NETWORK SIMULATION FOR THE EFFECTIVE EXTRACTION OF IP NETWORK STATISTICS

Lodewyk Swanepoel
B.Eng (Electronic)

Thesis Submitted In Partial Fulfillment Of The Requirements For
The Degree
Magister Engineering (Electronic)
School Of Electric and Electronic Engineering
At The
Potchefstroom University For Christian Higher Education

Supervisor: Prof ASJ Helberg
Potchefstroom
2003

ABSTRACT

This study investigates the extent to which a communication sessions' QoS parameters can be measured through only extracting TCP/IP header data. The effect on these measurements based on the point of header extraction within the network as well as the OSI stack are also investigated.

An overview of packet switched networks and packet switched network protocols are given. The disadvantages and advantages of different network architectures and protocols are also given. Different network simulation tools are discussed and compared to find the most appropriate simulation tool for this study.

Two network topologies are introduced and sessions are constructed and monitored through only using the TCP/IP header data. Sessions are established and maintained and the results obtained from these sessions are compared and the most appropriate solution is chosen.

The results have shown that extracting data at the edge routers for sessions are the most optimal solution. These sessions are established and maintained through using virtual private network technologies and protocols.

UITTREKSEL

Hierdie studie ondersoek die mate waartoe 'n kommunikasie sessie se kwaliteit parameters gemonitor kan word deur slegs data vanuit die pakkie se beheer kopstuk te neem. Die effek op die metings afhangende van die posisie binne die netwerk waar die data ontrek word asook die posisie in die OSI stapel word ook ondersoek.

'n Oorsig van pakkie geskakelde netwerke asook protokolle wat in hierdie netwerke gebruik word, word gegee asook a oorsig van simulasie pakette wat gebruik kan word om netwerke te simuleer. Die voordele en nadele van verskillende netwerk argitekture en protokolle word ook gegee.

Twee netwerk topologiee word voorgestel en gebruik om kommunikasie sessies op te stel. Die kommunikasie sessies se kwaliteit parameters word gemonitor en met mekaar vergelyk om die optimale posisie te vind om die data vanaf te ontrek.

Die resultate toon aan dat die optimale posisie om data vanaf die netwerk te ontrek is op die netwerk kant "routers". Die kommunikasie sessies word opgestel en onderhou deur virtuele privaat netwerk tegnologieë en protokolle.

ACKNOWLEDGEMENTS

I would like to thank the following people for their contribution to this study.

- My project supervisor Prof. A.S.J. Helberg for his guidance, support and advice.
- My work colleagues for their support and advice.
- My friend J.C. Olivier for his support, advice and encouragement.
- My brother M.J. Swanepoel for his support, advice and encouragement
- My Family & Friends for their support and encouragement.

TABLE OF CONTENTS

ABSTRACT i

UITTREKSEL ii

ACKNOWLEDGEMENTS iii

TABLE OF CONTENTS iv

LIST OF FIGURES vii

LIST OF TABLES x

NOMENCLATURE AND ABBREVIATIONS xiii

Chapter 1. Introduction 1

1.1 Introduction 1

1.2 Problem statement 5

1.3 Method of investigation 5

1.4 Research methodology and thesis layout 6

Chapter 2. Background study 9

2.1 Circuit switching networks 9

2.2 Packet switching networks 12

2.3 Open system interconnection (OSI) model 18

2.4 TCP/IP protocol suite 23

2.5 QoS and SLA's in packet switched networks 32

2.6 Similar simulations and their results (Other QoS over IP simulations) 41

2.7 Conclusion 47

Chapter 3. Theoretical investigation of TCP/IP header QoS data extraction 48

3.1 Introduction 48

3.2 IP protocol suite 48

3.3 IP in detail 49

3.4 TCP in detail 56

3.5 TCP and IP header QoS extractable data 59

3.6 Conclusion 65

Chapter 4. Network simulation techniques and software 66

4.1 Introduction 66

4.2 System definition and modeling techniques 66
4.3 Simulation packages 70
4.4 CNET simulation model 81
4.5 Conclusion 93

Chapter 5. Problem methodology 94
5.1 Introduction 94
5.2 Problem methodology 94
5.3 Conclusion 106

Chapter 6. Results 107
6.1 Introduction 107
6.2 Monitoring a session without loss in a small network 107
6.3 Monitoring a session with loss in a small network 109
6.4 Monitoring multiple sessions through flooding on a no loss medium sized network 110
6.5 Medium network sessions with dedicated links 112
6.6 Conclusion 114

Chapter 7. Conclusion and Recommendations 115
7.1 Introduction 115
7.2 Summary 115
7.3 Conclusions 116
7.4 Proposed solution 117
7.5 Recommendations for future work 118

References 119

Appendix A (Network technology overview) 125
A.1 FDM and TDM 125
A.2 Long distances calls over a circuit switched network 128
A.3 Packet switched network routing techniques 130
A.4 IP addressing 134
A.5 IP subnet addressing 136
A.6 Internet routing 139

A.7 TCP Congestion Control 140

Appendix B (IP level protocol discussion) 143

B.1 ICMP (Internet control message protocol) 143

B.2 ICMP (Internet Control Message Protocol) 144

B.3 IGMP (Internet group management protocol) version 0 146

B.4 IGMP version1 and version 2 149

B.5 RGMP (Router group port management protocol) 155

B.6 GGP (Gateway to gateway protocol) 155

B.7 IP in IP encapsulation 156

B.8 Internet stream protocol (ST) 157

B.9 CBT (Core based trees) 157

B.10 EGP (Exterior gateway protocol) 158

B.11 UDP (User datagram protocol) 159

B.12 IRTP (Internet reliable transaction protocol) 161

B.13 SDRP (Source demand routing protocol) 162

B.14 The normal distribution 164

LIST OF FIGURES

Figure 1.3.1 Simulation method 6
Figure 1.4.1 Research methodology layout 7
Figure 2.1.1. Creating a link over a circuit switched network 9
Figure 2.1.2. Circuit switched connection 10
Figure 2.1.3 Elements of a circuit switch node 11
Figure 2.1.4 Space and time division switches 12
Figure 2.2.1 Transmission of packets over a packet switched network 13
Figure 2.2.2 Transmission of data across a packet switched network 14
Figure 2.2.3 Comparison between circuit switching, virtual circuit packet switching and datagram packet switching 17
Figure 2.3.1 The seven different OSI Layers 19
Figure 2.3.2 The OSI environment 22
Figure 2.4.1 Comparison between the TCP/IP and the OSI protocol architectures 24
Figure 2.4.3 TCP/IP protocol suite 26
Figure 2.4.4. IP packet fields 27
Figure 2.4.5. TCP packet construction 29
Figure 2.4.6. UDP header 32
Figure 2.5.1 FIFO queuing 37
Figure 2.5.2 Priority queuing 38
Figure 2.5.3 Custom queuing 38
Figure 2.5.4 Weighted fair queuing 39
Figure 2.6.1 Simple simulation topology 42
Figure 2.6.2 Complex simulation topology 43
Figure 2.6.3 Simulation topology 44
Figure 2.6.4 Passive and active QoS monitoring techniques 45
Figure 3.3.1 IP header 50
Figure 3.4.1 TCP header 56
Figure 3.4.2 Pseudo header format 58
Figure 3.5.1 Extracting QoS fields from an IP header 63
Figure 3.5.2 Extracting QoS parameters from a TCP header 64
Figure 3.5.3 QoS parameter extraction from both TCP and IP headers 65
Figure 4.1.1 Model overview 68

Figure 4.4.1 CNET simulation model 81
Figure 4.4.2 Node 0 transmits a message to Node 1 83
Figure 4.4.3 Example of a topology file 85
Figure 5.2.1 Methodology used 95
Figure 5.3.1 Network layer flow diagram 96
Figure 5.3.2 Data link layer flow diagram 97
Figure 5.3.3 Host software layout 98
Figure 5.3.4 Data format 98
Figure 5.3.5 Router flow diagram 99
Figure 5.3.6 Small network topology 100
Figure 5.3.7 Medium network topology 101
Figure 6.2.1 Small network topology 107
Figure 6.4.1 Medium network topology 110
Figure A.1.1. Transmitter and receiver sections of a FDM system 126
Figure A.2.1 Switched network topology 128
Figure A.3.1. Fixed routing example network 131
Figure A.3.2. Example of flooding a network 133
Figure A.4.1. IP address construction 135
Figure A.4.2. Construction of the different IP addresses 136
Figure B.1.1 ICMP header construction 143
Figure B.3.1 IGMP header 147
Figure B.4.1 IGMP version 1 header 152
Figure B.4.2 IGMP version 2 header 154
Figure B.5.1 RGMP header construction 155
Figure B.6.1 GGP packet construction 156
Figure B.8.1 ST header format 157
Figure B.9.1 CBT header format 158
Figure B.10.1 EGP header format 159
Figure B.11.1 UDP header format 160
Figure B.11.2 Pseudo header contents if the field is carried via IPv4 160
Figure B.11.3 Pseudo header contents if the field is carried via IPv6 161
Figure B.12.1 IRTP header format 161
Figure B.13.1 SDRP header format 162

Figure B.15.1 Normal distribution 164

LIST OF TABLES

Table 1.1.1 IP probe features and accompanying feature specification requirements .. 3
Table 1.1.2 System operational requirements 4
Table 1.4.1 Thesis layout 8
Table 2.2.1 Comparison between packet and circuit switching networks 16
Table 2.5.1 Disadvantages of IntServ and DiffServ 36
Table 2.6.1 MQM header format 45
Table 2.6.2 MQM Ping message format 46
Table 2.6.3 MQM Beacon message format 46
Table 3.2.1 Examples of different layer protocols 49
Table 3.3.2 IP header version field contents 50
Table 3.3.2 Precedence field values and corresponding priorities 51
Table 3.3.3 Delay, throughput, reliability and monetary field values 51
Table 3.3.4 Flag field bits description 52
Table 3.3.5 Protocol field values and their respective protocols 54
Table 3.3.6 Class field values and their respective meanings 55
Table 3.3.7 5-bit option field values and descriptions 55
Table 3.4.1 TCP ECN field contents 57
Table 3.4.2 TCP header flag field information 57
Table 3.4.3 TCP options field content 59
Table 4.1.1 A short description of dynamic, continuous models and dynamic, discrete models 69
Table 4.3.1 Table representing different aspects of the NETSIM simulation model .. 71
Table 4.3.2 Table representing different aspects of the NIST network simulation tool 72
Table 4.3.3 Table representing different aspects of the CPSim network simulation tool 73
Table 4.3.4 Table representing different aspects of the INSANE network simulation tool 74
Table 4.3.5 Table representing different aspects of the NEST network simulation tool 75

Table 4.3.6. Table representing different aspects of the REAL network simulation tool 76
Table 4.3.7 Table representing different aspects of the NS network simulation tool . 78
Table 4.3.8 Table representing different aspects of the OPNET network simulation tool 79
Table 4.3.9 CNET network simulation tool attributes 80
Table 4.4.1 Global attributes 85
Table 4.4.2 Node attributes 87
Table 4.4.3 Link attributes 89
Table 5.3.1 Data format explanation 99
Table 5.3.2 Test 1 explanation 102
Table 5.3.3 Test 2 explanation 103
Table 5.3.4 Test 3 explanation 104
Table 5.3.5 Test 4 explanation 105
Table 6.2.1. Link delay information 107
Table 6.2.2 Measured router details towards node C 108
Table 6.2.3 Measured session details towards node A 108
Table 6.2.4 Maximum and minimum end-to-end delays 108
Table 6.3.1 Measured router details towards node C 109
Table 6.3.2 Measured session details towards node A 109
Table 6.3.3 Maximum and minimum end-to-end delays 110
Table 6.4.1 Medium network topology link information 111
Table 6.4.2 Measured session details towards node E from Node A 111
Table 6.4.2 Measured session details towards node E from Node A 111
Table 6.5.1 Measured session details towards node E from Node A 112
Table 6.5.2 Measured session details towards node A from Node E 112
Table 6.5.3 Measured session details towards node C from Node B 113
Table 6.5.4 Measured session details towards node B from Node C 113
Table 6.5.5 Measured session details towards node D from Node F 113
Table 6.5.6 Measured session details towards node F from Node D 113
Table A.1.1 International FDM carrier standards 126
Table A.1.2. International TDM standards 127
Table A.2.1 Routing table for figure A.2.1 129

Table A.3.1. Routing table for bridge B1 131
Table A.3.2. Routing table for bridge B2 132
Table A.3.3. Routing table for bridge B3 132
Table A.3.4. Routing table for bridge B4 132
Table A.4.1. Reference information about the five different IP address classes..... 135
Table A.4.2. A range of possible values for the first octet of each address class..... 136
Table A.5.1. Class B sub-netting reference table 138
Table A.5.2. Class C sub-netting reference table 138
Table B.1.1 ICMP type field contents 144
Table B.3.1 IGMP type field contents 147
Table B.3.2 Code field for a reply message scenario 148
Table B.4.1 IGMPv2 type field contents 154
Table B.6.1 GGP type field content 156
Table B.9.1. CBT type field definitions 158
Table B.13.1 SDRP flag field content 162
Table B.13.2 SDRP notification field contents 163

NOMENCLATURE AND ABBREVIATIONS

AAL	ATM adaptation layer
ACAP	Application configuration access protocol
AH	Authentication header
ANSI	American National Standards Institute
APEX	Application exchange core
ARIS	Architecture of integrated information systems
ARP	Address resolution protocol
ATM	Asynchronous transfer mode
ATMP	Ascend tunnel management protocol
AURP	Apple talk update based routing protocol
BFTP	Background file transfer protocol
BGP	Border gateway protocol
CBQ	Class based queuing
CCITT	Consultive Committee for International Telegraphy and Telephony
CFTP	Command line FTP
COPS	Common open policy service
CoS	Classes of Service
CPU	Central processing unit
CRANE	Common reliable accounting for network element
CSN	Circuit switched network
DCE	Distributed computing environment
DCN	Data communications network
DDX	D-II Data exchange
DHCP	Dynamic host configuration protocol
DICT	Dictionary server protocol
DRARP	Dynamic RARP
DRR	Deficit round robin
EGP	Exterior gateway protocol
EMSD	Efficient mail submission and delivery
ESP	Encapsulating security payload
FCAPS	Fault, Configuration, Accounting, Performance, Security
FDM	Frequency division multiplexing
FIFO	First in first out
FTP	File transfer protocol

GGP	Gateway to gateway protocol
GRE	Generic routing encapsulating
GUI	Graphical user interface
HMP	Host monitoring protocol
HTTP	Hyper text transfer protocol
IATP	Interactive agent transfer protocol
ICMP	Internet control message protocol
IDRP	Inter domain routing protocol
IEEE	Institute for electrical and electronic engineers
IFMP	Ipsilon flow measurement protocol
IGAP	IGMP for user authentication protocol
IGMP	Internet group management protocol
IGP	Internet gateway protocol
IGRP	Internet gateway routing protocol
IHL	Internet header length
InARP	Inverse address resolution protocol
IP	Internet protocol
IPPCP	IP payload compression protocol
IRTP	Internet reliable transaction protocol
ISO	International organization for standardization
ISP	Internet service provider
ITU	International telecommunications union
L2TP	Layer 2 Transfer Protocol
LAN	Local area network
LIFO	Last in first out
LLC	Logic Link Control
MAC	Media Access Control
MFE	Multiple Format Evaluation
MHRP	Mobile Host Routing Protocol
MIME	Multipurpose Internet Mail Extensions
MTU	Maximum Transfer Unit
NFS	Network File System
NIST	National Institute Of Standards And Technology
NMS	Network Management System
NS	Network Simulator
OSI	Open System Interconnection

