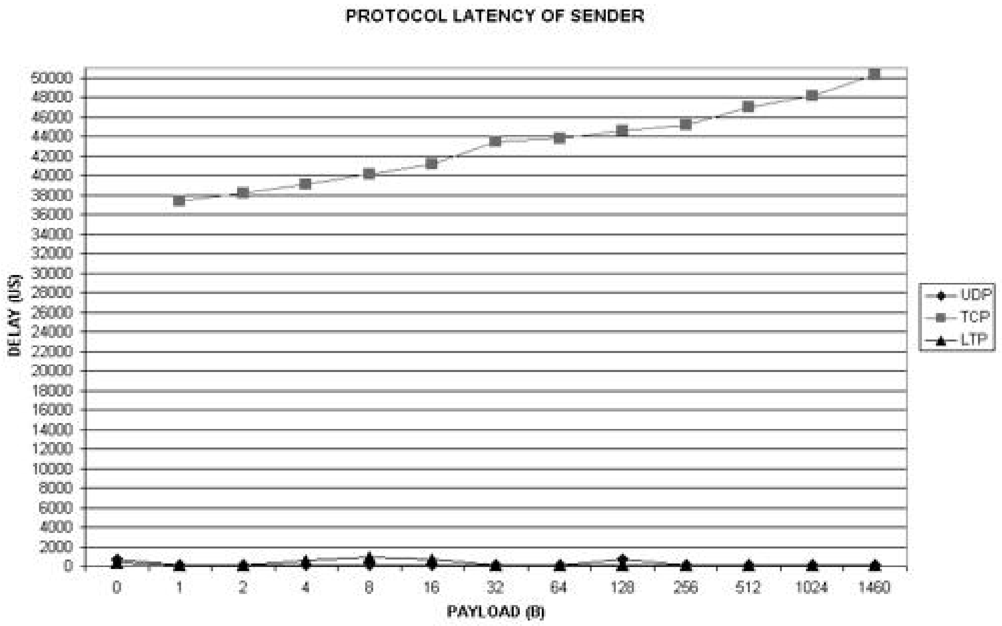


Figure 7.7: Protocol latency of sender, TCP delay, UDP and LTP not connected

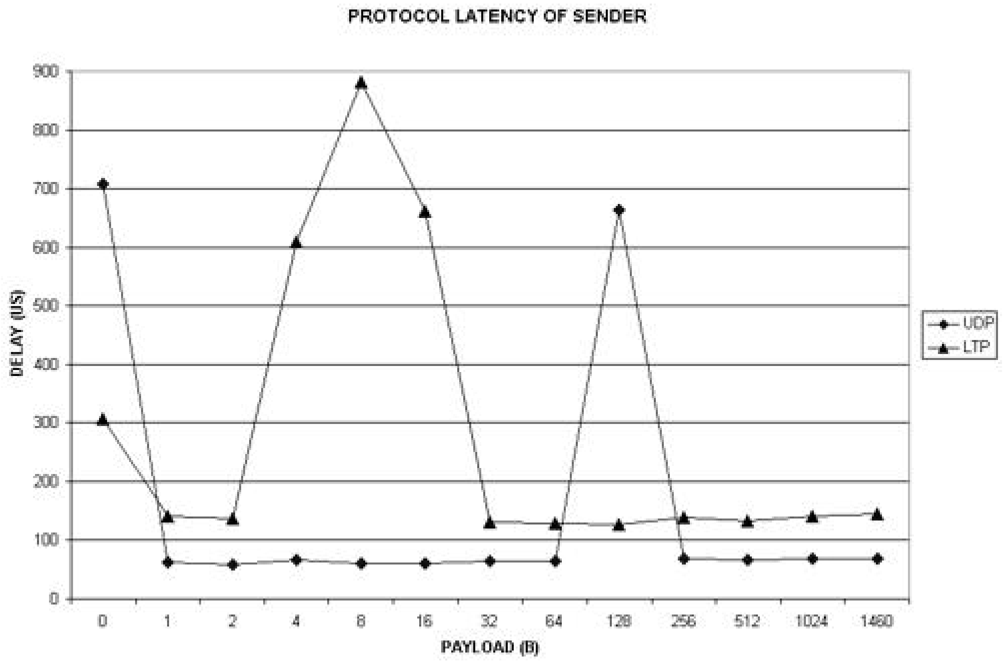


Figure 7.8: Protocol latency of sender, UDP and LTP not connected

Test Case 2

The second test case measures protocol latency of the sender as the function of payload in bytes. According to the settings, TCP should send data immediately and should not buffer it. In addition, UDP and LTP are connected. In Figure 7.9 and Figure 7.10 the results of the test are shown. The former figure represents results of all the protocols and the latter figure UDP and LTP only so that also the differences between UDP and LTP can be seen.