PIM	Primary Interface Module
PNNI	Private Network-To-Network Interface, Private Network Node Interface
PSN	Public Switched Network
PTP	Point-To-Point
PU for CHE	Potchefstroom University for Christian Higher Education
QoS	Quality of Service
RARP	Reverse Address Resolution Protocol
RED	Random Early Detection
RGMP	Router Group Management Protocol
RIP	Routing Information Protocol
RMON	Remote Network Monitoring
RSCP	Radio Resource Control Protocol
RSVP	Resource Reservation Protocol
SDRP	Source Demand Routing Packet
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SM	Service Management
SMDS	Switched Multi-Megabit Data Service
SMP	Simple Management Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	System Management Network Protocol
SNP	Sequence Number Packet
SQL	Structured Query Language
SRP	Signal Reservation Protocol
SS7	Signaling System 7
ST	Segment Type
STM	Synchronous Transfer Mode
TCP	Transfer Control Protocol
TDM	Time Division Multiplexing
TFTP	Trivial Trial Transfer Protocol
TMN	Telecommunication Management Network
TOM	Telecom Operations Map
TUBA	TCP/IP and UDP with bigger addresses
UDP	User Datagram Protocol
VPN	Virtual Private Network
VRRP	Virtual Router Redundancy Protocol

WAN	Wide Area Network
WFQ	Weighted Fair Queuing
XML	Extensible Markup Language
XTP	Express Transfer Protocol

Chapter 1. Introduction

Abstract - The aim of this chapter is to introduce the reader to the proposed research, problem scenario and possible solutions to the problem. The specific research methodology and current technologies and trends will also be discussed and established. Furthermore the beneficiaries of the research will be mentioned and the purpose of the project will be discussed.

1.1 Introduction

Installation of telecommunication networks is very expensive, therefore before such an expensive network can be installed it must be certain that the network will perform to its predefined and intended specifications. Even applications and data probes running on these networks must be reliable and fault free when they are installed. Faulty applications may cause downtime and implicated financial losses to the telecommunication company.

Modeling and emulating of these networks and network applications provides a reliable and more cost effective solution to telecommunication companies worldwide than installing networks with over-engineered bandwidth or debugging network and network applications in real time. The question may be asked why emulation is a better alternative than installing over-engineered networks and debugging network application software in real time?

When a real network is considered one must keep in mind the fact that such a network must be installed to perform tests on. Another factor that must be kept in mind is that a real network is hard to configure and its behavior is not easily reproducible and reliable. It is also difficult to develop and debug distributed applications in a single lab environment when a real network is considered. In contrast to an emulated network in which only a software model is needed, making it easy to vary and configure the emulated network configuration. Emulation also offers the advantage of easily reproducing network behavior at will as well as enabling applications to be co-located in a single local lab for developing and debugging [1,4].

Thus the days of over-engineering for bandwidth are numbered. Few companies can afford to throw extra megabits per second at a project when the budget calls for accurate, robust and economical network designs from the start [2]. Telkom SA Ltd. has recently (6/12/2002) issued a request for information (RFI) concerning the development of an IP probe which could be used to extract service level (SLA) and

quality of service (QoS) information for certain classes of service (CoS). The idea is to provide a value added IP connectivity service with QoS guarantees to its customers. Thus an advanced reporting and monitoring system is needed, which will be performed by the IP probes installed within the network. The implemented system must be able to measure the performance of the network in aggregate terms (Classes of Service) as well as on a per customer basis and must also be able to apply real time monitoring within the network to:

- Monitor threshold violations (e.g. delay bounds, throughput bounds, etc).
- Perform fault analysis.
- Allow lawful interception of traffic.
- Capture flow information for usage-based billing functions [3].

The implemented system must also have the ability to report data to overhead systems in various formats and functions for auditing purposes. Thus they require a solution that is capable of delivering the essential monitoring, measuring and reporting functionality to provide a quality value-add service to their customers. Information concerning the following features were requested, which could be subdivided into data capturing, data analysis, interception/traffic testing capabilities and security fields.

Field	Information required
Data capturing	• The type of data that can be captured as well as the device's ability to allow for flexible measurement and timing settings of measurements. • The ability to generate artificial traffic for measurement of QoS for various CoS. • The ability to measure throughput, delay, packet loss, jitter and other QoS metrics by the probe as well as the ability of the probe to track per customer and aggregate CoS traffic statistics. • A description of how per customer protocol analysis can be achieved, e.g. using RFC2547 Route descriptor or any other unique identification keys. • An indication of the ability of the probe to measure link utilization statistics in real-time.

	• A description of the ability of the probe to monitor flows for threshold violations, e.g. sending SNMP traps when thresholds are exceeded as well as information on how and what flow information may be captured for usage-based billing functionalities.
Data analysis	• A description of the probe's ability to calculate averages, percentile, and probability distribution function data from measured data obtained from the device. • The ability to correlate measured data from various probes into a single customer detailed accounting record (CDR) as well the ability to store these accounting records in a centralized database.
Interception/traffic testing	• A description of the ability of the router to filter data based on some unique identification key, e.g. using RFC2547 Route Descriptor as well as the ability of the device to copy data to an alternative interface (e.g. hard drive) for lawful interception. • The device's ability to generate diagnostics traffic as well as the ability of the probe to dynamically configure traffic monitoring profiles.
Security	• The ability of the device to perform intrusion detection at line rate as well as the device's ability to detect denial of service attacks. • The ability of the device to proactively act in the above-mentioned cases and then to notify a network management system by means of an SNMP trap or similar mechanism.

Table 1.1.1 IP probe features and accompanying feature specification requirements

The extracted data must be forwarded to Collection and Analysis databases, for which the following information was requested.

- Storage requirements of the data.
- Correlation features supported that allow multiple records for a single flow from one or many probes to be correlated into a single accounting record.
- Ability to generate detailed contextualized reports on a per customer and per CoS basis.

- Advice on the topology, e.g. should the database be centralized or distributed for failure protection [3].

The operational requirements of the probe that was requested can be classified as follows,

Operational requirement	Requirements
System requirements	• The system must be able to perform all of the previously mentioned functions concurrently. • The system must be able to perform analysis on the above data at line rate. • The system must be able to monitor multiple interfaces on a single device. • The system must be able to support RMON1 (RFC1757) and RMON2 (RFC2021). • The interfaces required are: STM1 ATM (current), STM4 ATM (future) and STM1/4 POS (future). • Advice is also required on the topology of the measurement system, i.e. should the device be in-line, on a separate router interface or both.
Network management	• A system of these probes must be manageable from a centralized point. • FCAPS capabilities are required to support the solution. • Advice is also required on how the system should interface with e-health. • Standard reporting interfaces are required e.g. SNMP, XML and SQL. • The northbound interfaces available to integrate this management system with others. • The data storage capability of the Network Management System (NMS) must also be defined. • Advice is also required on the topology of the NMS.

Table 1.1.2 System operational requirements

1.2 Problem statement

The installation and maintenance costs of telecommunication networks are expensive [2]. Due to competition between the telecommunication companies these costs must be minimized. This implies that the telecommunication companies must install reliable networks with exact bandwidth requirements to achieve these minimized costs. They can't install over-engineered networks and they can't afford downtime from their networks. They must guarantee certain performances from their network with certain bandwidths and certain throughputs, which is stated and agreed upon between the customer and the telecommunications company within the SLA.

To achieve these precise network performance criteria, these networks are first modeled. Telkom released an RFI (Request for Information) for information concerning the installation of IP probes within their current network. This RFI requests information of existing hardware implementations, making the need for simulating the installation and operation of these probes a fundamental part of the total solution. The purpose of this study is to investigate the extraction of adequate QoS data at various layers and points in a TCP/IP network.

1.3 Method of investigation

A simulated network will be constructed in a network simulator package due to cost constraints, as well as the lack of a large offline test bed. This approach has the added benefits that no service disruption or loss of data occurs. Further advantages include that a stable network configuration can be maintained which is seldom possible within a live network. Different simulation packages will be compared to determine which package will be used for the simulation. Different network topologies and network protocols will be investigated before the simulation topologies and protocols are created.

An overview of these network topologies and protocols can be seen in chapter two and appendix B of this thesis. Header data from the TCP and IP headers will be extracted from the packets at different locations within the network and stored for evaluation. The simulation method can be seen in figure 1.3.1. A simulation network will be created and from this simulation network TCP/IP header data will be extracted at different locations within the network. The extracted data will then be analyzed and

a conclusion will be drawn on the most appropriate position within the network as well the OSI stack to extract TCP/IP header data.

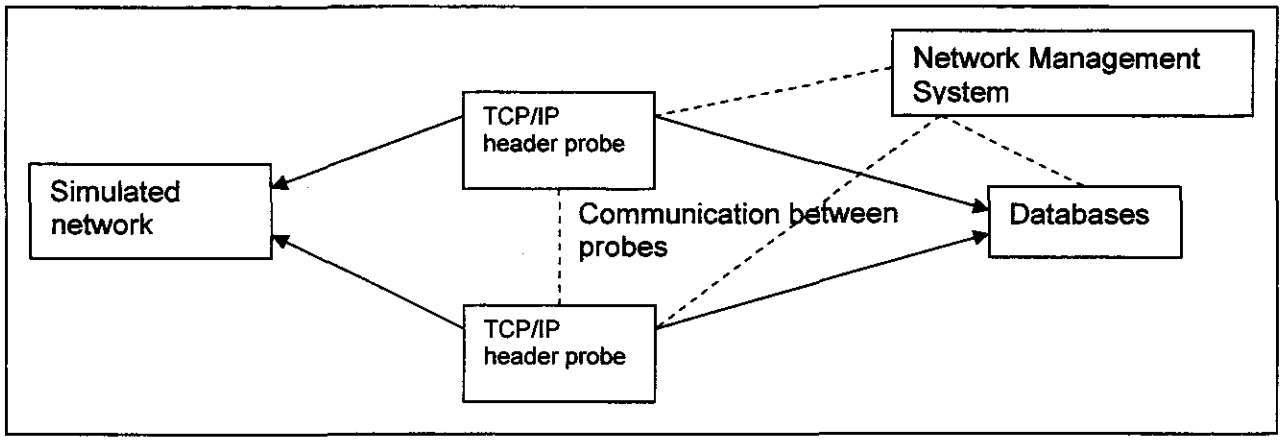

Figure 1.3.1 Simulation method

1.4 Research methodology and thesis layout

The structure of the research methodology can be seen in figure 1.4.1 followed by an explanation of each step. Chapter 1 has given the reader an introduction to the proposed research, problem scenario, and proposed methodology. Chapter 2 presents a background literature study of the different methods, topologies and protocols that will be used within the study. This study includes an overview and discussion of the OSI model as well as different topologies and protocols used within packet switched networks.

Chapter 3 gives a detailed discussion of the TCP/IP protocol suites. Along with detailed descriptions of all the fields contained within the headers. This chapter will also identify the different parameters needed to conduct QoS for certain CoS. Once these parameters have been identified, the parameters available for extraction from the IP and TCP headers for these services will be identified.

In chapter 4 a discussion is presented on the different methods used for telecommunication network modeling and the different tools that are available to perform these simulations. In chapter 5 the different test methods and network test topologies will be presented. Chapter 6 contains the results of the simulations in chapter 5. Chapter 7 contains a conclusion and a discussion of the results as well as future studies possible from this study. This study structure is summarized in table 1.4.1.

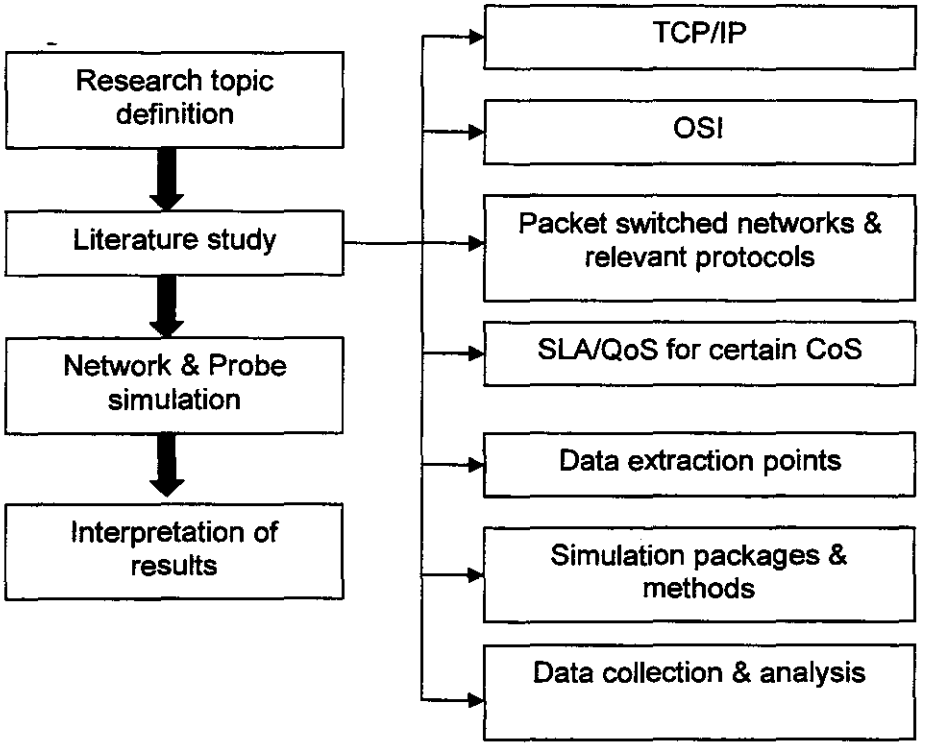

Figure 1.4.1 Research methodology layout

Chapter	Contents
Chapter 1: Introduction	• Introduction • Problem statement • Research methodology • Project beneficiaries
Chapter 2: Background study and literature overview	• OSI Overview • TCP/IP introduction • Circuit and packet switched network technology and topology overview
Chapter 3: TCP/IP in detail	• IP and TCP protocols • IP and TCP header field descriptions • IP and TCP header extractable data
Chapter 4: Network simulation techniques and tools	• Network simulation techniques and approaches overview • Network simulation packages and package comparison overview
Chapter 5: Problem methodology	• Simulated network topologies • Simulation tests

Chapter 6: Results	• Simulation results
Chapter 7: Conclusion	• Results discussion • Identification of further fields of study • Conclusion

Table 1.4.1 Thesis layout.