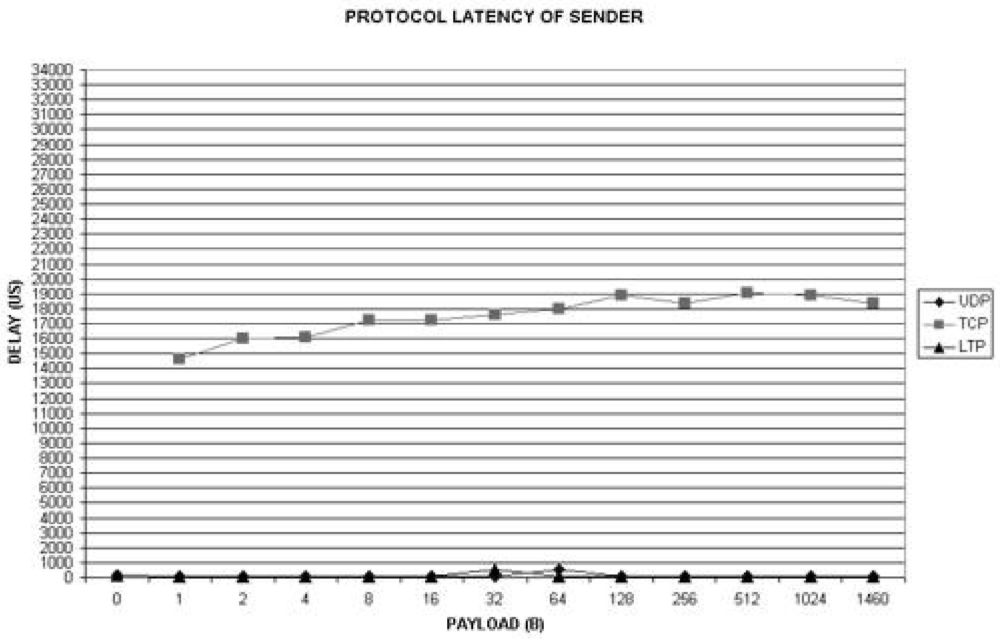


Figure 7.9: Protocol latency of sender, TCP no delay, UDP and LTP connected

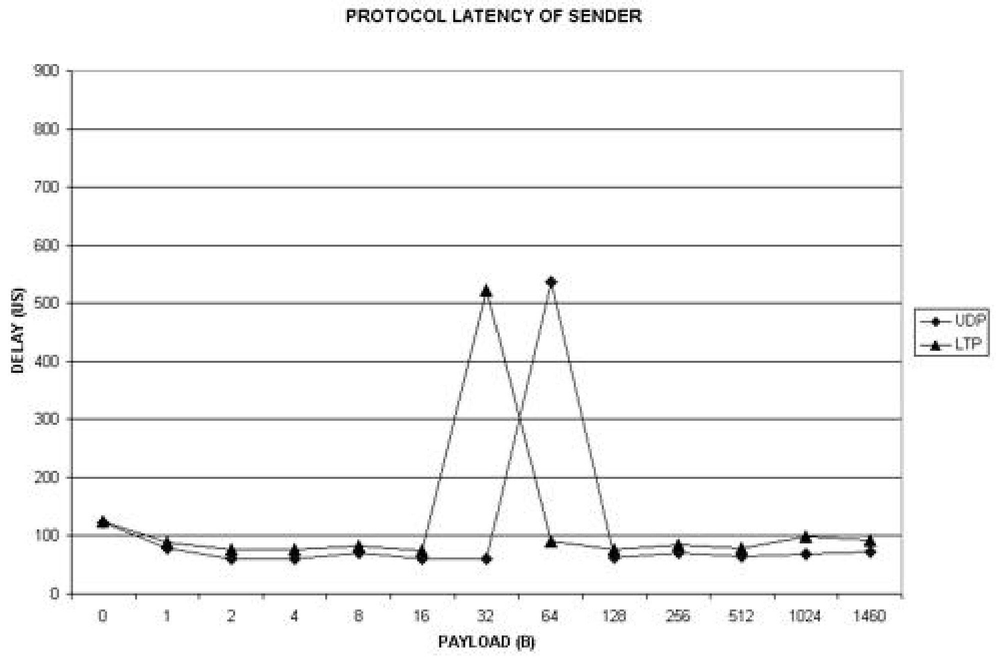


Figure 7.10: Protocol latency of sender, UDP and LTP connected

Test Case 3

The third test case measures protocol latency of the receiver as the function of payload in bytes. According to the settings, TCP should not send data immediately but can buffer it. In addition, UDP and LTP are not connected. In Figure 7.11 the results of the test are shown.