Chapter 2. Background study

Abstract – The aim of this chapter is to give the reader an overview of the different concepts used throughout this document. This chapter will therefore give an overview of fundamental networking concepts such as circuit switching, packet switching, OSI and TCP/IP. A more detailed discussion of TCP/IP can be found in chapter 3 of this document.

2.1 Circuit switching networks

Within a circuit switched network a fixed path between the source and the destination are created. Setting up, sending data and disconnecting the connection between the source and the destination are the three parts of a communication session with the use of circuit switched technology. With circuit establishment a circuit from the source to the destination must be created. This is achieved through switching through the network nodes until the destination is reached. Between each node a fixed connection is then established using Frequency Division Multiplexing (FDM) or Time Division Multiplexing (TDM), which is explained in appendix A (Network technology overview) of this document. After the connection is completed a test is made to determine if the destination is available to take the call or if the destination is busy. An example of such a transaction can be seen in figure 2.1.1 [5,6,7].

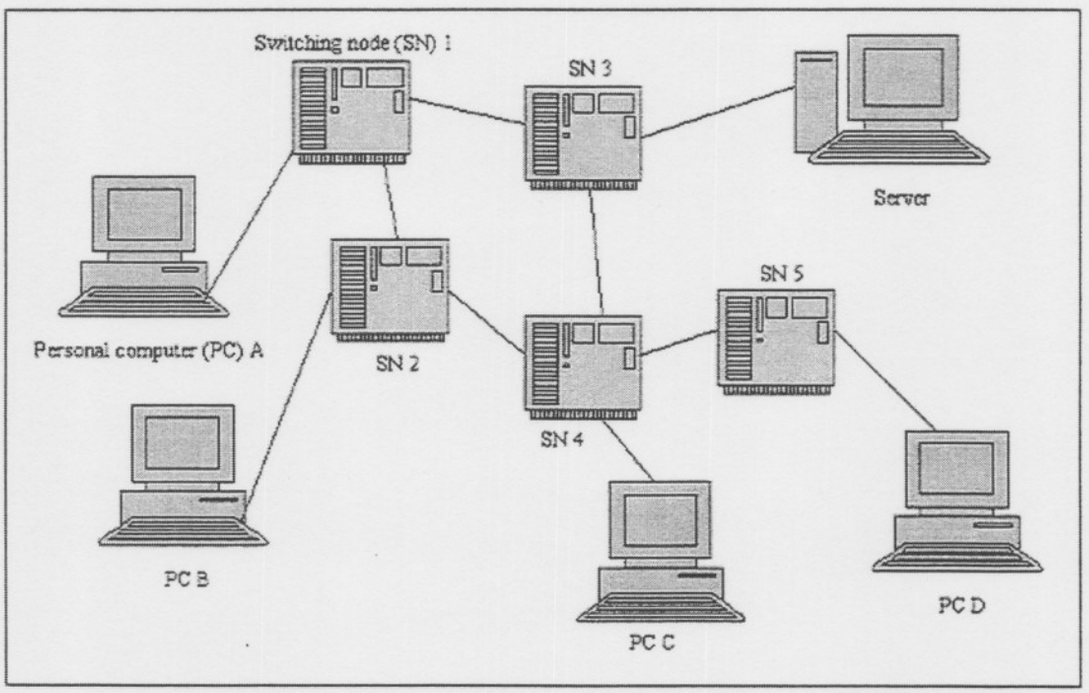

Figure 2.1.1. Creating a link over a circuit switched network

In figure 2.1.1 if PC A wants to communicate with PC D it will have to establish an end-to-end connection. This is achieved through transmitting a request to SN2

requesting a connection to PC D. SN 2 then finds a route to SN 4, which in turn finds a route to SN 5. SN 5 then establishes a connection with PC D. After the circuit has been established between the source and the destination the transmission of data is possible. The type of data may include analog data or digital data. The latter become the predominant type of data transmitted through the network for voice and video. After the data has been transmitted through the network, the call must be terminated with a terminating signal from one of the terminals. These connections are full duplex, and data can only be transmitted after the connection has been established [5,7].

The disadvantage of circuit switching is that it dedicates a certain bandwidth to the connection through the entire duration of the call. For voice transmission the bandwidth utilization may be high but for data transmission the bandwidth utilization isn't optimal. Figure 2.1.2 shows a typical circuit switched connection used within telephone networks where the subscriber is connected to the network via a subscriber line (subscriber loop). This loop is normally made of twisted pair and is a connection between the subscriber and the telecommunication network [5].

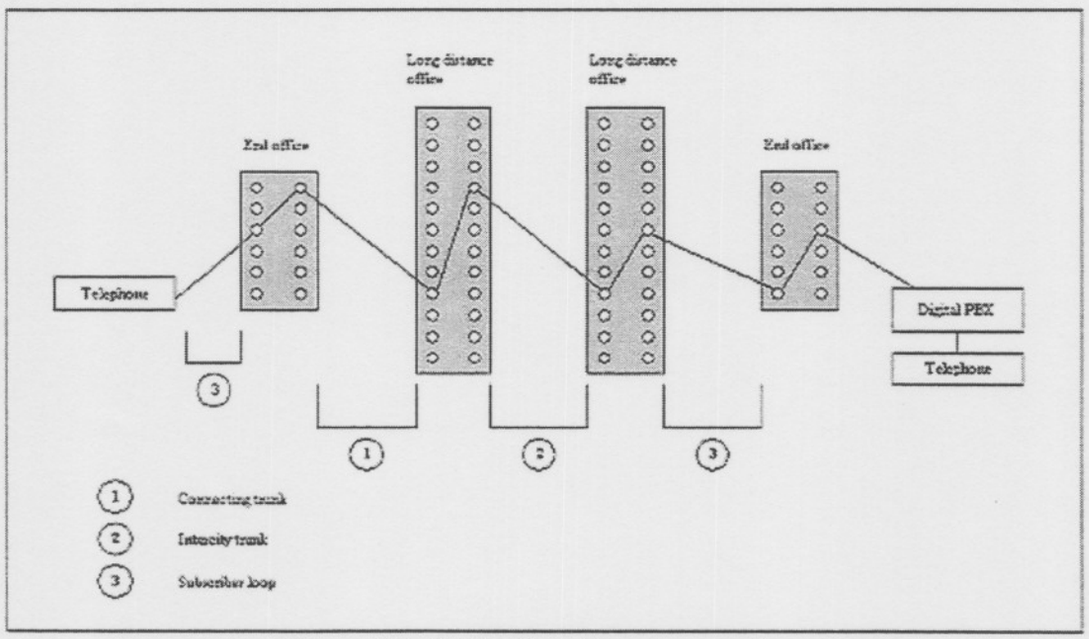

Figure 2.1.2. Circuit switched connection

The heart of a modern circuit switched network is the digital switch, which must provide a transparent full duplex signal path between any pair of attached devices. The different elements that make up a circuit switch node can be seen in figure 2.1.3.

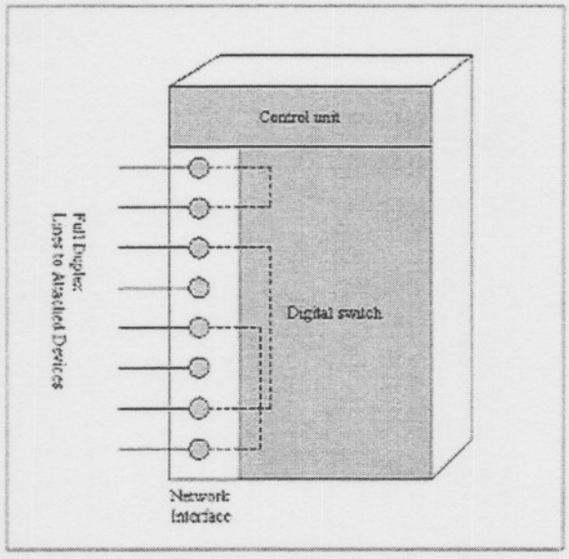

Figure 2.1.3 Elements of a circuit switch node

In figure 2.1.3 the network interface element represents the functions and hardware that is needed to connect to digital devices. The control unit performs three general tasks within the switching node. Firstly it establishes connections, secondly it maintains the established connections. And thirdly it does connection tear down between two connected devices. Different switching techniques are used within the switching node, which includes space division switching and time division switching for example. In a space division approach the signal paths are physically separated from each other in space. Each connection therefore requires the establishment of a physical path through the switch (Figure 2.1.4 (a)). The limitations of this switching technique are,

1. Firstly, the number of cross points grows with the square of the number of attached stations.
2. Secondly, the loss of crosspoint prevents connection between two devices whose lines intersect at that crosspoint.
3. Thirdly, the crosspoints are inefficiently utilized.

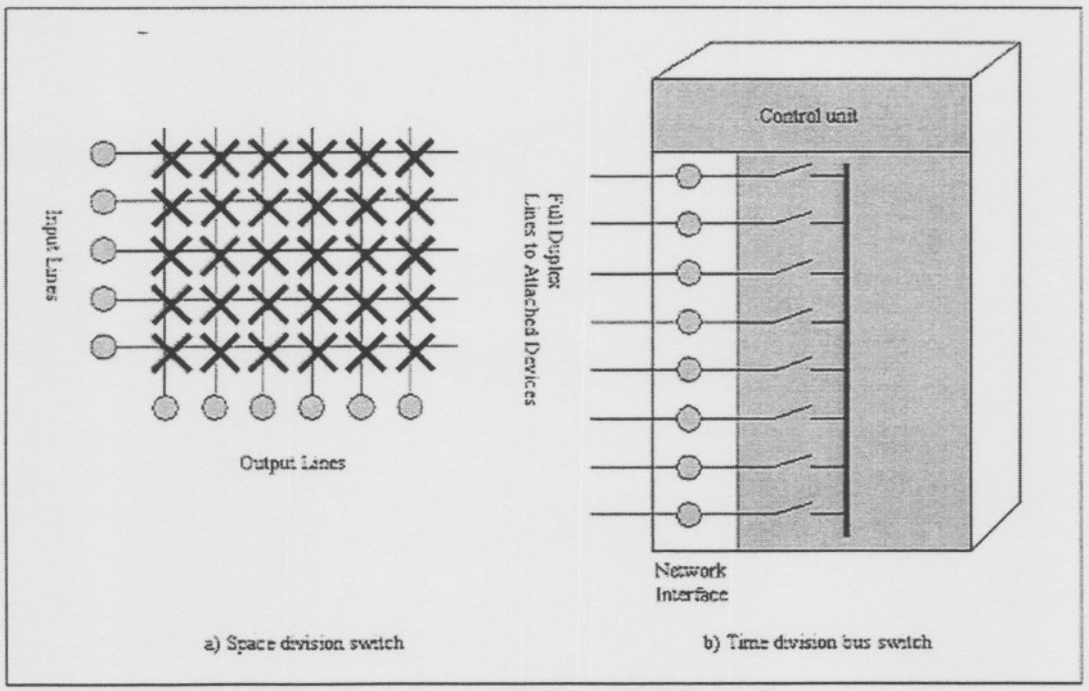

Figure 2.1.4 Space and time division switches

These shortcomings can be overcome through the use of multiple stage switches. Time division switching however involves portioning of lower speed bit streams into pieces that share a higher speed stream with other bit streams as can be seen in figure 2.1.4 (b). The inputs are sampled in turns, with the samples organized serially into slots and the number of slots equal to the number of inputs. Thus in figure 2.1.4 (b) a certain input are enabled for a short burst of data and at that same time a certain output is enabled establishing communication between the two devices for that short time space. A discussion of routing a call over long distances with more than one hub and switch can be found in appendix A of this document.

2.2 Packet switching networks

Telecommunication networks were originally constructed of circuit switched network technologies and was used to handle predominantly voice traffic, but this approach had shortcomings when data was transmitted over the network. These shortcomings included bandwidth utilization and interconnection problems due to fixed data rates. The problem needed to be solved and the answer was packet switched networks instead of circuit switched networks [5,6].

Packet switched networks provides more efficient bandwidth utilization for bursty data traffic than circuit switched networks. The data is transmitted in packets over the

network, with each packet containing information and control overhead. Virtual circuits can be established within a packet switched network for data packet transmission. Each packet can also be transmitted and treated independently within the network. If the latter is the case it is referred to as datagrams. Packet switched networks have many advantages including better bandwidth utilization, flexibility and resource sharing. These advantages however are at a cost. Some examples of PSN technologies are frame relay, Asynchronous Transfer Mode (ATM), Switched Megabit Data Services (SMDS) and X.25 [5,6,9].

Packet switched networks uses packets constructed of data and network overhead for transmission. The overhead includes information concerning the destination, origin and network routing data necessary to transmit the packet through the network. The packet is transmitted through the network and at each node it is briefly stored before being transmitted through the remaining network. This approach included advantages such as better bandwidth utilization, line efficiency, data rate conversion capabilities and priority sending. Figure 2.2.1 shows the use of packets for transmitting data over a packet switched network [5,7,6].

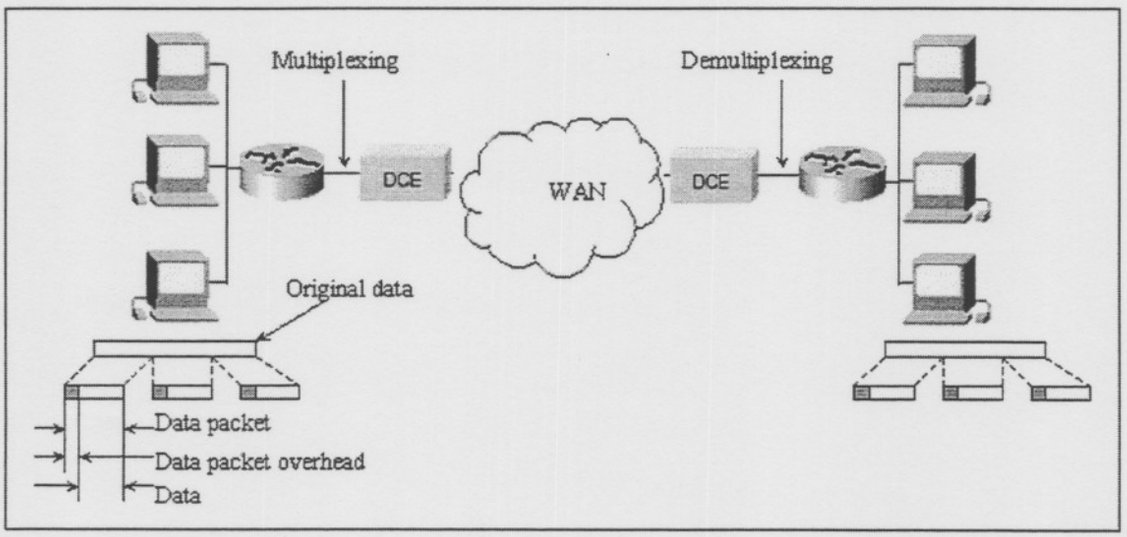

Figure 2.2.1 Transmission of packets over a packet switched network

In figure 2.2.1 the transmitted data are divided up into packets and multiplexed with other packets and transmitted through the network. On the receiving end demultiplexing takes place and the original data are constructed from the received packets. There are different ways of sending the packets over the network that includes datagrams and virtual circuits.

In a datagram approach each packet is treated independently and has no reference to the other transmitted packets. These packets could arrive in a different sequence at the destination than they were transmitted from the transmitting station. In figure 2.2.2 if station B wants to transmit to station D and the number of packets it wants to transmit is four then the datagram approach will work as follows.

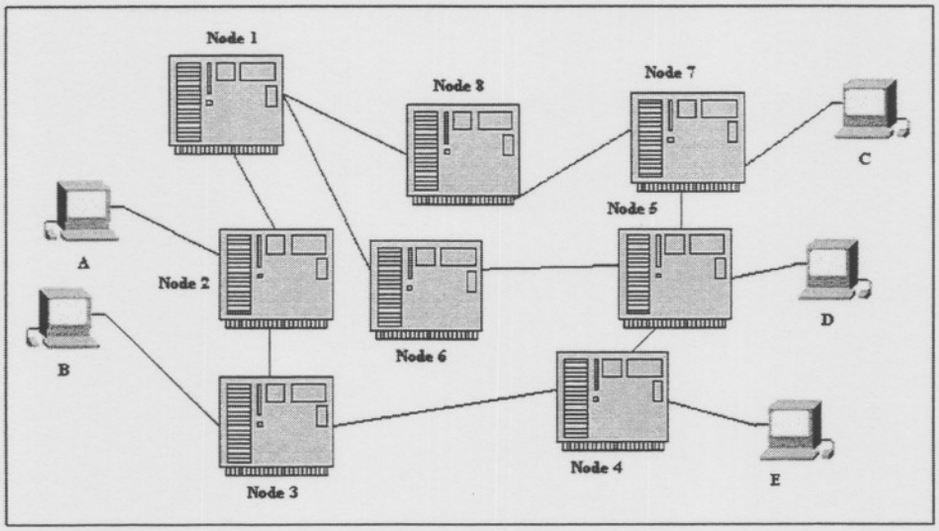

Figure 2.2.2 Transmission of data across a packet switched network

Station B transmits the packets (1,2,3 and 4) to Node 3 with each packet containing the destination address. Node 3 must now make routing decisions for the different packets. Node 3 can forward these packets to Node 2 or Node 4, if Node 2 has a shorter queue of incoming packets than Node 4 then Node 3 will transmit packet 1 to Node 2. Node 3 has to make the same decision for packets 2,3 and 4, Lets assume for packets 2,3 and 4 the queue to Node 4 is the shortest.

Packets 2,3 and 4 are hen transmitted to Node 4 and Node 4 transmits them to E. Packet 1 is at Node 2, Node 2 transmits the packet 2 to Node1. Node 1 now has to determine if it will transmit packet 2 to Node 6 or Node 8. Lets assume Node 6 has a shorter queue than Node 8 thus packet 2 is transmitted to node 6. Node 6 transmits packet 2 to Node 5, which in turn transmits packet 2 to Node 4 and Node 4 transmits packet 2 to E [5].

The arrival sequence of the packets isn't the same as the sequence it was transmitted in. It must also be remembered that if Node 3 fails all the packets in its queue (which contains packets 2,3 and 4) can be lost. This leaves E to figure out that

some of the packets have been lost in transit and need to be retransmitted. There are advantages when transmitting packets over a packet switched network with the datagram approach. These advantages include avoiding setup time as well as being more flexible, enabling data packets to be transmitted away from congestion within the network [5].

The second approach available for transmitting data packets across the network is virtual circuits. If station A needs to transmit data to station E using a virtual circuit approach, station A would transmit a control packet (Call request packet) to Node 2 requesting a logical connection to E. Node 2 routes the request to Node 3 which routes the request to Node 4. E receives the control packet and transmits a call accept packet to station A back through Nodes 4,3 and 2.

The route is now established and A can transmit to station E and receive data over the virtual connection. If the data transmission is completed the virtual channel can be closed with the use of a clear request packet which terminates the connection. Nodes can uphold and sustain more than one virtual circuit [5].

Thus a route is established between the two stations prior to data transmition. This route isn't the same as the route in a circuit switched network, because the packets are still stored and queued for output within a packet switched network. The disadvantage is that if a node fails all the virtual connections through that node will be lost. Virtual circuits are normally used in ATM, frame relay and X.25. The advantages when using a virtual circuit approach are sequencing, error control and higher data rates. This approach has higher data rates because decisions aren't made on a per packet basis [5].

When a comparison between circuit switching networks and packet switching networks must be drawn up aspects such as performance, different delay times and transmission times must be mentioned. These differences can be represented within the following table.

	Circuit Switching	Packet switching	
		Datagrams	Virtual circuits
Transmission path	Dedicated	No dedicated path	No dedicated path
Packet/continuous transmission of data	Continuous transmission of data	Transmission of packets	Transmission of packets
Fast enough for interactivity	Yes	Yes	Yes
Message storing	Messages aren't stored	Packets may be stored until delivery	Packets stored until delivery
Duration of established route	Established path/route for entire duration of the call	Path/Route established for each individual packet	Established path/route for entire duration of the call
Call set-up delay	Yes	No	Yes
Transmission delay	Negligible	Yes	Yes
Bandwidth	Fixed bandwidth	Dynamic use of bandwidth	Dynamic use of bandwidth
Overhead	No overhead after call set-up	Overhead in each packet	Overhead in each packet
Type of switching	Electromechanical, computerized	Small switching nodes	Small switching nodes

Table 2.2.1 Comparison between packet and circuit switching networks

Figure 2.2.3 shows a comparison between virtual packet switching, datagram packet switching and circuit switching between two points through four nodes. In figure 2.2.3 it is clear that the only delay in circuit switching is the delay in establishing the connection between the two points. After the connection has been established the data transmission delay is negligible.

For virtual circuit packet switching and datagram packet switching there is a process delay at each node for every packet. Virtual circuit packet switching has the extra delay of establishing the virtual circuit through the nodes before data transmission can begin. Because datagram transmission doesn't have path-establishing times it is faster for smaller amounts of data.

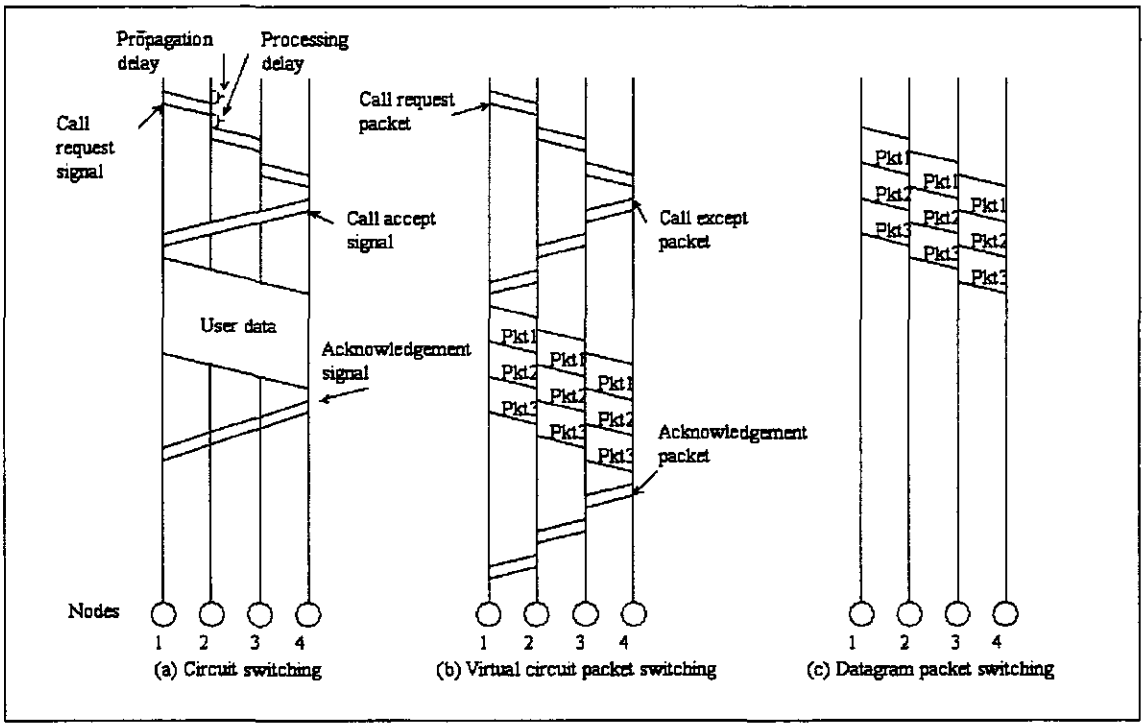

Figure 2.2.3 Comparison between circuit switching, virtual circuit packet switching and datagram packet switching

The best method for transmitting voice and video over a data network is with the use of virtual circuit switching. Real time voice and video application requires low jitter. With the use of virtual circuits the connection between the two points can be created and voice and video data can be transmitted at a speed suitable for real time interactivity.

When routing packets within a packet switching network more than one route can be established. These routes however must satisfy certain requirements. These requirements include correctness, simplicity, robustness, stability, fairness, optimality and efficiency. When a route is selected, that route is normally selected on the basis of some performance criteria, which can include the number of hops, cost, delay and throughput.

2.3 Open system interconnection (OSI) model

The OSI model was developed by the ISO (Organization for standardization) as a model for computer communication architecture and as a framework for developing protocol standards. The main task of the OSI was to develop and to define a set of layers and services for each layer that would partition group functions logically as well as keeping the number of layers substantially small so that the overhead wouldn't become cumbersome. The different principles used in guiding the design of the OSI model were summarized as follows [5,6].

- Keep the numbers of layers as low as possible to minimize the system-engineering task of describing and integrating the different layers.
- A boundary should be created at the point where the description of services can be small and the number of interactions across the boundary is minimized.
- Separate layers should be created to handle functions that are manifestly different in the process performed or the technology involved.
- Similar functions should all be collected within the same layer.
- Boundaries should be chosen at a point that has proven to be successful in the past.
- A layer should be created so that it has easily localized functions that could be changed in a major way so that it can take advantage of new advances in architecture, hardware, or software technology without changing the services expected from and provided to the adjacent layers.
- A boundary should be created where it may be useful in the future to have the corresponding interface standardized.
- A layer should be created where there are needs for different levels of abstraction in the handling of data. This data may include for example morphology, syntax and semantic data.
- If changes are made to functions or protocols it should be allowed within the layer in a manner so that it doesn't affect the other layers surrounding that specific layer.
- Each layer should be created so that its boundaries are limited by its upper and lower layers.

The following guidelines are also applied to the layers.

- Further sub grouping and organization of functions should be created to form sub layers within each respective layer in cases where it may be required by distinct communication services.
- Create where needed, two or more sub layers with a common, and therefore minimal, functionality to allow interface operation with adjacent layers.
- Bypassing of sub layers must be allowed.

The OSI model consists of seven interconnected layers with each layer performing certain functions. The seven layers can be seen in figure 2.3.1 followed by a short description of each layer [5,6].

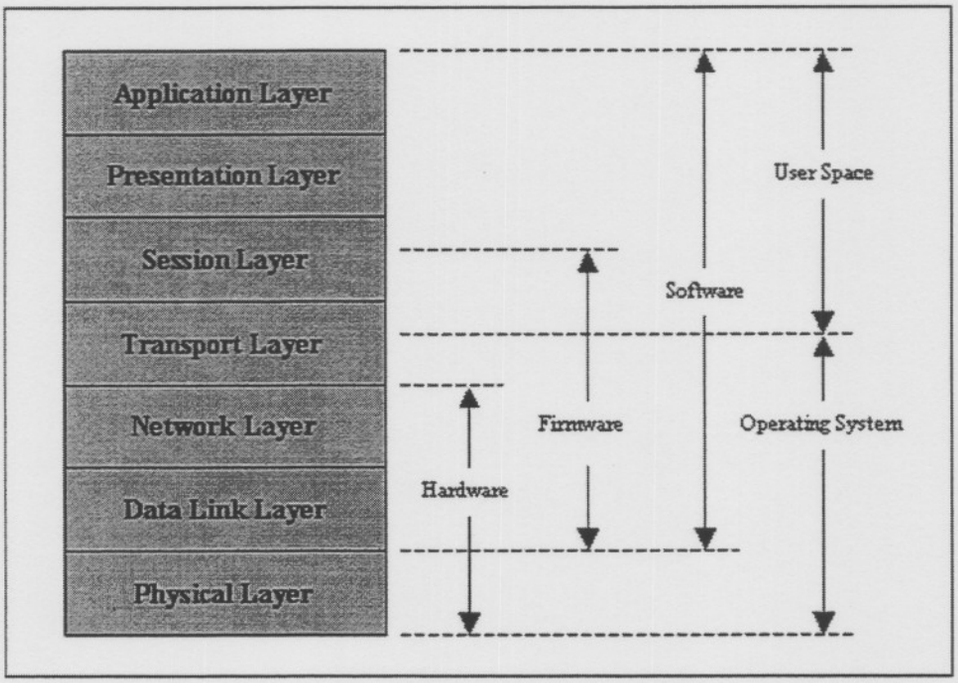

Figure 2.3.1 The seven different OSI Layers

The functions of the different layers are as follows;

- ***Application Layer***

The application layer provides access to the OSI environment for users and it also provides distributed information services. The application layer thus provides an interface between the software running on the computer and the network [5,10,6].

- ***Presentation Layer***

The presentation layer provides independence to the application processes from differences in data representation (syntax). It therefore performs code conversion and data reformatting (syntax translation).

- ***Session Layer***

The function of the session layer is to provide the control structure for communication between applications and establishes, manages and terminates connections (sessions) between cooperating applications. Thus the session layer decides when to turn communication on and off between two computers and it provides the mechanisms that control the data-exchange process and coordinates the interaction between them. It sets up and clears communication channels between two communicating components. Unlike the network layer (layer 3), it deals with the programs running in each machine to establish conversations between them. Some of the most commonly encountered protocol stacks, including TCP/IP, don't implement a session layer.

- ***Transport Layer***

The transport layer provides reliable, transparent transfer of data between end points as well as end-to-end error recovery and flow control. If the data is transmitted incorrectly this layer has the responsibility to ask for the re-transmission of the data. This layer acts as an interface between the bottom three layers and the top three layers by providing layer 5 (Session layer) with a reliable message transfer service. It thus hides the detailed operation of the underlying network to the session layer.

- ***Network Layer***

The network layer provides the upper layers with independence from the data transmission and switching technologies used to connect systems that is responsible for establishing, maintaining and terminating connections.

- ***Data Link Layer***

The data link layer provides reliable transfer of information across the physical link and transmits blocks (frames) with the necessary synchronization, error control and flow control. Thus the data link layer provides the network layer (layer 3) with reliable information-transfer capabilities. The data-link layer is often subdivided into two parts-Logical Link Control (LLC) and Medium Access Control (MAC)-depending on the implementation [5,10,6].

- ***Physical Layer***

The physical layer is concerned with the transmission of unstructured bit streams over physical mediums and deals with the mechanical, electrical, functional and procedural characteristics to access the physical medium [5,10]. A more detailed description of each layer can be found later on in the chapter.