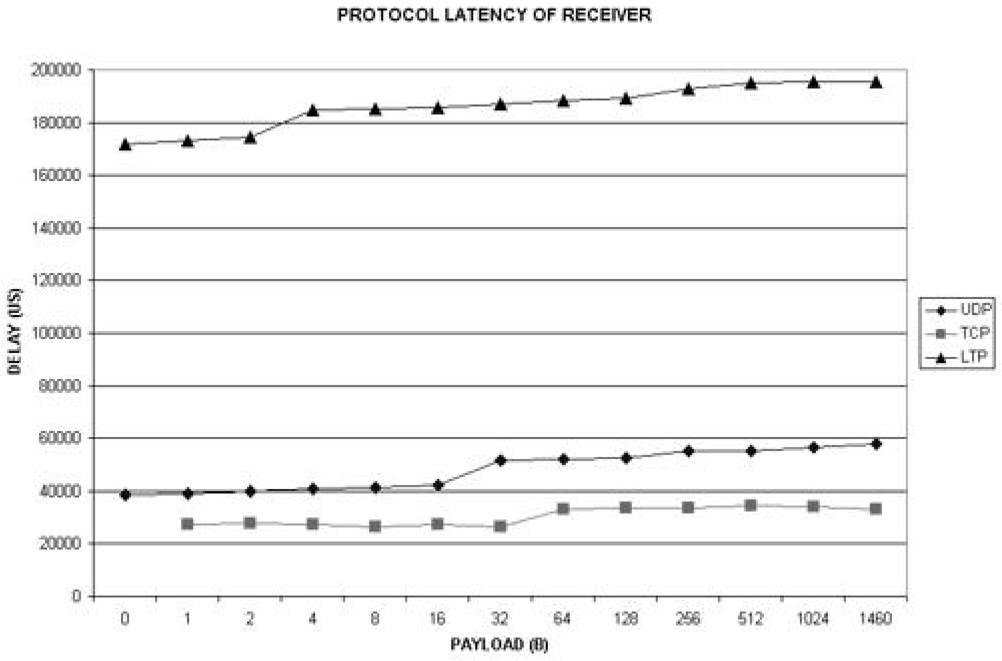


Figure 7.11: Protocol latency of receiver, TCP delay, UDP and LTP not connected

Test Case 4

The forth test case measures protocol latency of the receiver as the function of payload in bytes. According to the settings, TCP should send data immediately and should not buffer it. In addition, UDP and LTP are connected. In Figure 7.12 the results of the test are shown.

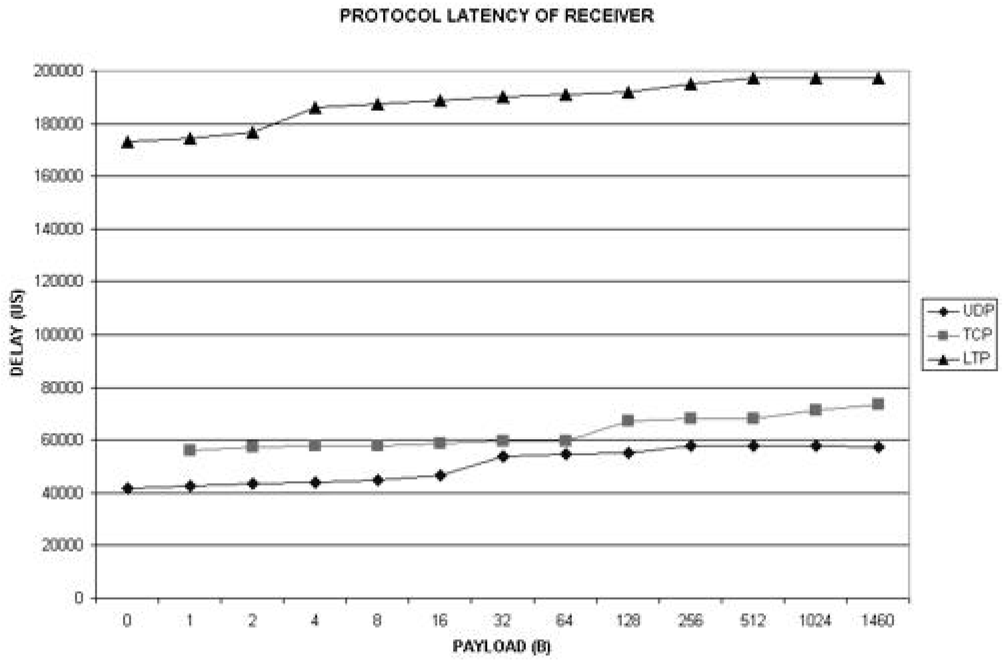


Figure 7.12: Protocol latency of receiver, TCP no delay, UDP and LTP connected

Test Case 5

The fifth test case measures protocol jitter of the sender as the function of payload in bytes. According to the settings, TCP should not send data immediately but can buffer it. In addition, UDP and LTP are not connected. In Figure 7.13 the results of the test are shown.