The Open System Interconnection (OSI) model includes a set of protocols that attempt to define and standardize the data communications process. Into the above mentioned seven layers are fitted the protocol standards developed by the ISO and other standards bodies, including the Institute of Electrical and Electronic Engineers (IEEE), American National Standards Institute (ANSI), and the International Telecommunications Union (ITU), formerly known as the CCITT (Comité Consultatif International Téléphonique et Télégraphique) [10].

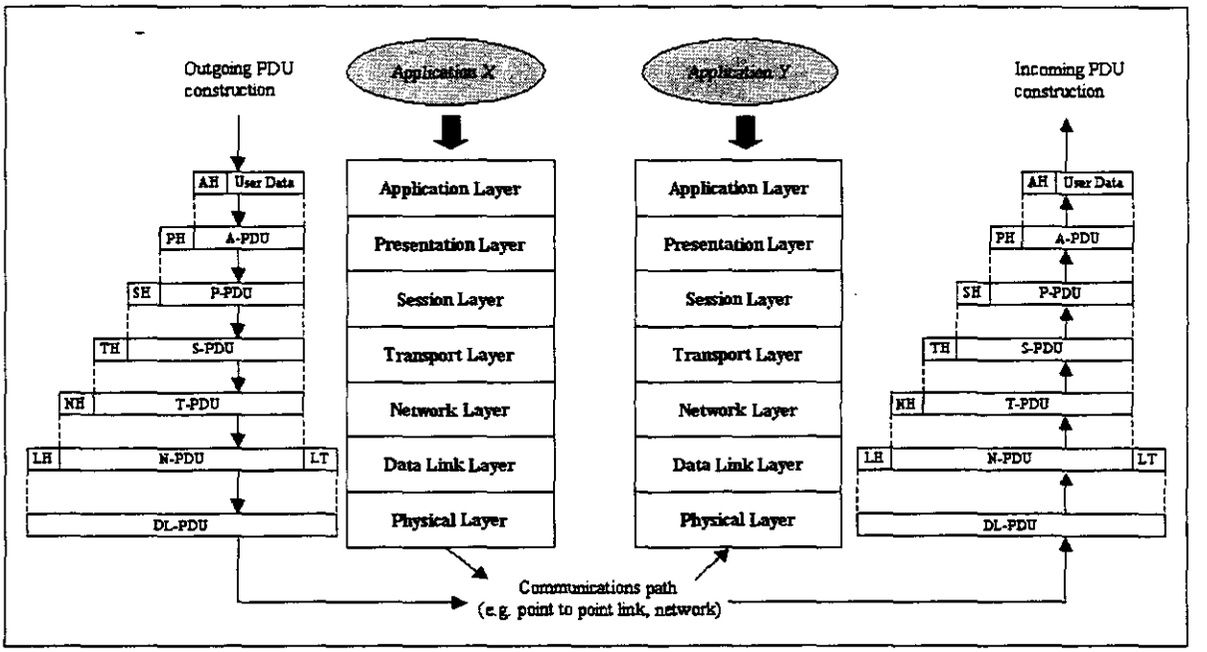

Figure 2.3.2 The OSI environment

Figure 2.3.2 illustrates the OSI architecture between two systems. Each system has seven layers as previously discussed. If application X running on the transmitting system wants to transmit a message to application Y on the receiving system, it will invoke the application layer (Layer 7). The application layer of the transmitting system will then establish a peer relationship with the application layer on the receiving system through the use of a layer 7 protocol. This protocol how ever requires the services of layer 6 the presentation layer, exactly the same applies for the presentation layer who requires the services of the session layer (Layer 5), and so on down to the physical layer who requires the services of the data link layer [5].

It must also be noted that there is no direct communication between the two peer entities except at the physical layer. When application X transmits a message to application Y, application X transmits the message to the application layer of the transmitting system, which then appends a header to the data that is going to be transmitted. The application layer then passes on the data with the appended header down to the presentation layer who treats the whole unit as data and then appends it's own header to the unit.

This process is followed through for each layer until it reaches layer two where a header and a trailer are added onto the unit. This entire unit is then passed onto the physical layer, which transmits the unit over the physical medium to the receiving

system. This transmission medium could be a packet switched topology or a circuit switched topology. At the receiving end the packet is disassembled as it is passed up through the layers until it reaches the application layer where the receiving application can use the data transmitted by the transmitting application [5].

The different layers within the model communicate with each other through the use of protocols. One or more standards can be developed at each layer for these protocols. The model in general terms defines functions that must be performed at the layer and that facilitate the standard- making process in two ways.

Firstly because the functions of each layer are well defined, standards can be developed independently and simultaneously for each layer. This speeds up the standards-making process. Secondly because the boundaries between these different layers are well defined, changes in standards in one layer need not affect already existing software in another layer making it easier to introduce new standards [5].

2.4 TCP/IP protocol suite

The TCP/IP (TCP – Transmission protocol, IP – Internet protocol) protocol suite recognizes that the task of communications is too complex and too diverse to be accomplished by a single unit. Accordingly, the task is broken up into modules or entities that may communicate with peer entities in another system. The TCP/IP protocols are part of the TCP/IP protocol suite with the TCP protocol providing connection-orientated services for higher layer applications.

These connections are created by the IP protocol which routes the packets through the network to create these connections. The communication task of TCP/IP can be organized into five relatively independent layers, which can be seen in figure 2.4.1 where it is compared with the OSI stack [6].

Physical layer: The physical layer covers the physical interface between a data transmission device and a transmission medium or network.

Network access layer: The network access layer is concerned with the exchange of data between the end system and the network to which it is attached. Thus it must

take care of access and routing of data across a network for two end systems that is attached to the same network.

Internet layer: If the two end systems are not connected to the same network, procedures are needed for the data to traverse multiple interconnected networks. These procedures are provided by the Internet layer, in which the IP protocol is implemented.

Host-to-host, or transport layer: The nature of the applications that will transmit and receive data may be different. However this data must be exchanged in a reliable fashion, meaning that the data transmitted must all arrive at the destination and the data must arrive in the same order as they were transmitted. These mechanisms that are needed to provide reliable transmission of data are provided by the host-to-host or transport layer. In this layer the TCP protocol are implemented.

Application layer: The application layer contains the logic needed to support the various user applications.

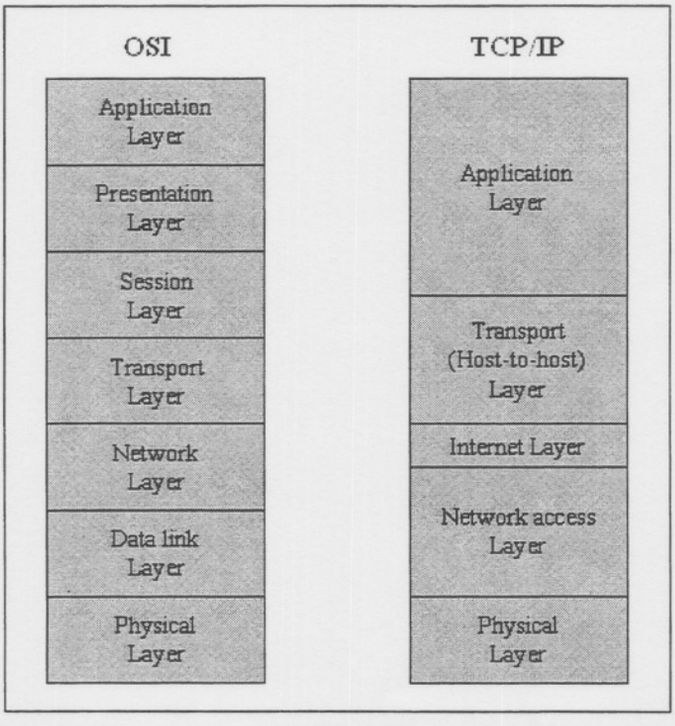

Figure 2.4.1 Comparison between the TCP/IP and the OSI protocol architectures

For successful communication each entity in the overall system must have an unique address and indeed two levels of addressing. Each host on the network must have a

unique global Internet address, as well as an address that is unique to the host. The unique global Internet address allows the data to be delivered to the desired address, and the host address allows the data to be delivered host-to-host with the use of the TCP address [5,6].

The IP protocol is implemented in all the end systems as well as the routers and acts as a relay to move blocks of data through these routers from host –to-host. TCP however is implemented in the end systems only and keeps track of the blocks of data that have been transmitted with the use of the IP protocol to assure that they have been delivered reliably and accurately. A typical network packet contains a data packet to which a TCP header, IP header and a network header have been added as illustrated in figure 2.4.2.

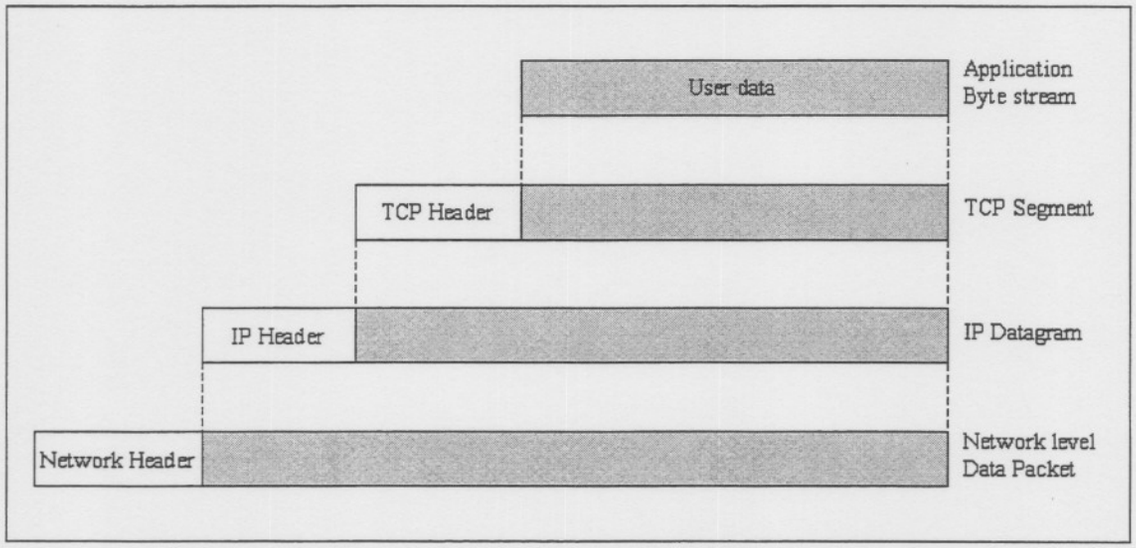

Figure 2.4.2 Protocol data units in the TCP/IP architecture

Different control information is added to the data through the use of the TCP and IP headers. The TCP header contains firstly the destination port address, secondly the sequence number and thirdly the checksum to check for an error in the transmission. The IP address however contains control information for the packet to find it's way through the network to the destination address. After the TCP and IP header are added the packet (IP datagram) are passed onto the network access layer which ads it's own header to create a packet or a frame. Information contained within these headers includes the destination network addresses and the facilities requests.

The TCP and IP protocols are part of the TCP/IP protocol suite, which can be seen in figure 2.4.3 with TCP providing connection-orientated services for higher layer

applications and IP providing routing for the packets. These higher layer application services in turn then provide specific services to the Internet users. These services include SMTP, TELNET and FTP amongst others.

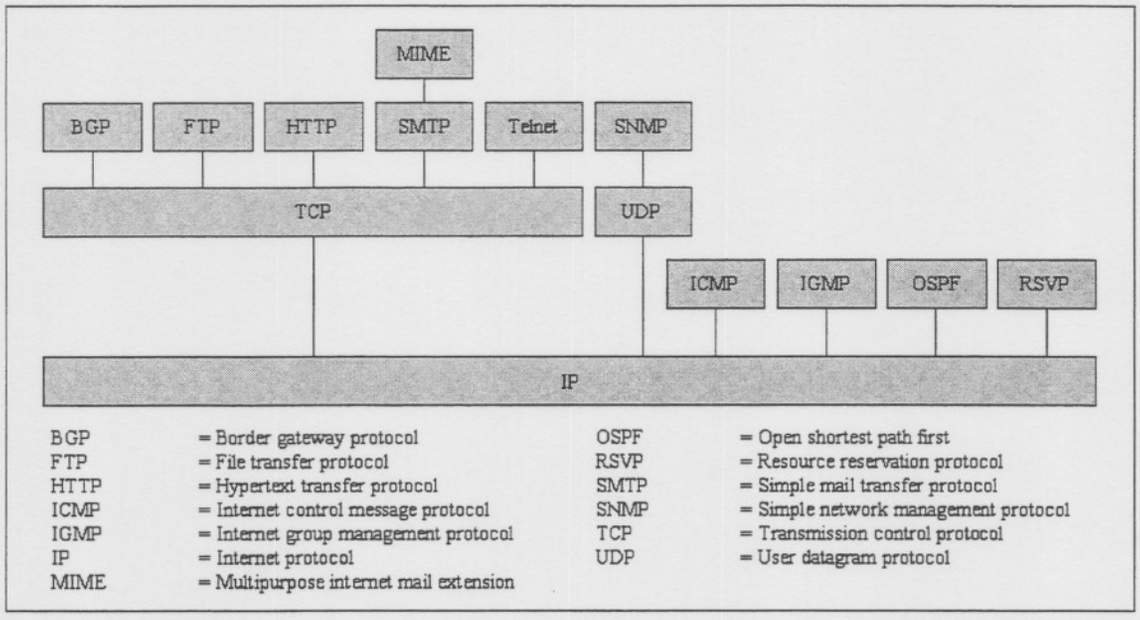

Figure 2.4.3 TCP/IP protocol suite

2.4.1 IP Protocol

The Internet Protocol (IP) is a network-layer (Layer 3) protocol that contains addressing information and some control information that enables packets to be routed. IP is documented in [RFC 791] and is the primary network-layer protocol in the Internet protocol suite. Along with TCP, IP represents the heart of the Internet protocols and has two primary responsibilities.

Firstly it must provide connectionless, best-effort delivery of datagrams through an inter-network of networking devices and secondly it must provide fragmentation and reassembly of datagrams to support data links with different maximum-transmission unit (MTU) sizes. An IP packet contains several types of information as can be seen from the IP protocols header in figure 2.4.4. Chapter 3 contains an in detail discussion of the IP and TCP header fields and their usage in upholding QoS for different CoS.

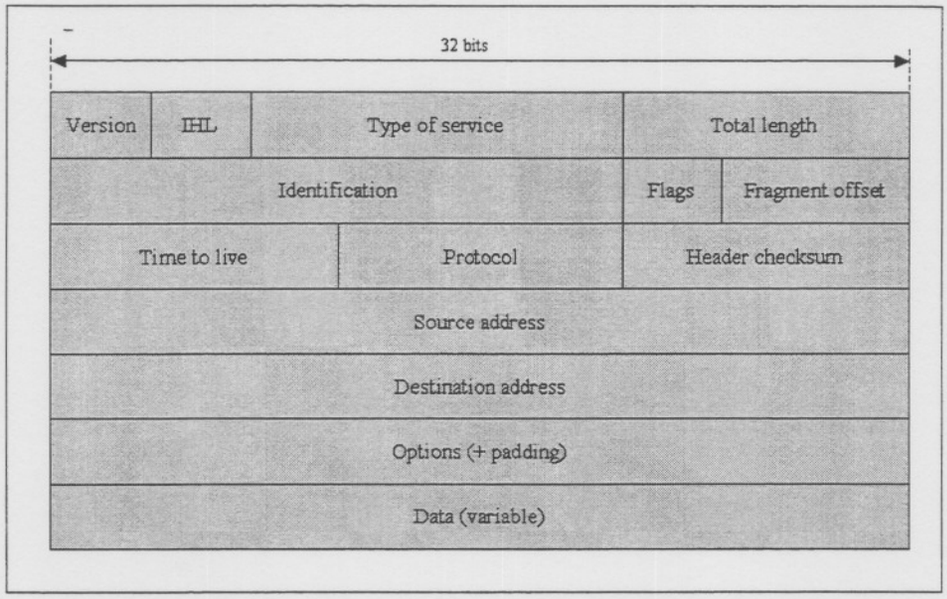

Figure 2.4.4. IP packet fields

The information contained within these fields is as follows.

- ***Version (4 bits)***: Indicates the version of IP currently used.
- ***IHL (4 bits)***: Indicates the datagram header length in 32-bit words.
- ***Type of service (8 bits)***: Specifies how an upper-layer protocol would like a current datagram to be handled, and assigns datagrams various levels of importance.
- ***Total length (16 bits)***: Specifies the length, in bytes, of the entire IP packet, including the data and header.
- ***Identification (16 bits)***: Contains an integer that identifies the current datagram. This field is used to help piece together datagram fragments.
- ***Flags (3 bits)***: Consists of a 3-bit field of which the two low-order (least-significant) bits control fragmentation. The low-order bit specifies whether the packet can be fragmented. The middle bit specifies whether the packet is the last fragment in a series of fragmented packets. The third or high-order bit is not used.
- ***Fragment offset (13 bits)***: Indicates the position of the fragment's data relative to the beginning of the data in the original datagram, which allows the destination IP process to properly reconstruct the original datagram.
- ***Time to live (8 bits)***: Maintains a counter that gradually decrements down to zero, at which point the datagram is discarded. This keeps packets from looping endlessly.

- ***Protocol (16 bits)***: Indicates which upper-layer protocol receives incoming packets after IP processing is complete.
- ***Header checksum (16 bits)***: Helps ensure IP header integrity.
- ***Source address (32 bits)***: Specifies the sending node.
- ***Destination address (32 bits)***: Specifies the receiving node.
- ***Options (variable)***: Allows IP to support various options, such as security.
- ***Data variable***: Contains upper-layer information [5,6,11].

A detailed discussion of the IP addressing method can be found in appendix A of this document and a detailed discussion of the IP protocol can be found in chapter 3 of this document.

2.4.2 TCP protocol

The TCP provides reliable transmission of data in an IP environment. TCP corresponds to the transport layer (Layer 4) of the OSI reference model. Among the services TCP provides are stream data transfer, reliability, efficient flow control, full-duplex operation, and multiplexing. With stream data transfer, TCP delivers an unstructured stream of bytes identified by sequence numbers. This service benefits applications because they do not have to chop data into blocks before handing it off to TCP. Instead, TCP groups bytes into segments and passes them to IP for delivery.

TCP offers reliability by providing connection-oriented, end-to-end reliable packet delivery through an inter-network. It does this by sequencing bytes with a forwarding acknowledgment number that indicates to the destination the next byte the source expects to receive. Bytes not acknowledged within a specified time period are retransmitted. The reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets. A time-out mechanism allows devices to detect lost packets and request retransmission [5,6,11].

TCP offers efficient flow control, which means that, when sending acknowledgments back to the source, the receiving TCP process indicates the highest sequence number it can receive without overflowing its internal buffers. Full-duplex operation means that TCP processes can both send and receive at the same time. Finally, TCP's multiplexing means that numerous simultaneous upper-layer conversations can be multiplexed over a single connection. The construction of a TCP packet can

be seen in figure 2.4.5 followed by a description of every field within the packet header.

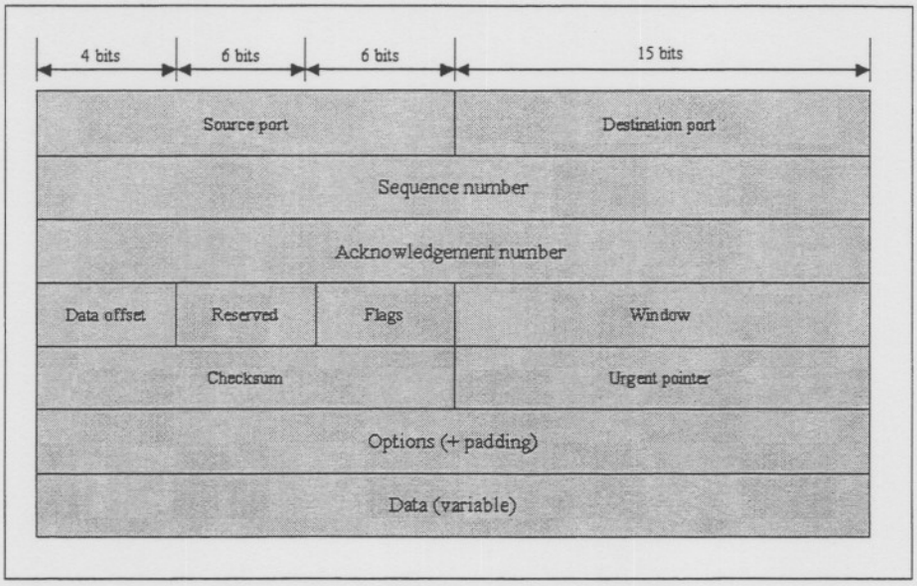

Figure 2.4.5. TCP packet construction

The information contained within these fields is as follows.

- ***Source port (16 bits) and destination port (16 bits)***: The source and destination port numbers identifies points at which upper-layer source and destination processes receive TCP services.
- ***Sequence number (32 bits)***: The sequence number usually specifies the number assigned to the first byte of data in the current message. In the connection-establishment phase, this field can also be used to identify an initial sequence number to be used in an upcoming transmission.
- ***Acknowledgement number (32 bits)***: The acknowledgement number contains the sequence number of the next byte of data the sender of the packet expects to receive.
- ***Data offset (4 bits)***: This value indicates the number of 32-bit words in the TCP header.
- ***Reserved (6 bits):*** This field remains reserved for future use.
- ***Flags (6 bits)***: The flags field carries a variety of control information, which includes.

  URG: Urgent pointer field significant.
  ACK: Acknowledgement field significant.

PSH: push function.
RST: Reset the connection.
SYN: Synchronize the sequence numbers.
FIN: No more data from sender.

- ***Window (16 bits)***: This field specifies the size of the sender's receive window (that is, the buffer space available for incoming data).
- ***Checksum (16 bits)***: Indicates whether the header was damaged in transit.
- ***Urgent pointer (16 bits)***: Points to the first urgent data byte in the packet.
- ***Options (16 bits)***: The options field specifies various TCP options.
- ***Data***: Contains upper-layer information [5,6,11].

2.4.2.1 TCP connection establishment

To use reliable transport services, TCP hosts must establish a connection-oriented session with one another. Connection establishment is performed by using a "three-way handshake" mechanism. A three-way handshake synchronizes both ends of a connection by allowing both sides to agree upon initial sequence numbers. This mechanism also guarantees that both sides are ready to transmit data. This is necessary so that packets are not transmitted or retransmitted during session establishment or after session termination.

Each host randomly chooses a sequence number used to track bytes within the stream it is sending and receiving. Then, the three-way handshake proceeds in the following manner: The first host (Host A) initiates a connection by sending a packet with the initial sequence number (X) and SYN bit set to indicate a connection request. The second host (Host B) receives the SYN, records the sequence number X, and replies by acknowledging the SYN (with an ACK = X + 1). Host B includes its own initial sequence number (SEQ = Y). An ACK = 20 means the host has received bytes 0 through 19 and expects byte 20 next. This technique is called forward acknowledgment. Host A then acknowledges all bytes Host B sent with a forward acknowledgment indicating the next byte Host A expects to receive (ACK = Y + 1) [11].

2.4.2.2 Positive acknowledgement and retransmission (PAR)

A simple transport protocol might implement a reliability-and-flow-control technique where the source sends one packet, starts a timer, and waits for an acknowledgment before sending a new packet. If the acknowledgment is not received before the timer expires, the source retransmits the packet. Such a technique is called positive acknowledgment and retransmission (PAR).

By assigning each packet a sequence number, PAR enables hosts to track lost or duplicated packets caused by network delays that result in premature retransmission. The sequence numbers are transmitted back in the acknowledgments so that the acknowledgments can be tracked.

2.4.2.3 TCP sliding window

A TCP sliding window provides more efficient use of network bandwidth than PAR because it enables hosts to send multiple bytes or packets before waiting for an acknowledgment. In TCP, the receiver specifies the current window size in every packet. Because TCP provides a byte-stream connection, window sizes are expressed in bytes. This means that a window is the number of data bytes that the sender is allowed to send before waiting for an acknowledgment.

Initial window sizes are indicated at connection setup, but might vary throughout the data transfer to provide flow control. A window size of zero, for instance, means, "Send no data." In a TCP sliding-window operation, for example, the sender might have a sequence of bytes to send (numbered 1 to 10) to a receiver who has a window size of five. The sender then would place a window around the first five bytes and transmit them together. It would then wait for an acknowledgment.

The receiver would respond with an ACK = 6, indicating that it has received bytes 1 to 5 and is expecting byte 6 next. In the same packet, the receiver would indicate that its window size is 5. The sender then would move the sliding window five bytes to the right and transmit bytes 6 to 10. The receiver would respond with an ACK = 11, indicating that it is expecting sequenced byte 11 next. In this packet, the receiver might indicate that its window size is 0 (because, for example, its internal buffers are full). At this point, the sender cannot send any more bytes until the receiver sends another packet with a window size greater than zero [11]. TCP congestion control

can be used to control the rate of TCP traffic flow. A detailed description of TCP congestion control can be found in appendix A of this document.

2.4.2.4 User datagram protocol (UDP)

UDP is a connectionless transport-layer protocol (Layer 4) that belongs to the Internet protocol family. UDP is basically an interface between IP and upper-layer processes. UDP protocol ports distinguish multiple applications running on a single device from one another. Unlike the TCP, UDP adds no reliability, flow-control, or error-recovery functions to IP. Because of UDP's simplicity, UDP headers contain fewer bytes and consume less network overhead than TCP. UDP is useful in situations where the reliability mechanisms of TCP are not necessary, such as in cases where a higher-layer protocol might provide error and flow control.

UDP is the transport protocol for several well-known application-layer protocols, including Network File System (NFS), Simple Network Management Protocol (SNMP), Domain Name System (DNS), and Trivial File Transfer Protocol (TFTP). The UDP packet format contains four fields, as shown in Figure 2.4.6. These include source and destination ports, length, and checksum fields [11].

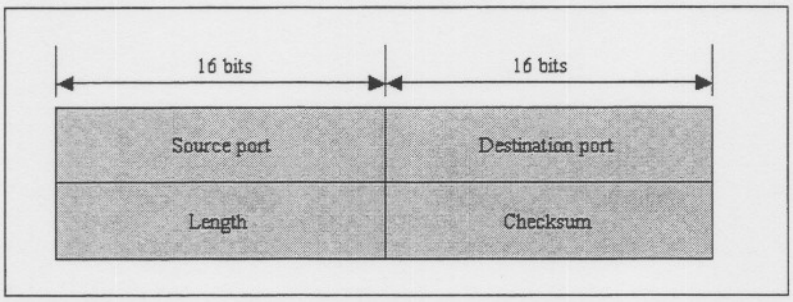

Figure 2.4.6. UDP header

Source and destination ports contain the 16-bit UDP protocol port numbers used to demultiplex datagrams for receiving application-layer processes. A length field specifies the length of the UDP header and data. Checksum provides an (optional) integrity check on the UDP header and data.

2.5 QoS and SLA's in packet switched networks

QoS can be defined as providing a certain flow in a network with better service than other flows. These better services can be provided over a variety of underlying

technologies, which includes Frame Relay, ATM, IP and routed networks for example. If QoS needs to be implemented on an end-to-end bases QoS capabilities must be incorporated on end system architecture software such as the end system operating system, as well as on network devices itself [5].

The different metrics that must be kept in mind when we are talking about QoS includes service availability, delay and delay jitter for example. Delay represents the time the packet takes to travel through the network from one end to the other. Whereas delay jitter is the variation packets encounter following the same route through the network.

Therefore QoS concerns both network elements such as routers, which must give a higher priority to certain flows than other flows, as well as end host software, which must construct these packets and their relevant headers. Two different approaches to QoS exist with two different services, integrated services (resource Reservation) and differentiated services (Prioritization). These different services are accomplished through a set of protocols and algorithms and can be classified as follows [5,6].

Reservation protocol (RSVP): RSVP provides the signaling to enable network resource reservation or integrated services. RSVP is typically used on a per flow bases but it could also be used to reserve aggregate resources.

The RSVP protocol is currently used in the IntServ architecture where it provides per flow signaling to the network. IntServ is used to quantify these QoS requirements using an admission-control-based approach. IntServ however suffers from scalability problems as well as complexity problems and deployment problems.

Differentiated services (DiffServ): DiffServ provides a coarse and simple way to categorize and prioritize network traffic (flow) aggregates. Although others are possible, there are currently two standards per hop behaviors (PHB's) defined that effectively represent two service levels or traffic classes.

- The first class is Expediting Forwarding (EF), which contains a single DiffServ code point or value. EF minimizes delay and jitter, and provides the highest level of aggregate quality of service. Any traffic that exceeds the traffic profile defined by local policy is discarded.

- The second class is Assured Forwarding (AF), which contains four classes and three drop-precedence's within each class making a total of twelve code points possible. Excess AF traffic is not delivered with as high probability as the traffic within the profile. This causes the traffic to be demoted but not necessarily dropped.

DiffServ is a very simplistic approach and is therefore not that powerful, but if it makes use of RSVP parameters or specific application types to identify constant bit-rates (CBR) it may be possible to establish well defined aggregate flows that may be dedicated to fixed bandwidth pipes, making it possible to share resources efficiently while still providing QoS. In contrast to DiffServ, IntServ was proposed to reserve resources in a best effort network in advance so that selected flows can enjoy the privilege of being treated with more resources.

Subnet Bandwidth Management (SBM): Enables categorization and prioritization at Layer 2 (the data-link layer in the OSI model) on shared and switched IEEE 802 networks.

Multi Protocol Labeling Switching (MPLS): Provides bandwidth management for aggregates via network routing control according to labels in (encapsulating) packet headers. MPLS combines the best of both worlds - ATM's circuit switching and IP's packet routing. It is a hybrid technology that enables very fast forwarding in the core and conventional routing at the edges.