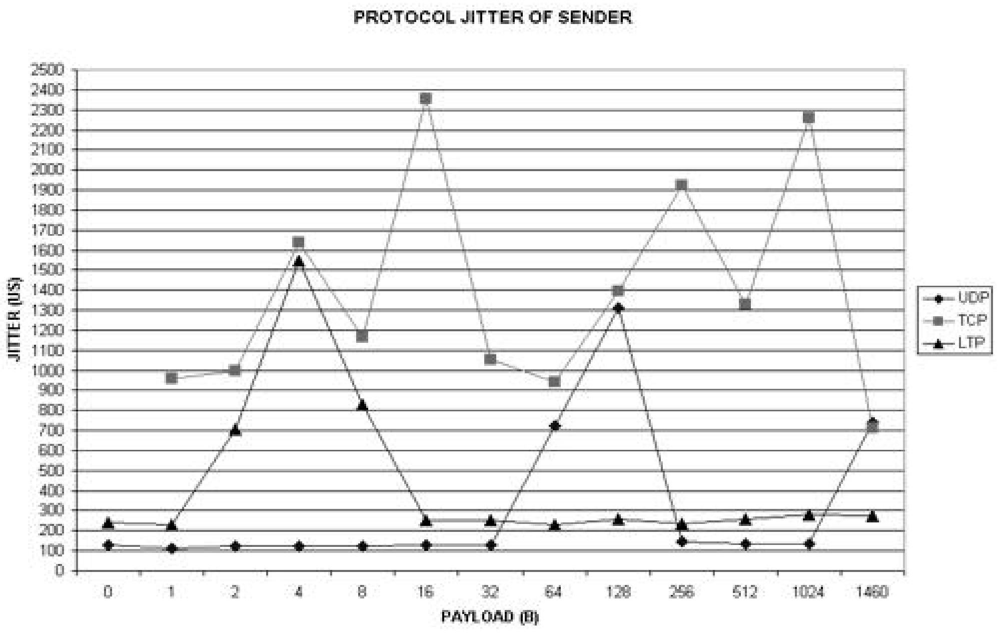


Figure 7.13: Protocol jitter of sender, TCP delay, UDP and LTP not connected

Test Case 6

The sixth test case measures protocol jitter of the sender as the function of payload in bytes. According to the settings, TCP should send data immediately and should not buffer it. In addition, UDP and LTP are connected. In Figure 7.14 the results of the test are shown.

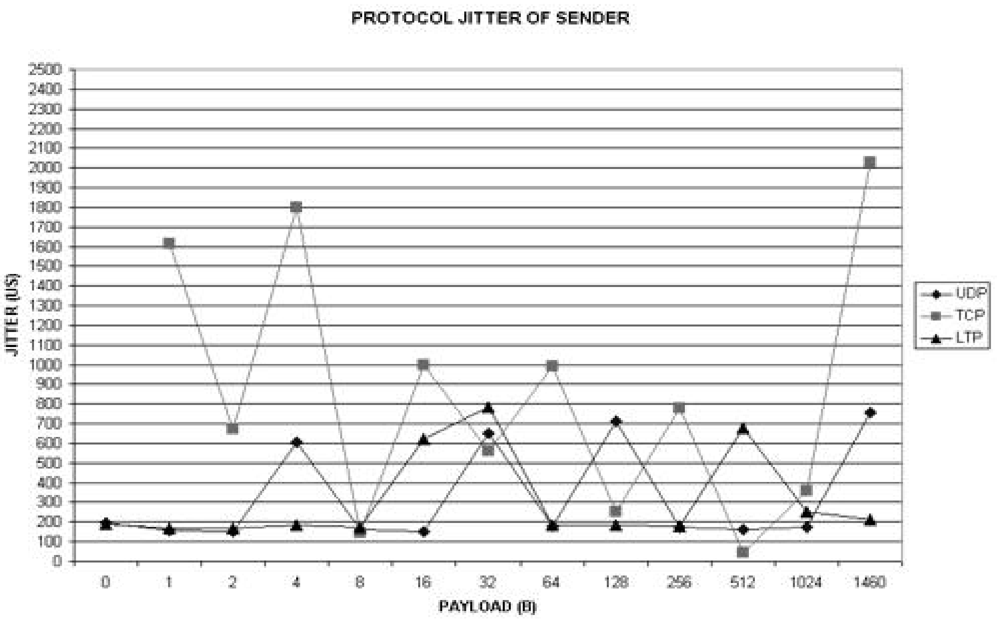


Figure 7.14: Protocol jitter of sender, TCP no delay, UDP and LTP connected

Test Case 7
The seventh test case measures protocol jitter of the receiver as the function of payload in bytes. According to the settings, TCP should send data immediately and should not buffer it. In addition, UDP and LTP are connected. In Figure 7.15 the results of the test are shown.