It is similar to DiffServ in some respects, as it also marks traffic at ingress boundaries in a network, and un-marks at egress points. But unlike DiffServ, which uses the marking to determine priority within a router, MPLS markings (20-bit labels) are primarily designed to determine the next router hop. MPLS is not application controlled (no MPLS API's exist), nor does it have an end-host protocol component. Unlike any of the other QoS protocols described in this paper, MPLS resides only on routers.

And MPLS is protocol-independent (i.e., "multi-protocol"), so it can be used with network protocols other than IP (like IPX, ATM, PPP or Frame-Relay) or directly over data-link layer as well [MPLS Framework, MPLS Architecture]. Packets are assigned a label at the entry to a MPLS domain, which is often the core backbone of a

provider, and are switched inside the domain by a simple label lookup. The labels determine the quality of service the packet gets.

The packets are stripped of the labels at the egress router and might be routed in the conventional fashion thereafter before it reaches its final destination. In conventional IP routing, the next hop for a packet is chosen by a router on the basis of the packet's header information and the result of running a network layer routing algorithm. Choosing the next hop at the routers is thus a composition of two functions:

- Partitioning the whole set of possible packets into Forwarding Equivalence Classes (FEC).
- Mapping each FEC to a next hop.

The mapping of packets to FEC's is done at every router where largest prefix match algorithms are used to classify packets into FEC's. In MPLS the assignment is done only once - at the entry of the packet in the MPLS domain (at the ingress router). The packets are assigned a fixed length value ("label") depending upon the FEC category to which it belongs and this value is sent along with the packet.

At later hops, routers and switches inside the MPLS domain do not have to use complex search algorithms, but simply use this label to index into their routing tables which gives the address of the next hop for the packet, and a new value for the label. This new label replaces the packet's label and the packet is forwarded to the next hop. This is exactly like switching. It is multi-protocol since its techniques can be used with any network layer protocol.

It must be remembered that in context packet switching is still a best effort service with no definite bound on delay and jitter for example. The disadvantages of the different methods for providing certain flows with better service than other can be classified as follows [5,6].

Type of Service	Disadvantages
IntServ	• Because IntServ dedicates resources in a network in advance it makes routers very complicated. This makes the need for intermediate routers to have RSVP support a reality. • RSVP imposes maintenance of soft states at the intermediate routers. This implies that routers have to constantly monitor and update states on a per flow basis. In addition the periodic messages sent add to the congestion in the network • Because RSVP is purely receiver based, the reservations are initiated by willing receivers. But in many cases, it is the sender who has the task of initiating a QoS based flow. Thus causing RSVP to fail the accommodation of such flows. • IntServ is not scalable with the number of flows. Making routing very difficult for increasing number of flows and therefore making the backbone routers slow. • There are also no negotiating and backtracking, but DiffServ was proposed for this task.
DiffServ	• DiffServ is sender orientated. • If an end-to-end QoS must be guaranteed, providing QoS on a per hop basis won't be good enough. • DiffServ cannot account for dynamic SLA's between the customer and the provider. It assumes a static SLA configuration. But in the real world network topologies change very fast. • Some long flows like high bandwidth videoconferencing require per-flow guarantees. But DiffServ only provides guarantees for the aggregates.

Table 2.5.1 Disadvantages of IntServ and DiffServ

2.5.1 Queuing mechanisms used for QoS guarantees on packet switched networks.

There are currently many queuing techniques that are used on routers placed within packet switching networks to handle an overflow of traffic and to determine which, packets to drop if such an event occurs. These queuing methods include FIFO, Priority Queuing, Custom Queuing, Weighted Fair Queuing, Random Early Detection and Weighted Random Early Detection for example.

2.5.1.1 FIFO (First in first out) queuing

Routers traditionally used a FIFO queuing approach at their output ports. The router would contain a single queue on its output port and when a new packet arrives from the input it would be added to the back of the queue. As long as the queue isn't empty the router transmits the packets in the order it arrived.

This type of queuing technique had several drawbacks, firstly no priority is given to packets with higher priorities. Secondly if short packets were caught behind a large packet their average delay will increase more than the average delay of the longer packet, meaning in short that flows of larger packets get better service. Thirdly, a greedy TCP connection can crowd out more altruistic connections, causing other connections along the same path to back off more, if the greedy TCP connection does not back off. This approach can be seen in figure 2.5.1 [12,13,14].

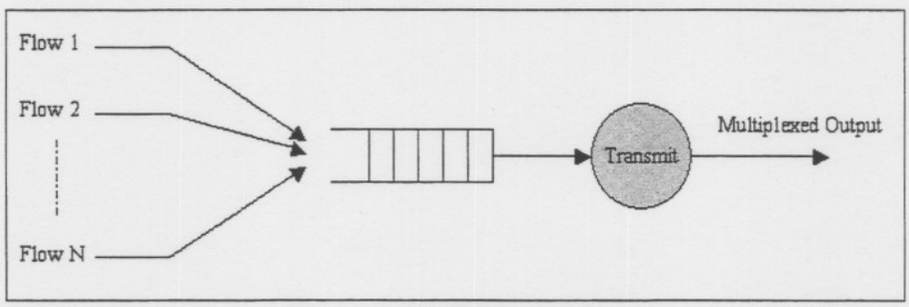

Figure 2.5.1 FIFO queuing

2.5.1.2 Priority queuing (PQ)

This protocol ensures that the traffic with the highest priority is processed first, and is therefore useful for time sensitive protocols. The packets can be prioritized according to many facets, including type, size and destination address for example. This approach contains four queues, high, medium, normal and low. The high queue is

processed before the medium queue and so forth. This type of queuing can be seen in figure 2.5.2 [5,6].

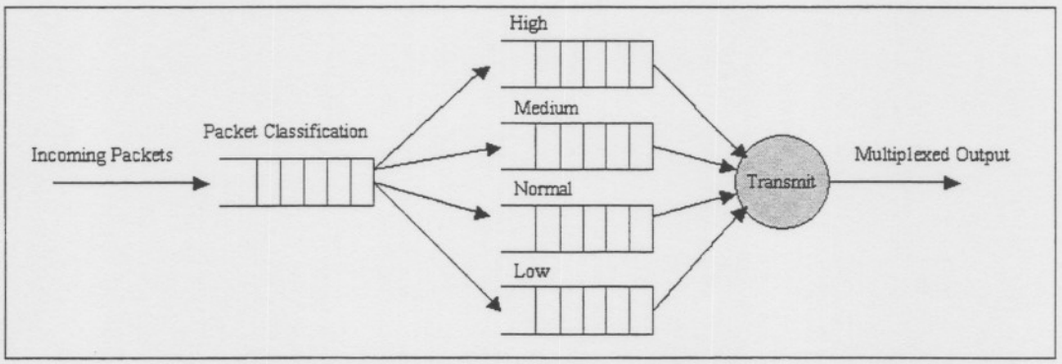

Figure 2.5.2 Priority queuing

2.5.1.3 Custom queuing (CQ)

In a custom queuing approach the network bandwidth is shared between different applications with different minimum latency and bandwidth requirements. To achieve this custom queuing assigns different amounts of queue space to different protocols and then handles them in a round robin fashion. Thus the protocol with the highest priority is assigned the biggest queue space, making custom queuing more useful to provide guaranteed bandwidth at potential congestion points.

The message is placed in one of 17 queues, and then processed in a round robin fashion. A network administrator controls the length of the queues for different protocols. After the appropriate amount of packets is transmitted for the queue, the next queue is processed. Figure 2.5.3 represent the custom queuing method, this method is also very attractive for applications that need very low amounts of jitter and delay [12,13,14].

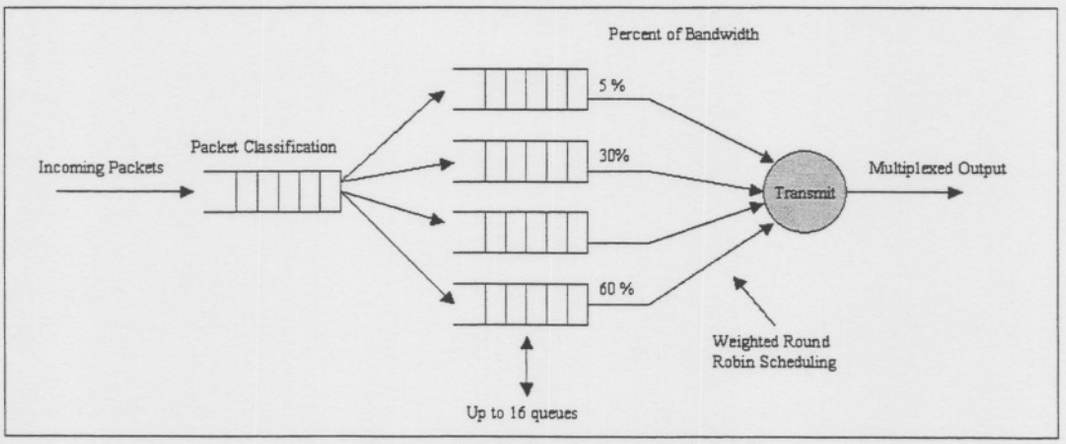

Figure 2.5.3 Custom queuing

2.5.1.4 Weighted fair queuing (WFQ)

WFQ is a sophisticated set of algorithms designed to reduce delay variability and provide predictable throughput and response time for traffic flows. A goal of WFQ is to offer uniform service to light and heavy network users alike. WFQ ensures that the response time for low-volume applications is consistent with the response time for high-volume applications. Applications that send small packets are not unfairly starved of bandwidth by applications that send large packets.

WFQ is a flow-based queuing algorithm that recognizes a flow for an interactive application and schedules that application's traffic to the front of the queue to reduce response time. Low-volume traffic streams for interactive applications, which include a large percentage of the applications on most networks, are allowed to transmit their entire offered loads in a timely fashion [12,13,14].

High-volume traffic streams share the remaining capacity. Unlike custom and priority queuing, WFQ adapts automatically to changing network traffic conditions and requires little to no configuration. It is the default queuing mode on most serial interfaces configured to run at or below 2.048 Mbps. This type of queuing can be seen in figure 2.5.4.

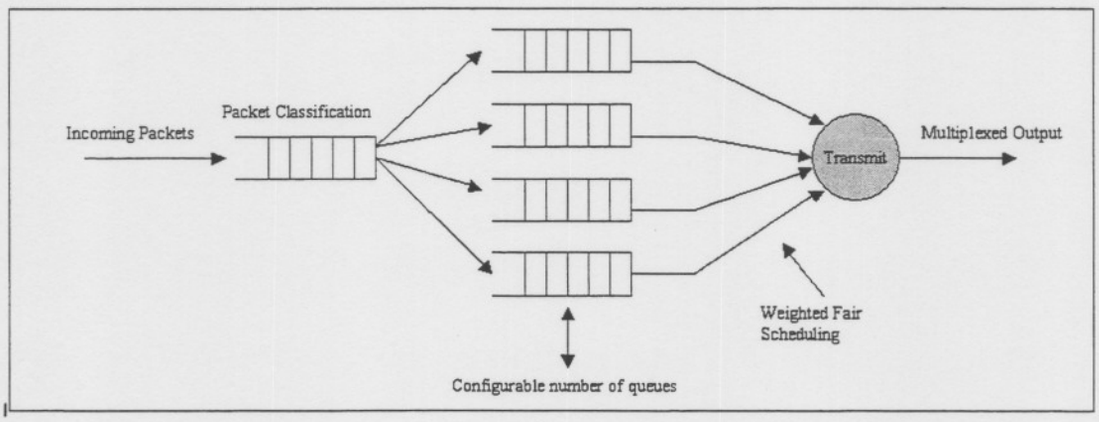

Figure 2.5.4 Weighted fair queuing

For applications that use the IP precedence field in the IP header, WFQ can allocate bandwidth based on precedence. The algorithm allocates more bandwidth to conversations with higher precedence, and makes sure those conversations get served more quickly when congestion occurs. WFQ assigns a weight to each flow,

which determines the transmit order for queued packets. IP precedence helps determine the weighting factor.

WFQ also works with RSVP. RSVP uses WFQ to allocate buffer space, schedule packets, and guarantee bandwidth based on resource reservations. Additionally, WFQ understands the discard eligibility bit (DE), and the forward explicit congestion notification (FECN) and backward explicit congestion notification (BECN) mechanisms in a Frame Relay network. Once congestion is identified, the weights used by WFQ are altered so that a conversation encountering congestion transmits less frequently.

WFQ is a critical component of DiffServ and defines the scheduling policies for the outgoing links bandwidth allocation for the different classes. The goal of WFQ is to provide a packet-based approximation of the generalized processor sharing. A generalized processor sharing scheduler is work conserving and can be regarded as the limiting form of a weighted round robin policy, where traffic from each session is treated as infinitely divisible fluid. If there are N classes the different sessions are characterized by positive weights ($w_1 - w_N$). The QoS of each class in terms of queuing-delay and packet loss depends on the assignment of these weights as well as the allocation of buffer sizes to all the classes [5,6].

2.5.1.5 Random early detection (RED)

The queuing theorems mentioned in the above sections, deals with congestion management and not with preventing congestion before it occurs. RED tries to prevent congestion before it occurs, by monitoring the network at different points and dropping packets if congestion begins to increase. The source nodes will detect the dropped traffic and in turn they will slow their transmission rate. If packets are dropped, the TCP window size will decrease and the transmission rate will decrease.

One of the advantages of RED is that it drops packets at random. Dropping packets at random reduces the potential for multiple sessions synchronizing their behavior. Because of these advantages that RED contains it is well suited for a central site router in for example a hub and spoke topology [12,13,14].

2.5.1.6 Weighted random early detection (WRED)

In this approach the standard capabilities of red are used in conjunction with for example IP precedence bits. This allows certain packets to have a higher preferential than other packets. Thus implies that packets with lower priorities will be discarded first when congestion increases. RSVP can also be used to adjust the priority settings of WRED. Another approach is to mark the packets with an in/out tag in at the edge of the domain. If congestion occurs, packets with the out tag will be dropped first at a lower congestion rate than packets with an in tag by the core routers.

2.6 Similar simulations and their results (Other QoS over IP simulations)

The area of provisioning QoS (Quality of Service) in an internet (best effort) environment has been researched and debated extensively in the past as well as the present. Methods used ranges from different routing methods, classifiers, scheduling, queuing, buffer management, admission control, shaping, policing and capacity planning for example.

The trend to provide a service environment where customers are billed according to the quality of their connection as well as on usage is obvious and drives network operators towards a service quality rather than connection time orientated infrastructure [25,27].

How do we measure QoS over an IP based infrastructure? To simulate this, workload simulation tools can be used. Workload simulation tools allow the network planner/analyst to proactively plan the performance and capacity of a network infrastructure and network applications.

When the performance or QoS of a session are considered, a session must first be characterized. Measures that can be used to quantify the performance or QoS of a session are but are not limited to message/packet arrival rate, variability of message/packet arrival rate, length of the session, expected message/packet length and length distribution, allowable delay, reliability and ordering of packets [26].