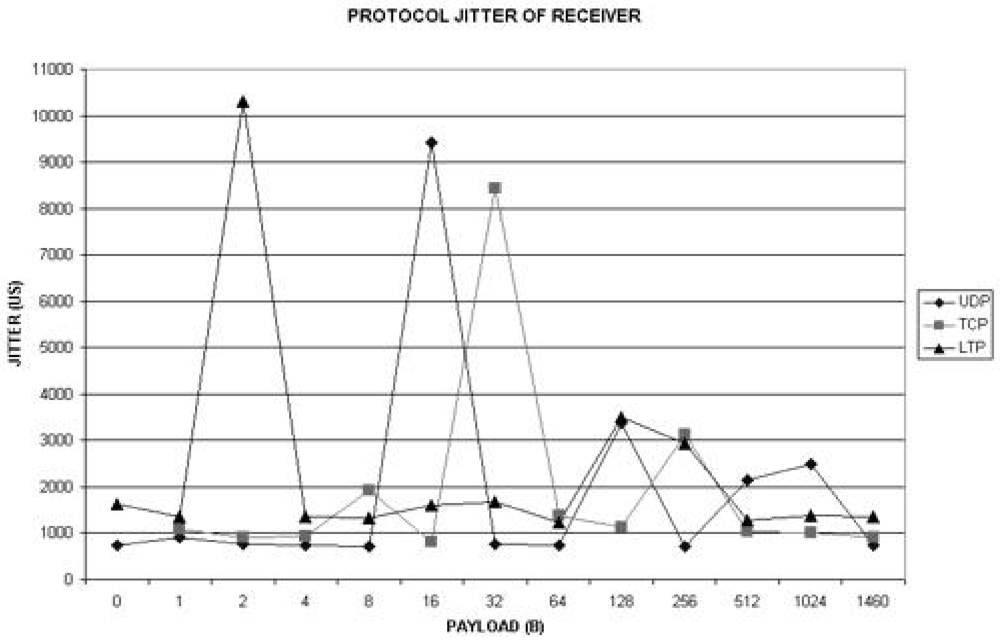


Figure 7.15: Protocol jitter of receiver, TCP delay, UDP and LTP not connected

Test Case 8
The eighth test case measures protocol jitter of the receiver as the function of payload in bytes. According to the settings, TCP should send data immediately and should not buffer it. In addition, UDP and LTP are connected. In Figure 7.16 the results of the test are shown.

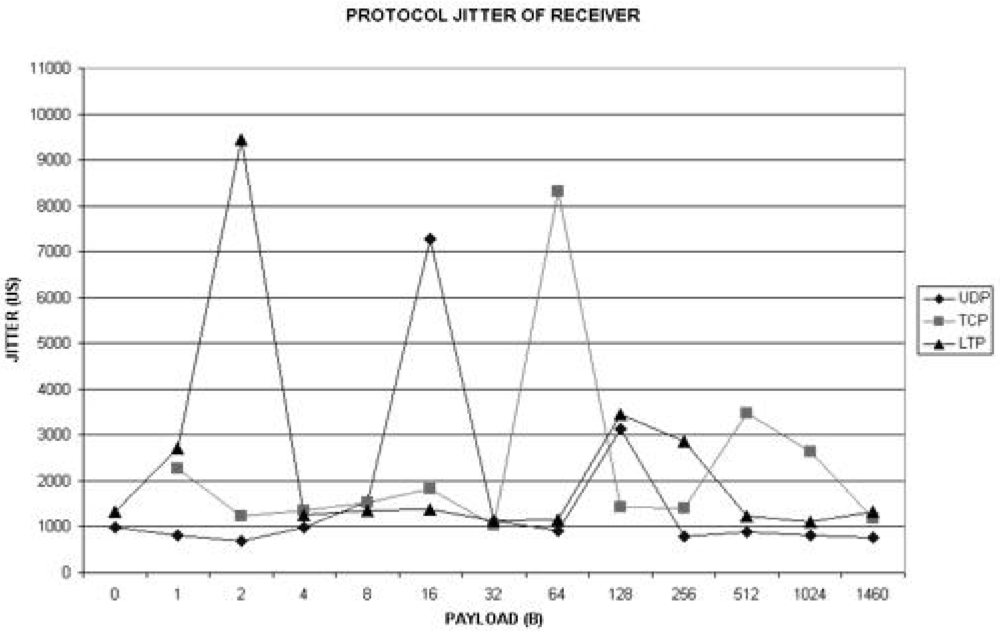


Figure 7.16: Protocol jitter of receiver, TCP no delay, UDP and LTP connected

7.3 Evaluation of the Test Results

To compare Fake Tunneling against the current technology, the author has performed tests to find out and show the differences between the protocols. As a result, LTP was compared against TCP and UDP and the results of each of the test cases are analyzed. It should be noted that although the time scale is in microseconds and milliseconds and although the lack of real-time network and operating systems (OS) has introduced interference, there are differences between the protocols. The interference can be seen as a peak on a stable curve

Test Case 1
This test measured delay of protocol stack on the sender side and in Figure 7.7 and Figure 7.8 the results of the test are shown. The unsuitable characteristics of TCP in real-time data transportation are shown as delay that is significant compared to UDP and LTP. The delay of TCP increases as the size of the payload increases. In addition, the delay of UDP and LTP compared to TCP is more stable and the size of the payload does not seem to affect although there are interference. As a result, delay of LTP on the sender side is similar to UDP and significantly better compared to TCP.

Test Case 2
This test measured delay of protocol stack on the sender side and in Figure 7.9 and Figure 7.10 the results of the test are shown. When TCP sent data immediately and both UDP and LTP were connected, the delay decreased for TCP and LTP but not for UDP. Although compared to the results of test case 1, the improvement of TCP is significant, the difference is still very high compared to the other protocols. Instead, delay of LTP has decreased to the level of UDP. As a result, delay of LTP can be decreased if it is connected.

Test Case 3
This test measured delay of protocol stack on the receiver side and in Figure 7.11 the results of the test are shown. The delay of TCP and UDP is smaller compared to LTP and the difference seems significant. The reason for that seems that because TCP uses a connection as an abstraction so the data receiving can be quicker compared to UDP and LTP that construct a packet that contains information about the sender. The difference of LTP compared to the other protocols relates to the implementation issues if the used mechanism that locks all the threads for a while is worse than use of a non thread locking mechanism. As a result, delay of LTP on the receiver side is not as good as delay of other protocols.

Test Case 4
This test measured delay of protocol stack on the receiver side and in Figure 7.12 the results of the test are shown. When TCP sent data immediately and both UDP and LTP were connected, the delay increased for TCP but not for UDP. The reason for this relates characteristics of TCP that provide reliable data transmission that increases data transportation when acknowledgements are sent to indicate that data is received. Although according to this and the previous test case the receiver side of LTP seems to be less capable compared to the other protocols, LTP can also be used in conjunction with the current implementations. As a result, Fake Tunneling can use LTP on the sender side and TCP on the receiver side and the delay can be decreased.

Test Case 5
This test measured jitter of protocol stack on the sender side and in Figure 7.13 the results of the test are shown. The jitter of UDP and LTP is smaller compared to TCP and the difference seems significant. The jitter of LTP is very similar to UDP and they both seem to be stable although there is interference. Instead the jitter of TCP seems to vary a lot and the reason relates to the buffering. As a result, jitter of LTP is similar to UDP and better than TCP.

Test Case 6
This test measured jitter of protocol stack on the sender side and in Figure 7.14 the results of the test are shown when TCP sent data immediately and both UDP and LTP were connected. The jitter decreased for all the protocols although TCP is still worse than the other protocols. As a result, jitter of LTP can be decreased if it is connected.

Test Case 7
This test measured jitter of protocol stack on the receiver side and in Figure 7.15 the results of the test are shown. Although there is interference, the jitter is very similar to all the protocols and denotes that the characteristics of the test bed were similar to all the protocols. In addition the behavior of the protocols stacks on the receiver side seem to be stable.

Test Case 8
This test measured jitter of protocol stack on the receiver side and in Figure 7.16 the results of the test are shown when TCP sent data immediately and both UDP and LTP were connected. As in the previous test case, there seem to be no difference.

Previously it was shown that the concept of Figure 5.1 was correct so that the result of the improvement shown in Figure 5.3 could be justified. In addition, it was previously noted that UDP could not handle restriction that MTU causes although according to IP larger packet could have been used. According to the result of the tests that were declared, there are significant differences between the protocols. Analyzes and synthesis of the tests have been shown that the characteristics of LTP on the sender side is more similar to UDP than TCP. Although the current delay of LTP on the receiver side is not as good as the other protocols, while TCP achieved smallest delay, the fact that LTP can also be used with the current TCP implementations also on the receiver side makes it superior compared to the current technology.

7.4 Evaluation of the Criteria

The purpose of the criteria stated in Chapter 2 are to evaluate the concept of Fake Tunneling that the author has developed as the solution to the research problem stated in Chapter 2 and also to evaluate the quality of this thesis. For clarity, each criterion is reviewed and shown in italic. After that, the evaluation of that particular criterion is made.

Criterion 1:

The solution should be feasible and it should be able to be implemented in a reasonable time.

The solution of the author declares a new way to exploit the current technology that is consistent with the environment so that real-time data transportation can be achieved. The solution is also designed and implemented in a reasonable time. As a conclusion, the Criterion 1 is fulfilled.

Criterion 2:

The solution should be capable to real-time data transportation so that use of multi-user VE is achieved between users at different places and that the users are capable to real-time interaction.

In the tests that the author performed, it was shown that the properties of LTP that the solution of the author uses are more close to the properties of UDP than TCP and suits well for real-time data transportation. As a result of the research the author has produced a solution and implementation that can be used to connect users of Virtual Environments (VE) over the network so that real-time interaction can be achieved. As a conclusion, the Criterion 2 is fulfilled extremely well.

Criterion 3:

The design of the solution should be done according to the current Internet protocols. Also, the use of the solution must be possible with the current protocol implementations.

It was clear that the solution could not even be successful if the characteristic of the current Internet protocols and Internet standards were omitted. As a result, the solution of the author exploits and improves the current technology. Also, the solution is designed and implemented according to the standard, de facto, interfaces. As a conclusion, the Criterion 3 is fulfilled extremely well.

Criterion 4:

The platform should be executable on different kinds of operating systems (OS) and it should be able to be attached to the existing VE.

Because the network consists of heterogeneous hardware and software is was clear that the use of the widely accepted Java technology and socket abstraction would be the best solution. The use of Java technology makes the implementation independent on the OS and the use of socket abstraction enables porting of the implementation to another OS although this requires changing some parts of the implementation. As a conclusion, the Criterion 4 is fulfilled.

7.5 Summary

In this chapter the test bed and test cases of Fake Tunneling were described, results of test cases were presented and evaluation of the test results was considered. It was stated that the properties of LTP are similar to UDP and can be used with current TCP implementations as denoted in the fundamental principles of Fake Tunneling. As a result, a significant improvement to the current technology can be achieved with the use of Fake Tunneling. In addition, evaluation of the criteria stated in Chapter 2 was considered. As a result, it was justified that the solution that the author has developed as a concept of Fake Tunneling fulfills well the requirements that were stated for this thesis.

8 Conclusions

To enable real-time data transportation on an IP-based network, such as the Internet, a suitable protocol should be used so that real-time interaction could be achieved between the users of a multi-user Virtual Environment (VE). Nevertheless, the current technology, namely TCP and UDP, is not adequate to this because the Internet does not guarantee Quality of Service (QoS) in general. In addition, firewalls can set limitations to the data transportation for security reasons. The former limitation can affect TCP and the latter limitation can affect UDP. As a result of this thesis, the author has developed the concept of Fake Tunneling. The justification for such a concept is the fact that the characteristics of the current transport protocols can be unsuitable for real-time data transportation. Also, the current technology has been improved so that the properties of the current transport protocols are exploited. The purpose of this was to benefit from the acceptance of one transport protocol, namely TCP, and from the characteristics of the other transport protocol, namely UDP, so that real-time data transportation could be achieved. As a result, the author developed Lume Tunneling Protocol (LTP) to be used as the transport protocol with Fake Tunneling. Because LTP uses the exact form of TCP, it can transport data over a network such as the Internet. In addition, because LTP uses data transportation similar to UDP, real-time data transportation can be achieved. According to the tests that were performed it turned out that LTP is superior to TCP and UDP and therefore Fake Tunneling is superior to the current technology.

The solution is based on the current technology so limitations are also related to it. Although, a change in the current technology that is used on the Internet is not likely. But, the use of the protocol header of TCP instead of UDP introduces protocol header overhead although the effect of it decreases as the payload size increases. In addition, TCP and UDP are in general implemented as a part of the operating system (OS). As a result, the performance of LTP could further be improved if it was also implemented as a part of the OS.

The further development of Fake Tunneling is continuing and it will be used to connect VE of VTT Technical Research Centre of Finland to the VE of the partners. In addition, the performance of Fake Tunneling on the receiver side could be improved. As a result, further tests to measure the characteristics of LTP compared to other protocols are going to be performed. Finally, publications of Fake Tunneling will be released.

REFERENCES

[1] Project T1P2P, date 2002-09-13,
http://www.vtt.fi/

[2] Project View of the Future, date 2002-09-13,
http://www.vtt.fi/

[3] Foster I., Kesselman C., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999, p.677,
ISBN 1-55860-475-8

[4] Metz C, IP QoS: Traveling in First Class on the Internet
Internet Computing, IEEE , Volume: 3 Issue: 2 , Mar/Apr 1999,
http://www.ieee.org/

[5] Tanenbaum A. S., Computer Networks, 3rd Edition, Prentice Hall PTR, USA, 1996, p.813,
ISBN 0-13-349945-6

[6] Huston G., RFC 2990 Next Steps for the IP QoS Architecture, 2000,
http://www.ietf.org/

[7] Anderson R. J., Security Engineering A Guide to Building Dependable Distributed Systems, John Wiley & Sons, USA, 2001, p.612,
ISBN 0-471-38922-6

[8] Napster, date 2002-10-25,
http://www.napster.com/

[9] Seti@Home, date 2002-10-25,
http://www.setiathome.berkeley.org/

[10] Buttazzo G.C., Hard real-time computing systems: predictable scheduling algorithms and applications, USA, 1997, p.379,
ISBN 0792399943

[11] International Organization for Standardization (ISO), ISO 9241-11:1998, date 2002-09-16,
http://www.iso.org/

[12] Nielsen J., Usability Engineering, Morgan Kaufmann Publishers, USA, 1994, p.362,
ISBN 0-12-518406-9

[13] Helander M., Landauer T.K, Prabhu P, Handbook of Human-Computer Interaction, 2nd Edition, Elsevier Science Publishers, Holland, 1997, p.1582,
ISBN 0-444-81876-6

[14] Comer D.E., Internetworking with TCP/IP Principles, Protocols and Architecture, 4th Edition, Prentice Hall, USA, 2000, p.750, ISBN 0-13-018380-6

[15] Tanenbaum A.S., Modern operating systems, Prentice Hall International, USA, 1992, p,728, ISBN 0-13-595752-4

[16] Stallings W., Operating systems Internals and Design Principles, 3rd Edition, Prentice Hall, USA, 1998, p.781, ISBN 7-302-02976-8

[17] Young H. D., Freedman R.A., University Physics, 9th Edition, Addison-Wesley Publishing Company, 1996, p.1484, ISBN 0-201-31132-1

[18] Postel J.B., RFC 821 Simple Mail Transfer Protocol, 1982, date 2002-10-25, http://www.ietf.org/

[19] The Internet Engineer Task Force, date 2003-01-17, http://www.ietf.org/

[20] Chwan-Hwa W., Irwin D., Emerging multimedia computer communication technologies, Prentice Hall, 1998, p.464, ISBN 0-13-079967-X

[21] Berners-Lee T., Cailliau R., Luotonen A., Nielsen H.F., Secret A., The World-Wide Web, Communications of the ACM, v.37 n.8, p.76-82, Aug. 1994 http://portal.acm.org/

[22] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T., RFC 2616 Hypertext Transfer Protocol - HTTP 1.1., 1999, date 2002-11-29, http://www.ietf.org/

[23] Barkai D., Peer-to-Peer Computing Technologies for Sharing and Collaborating on the Net, USA, Rich Bowles, 2002, ISBN 0-9702846-7-5

[24] Plummer D.C., RFC 826 An Ethernet Address Resolution Protocol, 1982, date 2002-11-29, http://www.ietf.org/

[25] Droms R., RFC 2131 Dynamic Host Configuration Protocol, 1997, date 2002-11-29, http://www.ietf.org/

[26] Mockapetris P., RFC 1034 Domain Names - Concepts and facilities date 2002-11-29, http://www.ietf.org/

[27] Cerf, V.; Kahn, R., A Protocol for Packet Network Interconnection, IEEE Transactions on Communications, Volume: 22 Issue: 5, May 1974, Page(s): 637 -648, date 2002-11-29, http://www.ieee.org/

[28] Leiner B., Cole R., Posten J., Mills D., The DARPA Internet Protocol Suite, IEEE Communications Magazine, Volume: 23 Issue: 3 , Mar 1985, Page(s): 29-34, date 2002-11-29, http://www.ieee.org/

[29] Postel J., RFC 791, Internet Protocol, 1981, date 2002-11-29, http://www.ietf.org/

[30] Postel J., RFC 793, Transmission Control Protocol, 1981, date 2002-11-29, http://www.ietf.org/

[31] Postel J., RFC 768, User Datagram Protocol, 1980, date 2002-11-29, http://www.ietf.org/

[32] Schulzrinne H., Casner S., Frederick R., Jacobson V., RFC 1889 A Transport Protocol for Real-Time Applications, 1996, date 2002-11-29, http://www.ietf.org/

[33] Fowler M., Scott K., UML Distilled Applying the standard object modeling language, Addison Wesley, USA, 1997, p.179, ISBN 0-201-32563-2

[34] Sun Microsystems, date 2003-01-10, http://java.sun.com/

[35] GNU Project, date 2003-01-10, http://www.gnu.org/

[36] Miyoshi A., Kitayama T., Tokuda H., Implementation and Evaluation of Real-Time Java Threads, Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE , 2-5 Dec 1997, Page(s): 166 -175 http://www.ieee.org/

[37] Project RTJOS, date 2003-04-24, http://www.vtt.fi/

[38] The GNU Compiler for the Java Programming language, date 2003-04-11, http://gcc.gnu.org/java/

[39] Scout operating system http://www.cs.princeton.edu/nsg/scout/, date 2003-04-11,

[40] Ethereal network protocol analyzer, date 2003-01-17, http://www.ethereal.com/