Extensive research into the IP QoS phenomenon has been performed. The following paragraphs represent some IP QoS approaches and their results. Firstly a method used to simulate packet loss in an IP network will be discussed, followed by a

method to simulate and analyze QoS parameters in an IP network using real time video streaming. This will be followed by a description of a method where active and passive monitoring was implemented with probes. Lastly, a stochastic modeling method for a TCP/IP session will be presented. The aim of these paragraphs is to give the reader a clearer view of IP QoS simulation techniques and approaches [25,26,27].

2.6.1 Simulation and analysis of packet loss in IP networks

In this simulation ns2 was used to simulate the loss of packets in the IP network. Tracing on a per packet level were performed during the simulation. The two main types of packet loss scenarios that were investigated were packet loss on routers due to queue overflow and packet loss on a per flow basis [27].

To investigate the packet loss due to router queue overflow a simple topology constructed from *n* traffic sources, one router and one destination (figure 2.6.1) were used. The router employs a FIFO drop tail queue management policy. The main aim of the simulation was to obtain insight into the loss patterns and the effect that transport protocols have on the loss behavior [27].

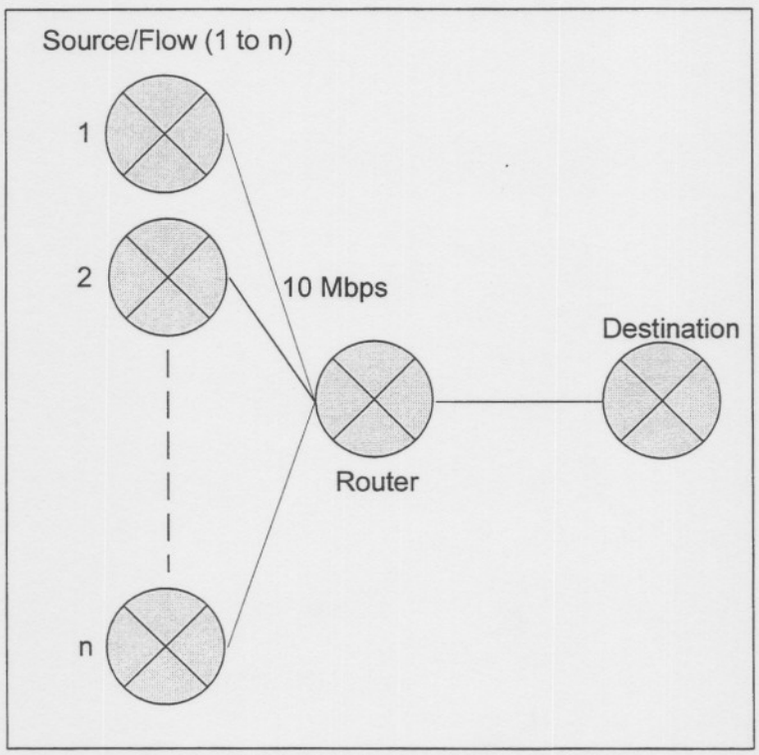

Figure 2.6.1 Simple simulation topology

In the first test only UDP sources was used to determine the effect on UDP packet loss based on queue overflow. In the second scenario only TCP sources was used to determine the effect on TCP packet loss due to queue overflow. In the last test TCP and UDP sources was used. To obtain a more realistic representation of packet loss, a complex network (figure 2.6.2) was used. Sources included TCP as well as UDP.

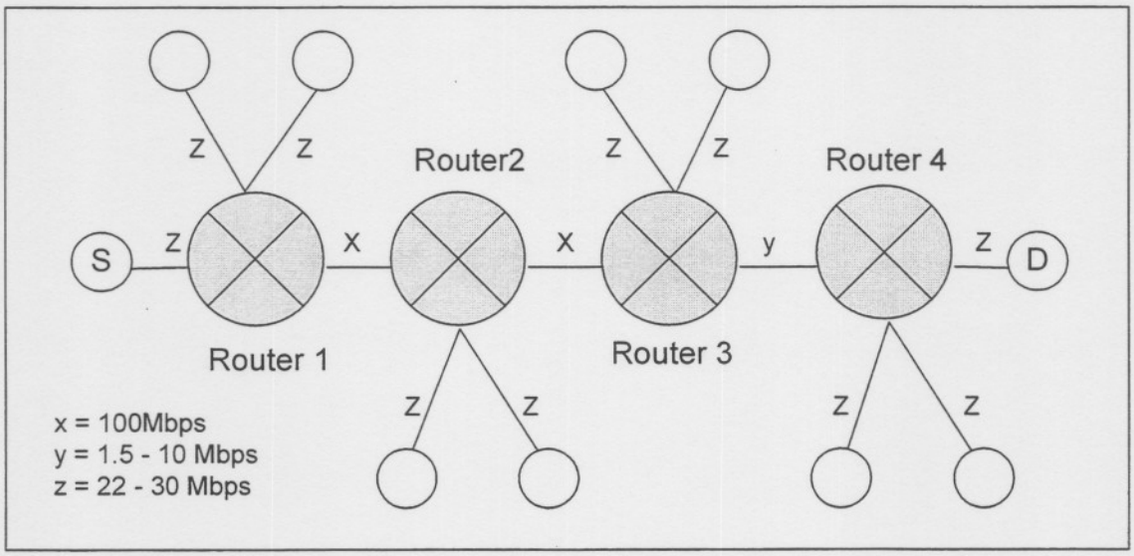

Figure 2.6.2 Complex simulation topology

The research showed that lost packets are often grouped in bursts or lost episodes. Although single losses can be handled, concealed, or corrected by the end user applications, multiple consecutive losses have adverse effects on the transfer of multimedia traffic [27].

The contribution of lengthier loss episodes for TCP sources is in general smaller than in the UDP case. The main reason for the different behavior of the TCP loss is the nature of the TCP sources, which adapt to the changes in the network conditions. The decrease of the congestion window of a particular TCP source upon a detection of lost packets directly reflects on the number of generated packets in one round-trip time. The maximum length of the loss episodes is short, and for the various buffer sizes and number of sources deployed in the TCP simulations, the loss episodes of length one (single losses) contribute to more than 90% of the total number of loss episodes [27].

The difference between the UDP and TCP loss behavior was even more pronounced in the scenario with mixed UDP/TCP sources, where it was showed that both the average loss rates and the lengths of loss episodes are smaller for the TCP senders.

While the_TCP senders are adaptive, the UDP senders generate their packets whenever they have packets to be sent, without considering the current network conditions. Although the simulations with the more complex topology involved richer traffic composition, the contribution of loss episodes showed a similar pattern to the simulations with simple topology [28,29].

2.6.2 Simulating and analyzing QoS parameters in IP networks using real time traffic.

In this simulation ns2 was used to simulate the transmission of video traffic and the analysis of the QoS parameters. The main QoS parameters that was investigated was delay and packet loss. The analysis focused on routers utilizing a FIFO (First in first out) buffer and a drop-tail queue management policy. The network topology that was used can be seen in figure 2.6.3.

The results showed that the packet delay distribution not only decays very slowly, but also exhibits a rapid increase pattern near the tail of the distribution. The autocorrelation function of packet delay exhibited some degree of periodicity and oscillation in the large lag scale and a linear decay pattern in the small lag scale. The research also showed that the delay jitter distribution has an exponential- like decay pattern [31].

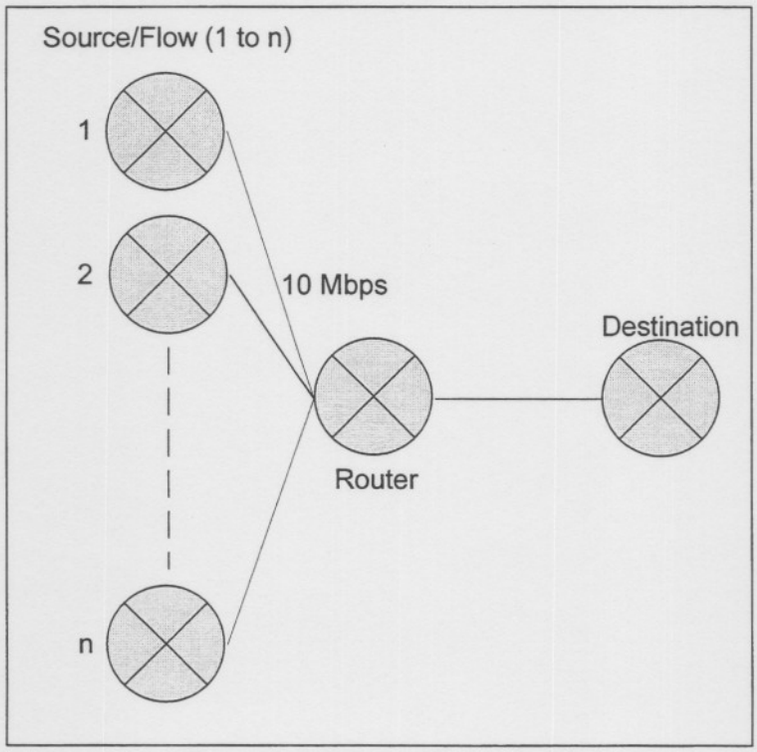

Figure 2.6.3 Simulation topology

2.6.3 Measuring IP QoS measurements in IP multicast networks using network probes (MQM (Multicast Quality Monitor))

In this approach RTP (Real Time Protocol) has been selected for the most QoS measurements. The only measurement that was not based on RTP was delay. MQM uses its own ping mechanism to measure both, the one way delay in each direction and the round trip delay. MQM uses RTP streams to measure the packet loss and jitter. This is achieved through either passive or active measuring [32].

Passive: Probes join an existing multicast session and decode the information in the received RTP packets.
Active: In this approach probes simulate typical sessions with as least as possible impact on the network.

The probe locations within the network for passive and active monitoring can be seen in figure 2.6.4 followed by the ping message, beacon message and header format.

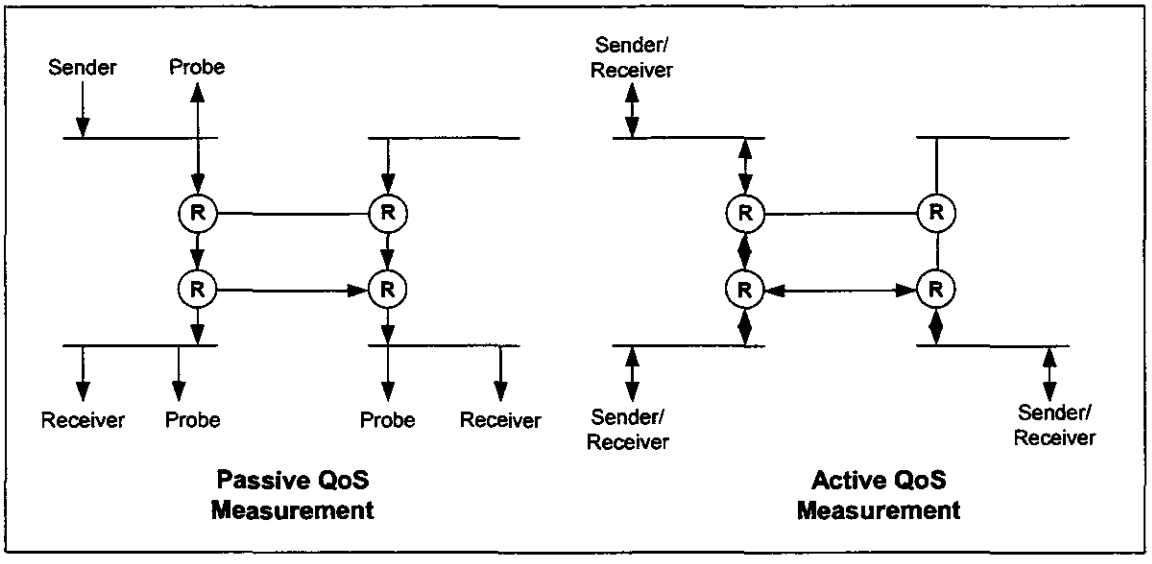

Figure 2.6.4 Passive and active QoS monitoring techniques

The message format that was used can be seen in table 2.6.1. to table 2.6.3.

Pos	Size	Content
0	1	Version
1	1	Padding
2	2	Type

Table 2.6.1 MQM header format

Pos	Size	Content
0	4	MQM Header
4	4	IP address from the originator
8	4	Timestamp sender
12	4	Timestamp receiver

Table 2.6.2 MQM Ping message format

Pos	Size	Content
0	4	MQM Header
4	2	Hold time
6	2	Message length
8	4	Target probe address 1
12	4	Multicast group address 1 (Is used to specify if to receive or to transmit to this group)
.....	4	Target probe address n
.....	4	Multicast group address n

Table 2.6.3 MQM Beacon message format

The MQM design distinguishes between the test of functionality (reliability) and the measurement of the QoS of an IP multicast network. Functionality testing is done without any involved central intelligence. With the MQM model, it is possible to check the function also behind the problem. Also, the measurement of the reliability of the MQM is done with only a very low impact on the network. Depending on the type of the IP multicast services, the additional congestion of the network may be very small if there are some broadcast services which can be used for the QoS measurement.

The primary idea of the MQM is to measure the QoS depending to the services (applications) in the network. The model is not restricted for a local use only. It is designed to support distributed services as well [32].

2.6.4 Stochastic modeling of a single TCP/IP session over a random loss channel

This researched focused on an analytical framework for modeling the performance of a single TCP session in the presence of random packet loss. This research showed that random loss severely affected the flow throughput. The results also showed that increasing the buffer size doesn't necessarily increase the throughput. If the buffer size is increased for low loss flows the throughput is increased, but if the buffer size is increased for flows with high packet loss there is no positive impact on the throughput [30].

2.6.5 Conclusion

Currently efforts such as integrated services, differentiated services, explicit congestion notification and MPLS for example are used to provide QoS in the internet. In many respects, simulating the Internet and QoS within the internet is fundamentally harder than simulation in other domains. In the Internet, due to scale, heterogeneity, and dynamics, it can be difficult to evaluate the results from a single simulation or set of simulations.

The above sections presents methods where simulations were used to determine specific QoS parameters in an IP network as well as methods to determine QoS in multicast IP networks and methods used to simulate probing of the network to determine the QoS parameters. The method used in this thesis will look at simulating IP multicasts as well as flows in an IP network and extracting TCP/IP header data to perform analysis of the QoS parameters of the specific flows or communication sessions.

2.7 Conclusion

Now that an overview of circuit switching and packet switching techniques and methods are finished a more detailed description of TCP/IP can be given. This chapter was intended to give the reader a broad overview of the circuit and packet switched technology before a more detailed overview of the TCP/IP protocols and their respective header fields are given. The detail is necessary to understand the processes involved in implementing and programming OSI levels and a TCP/IP protocol into a network simulation package.

Chapter 3. Theoretical investigation of TCP/IP header QoS data extraction

Abstract – This chapter will give the reader a detailed description of the IP and TCP protocol suites. Each field in the TCP and IP headers will be discussed in detail and their place within the networking infrastructure and their involvement in this study will be established. The method of extracting data from these relevant header fields will also established.

3.1 Introduction

In chapter 2 an overview of the TCP and IP headers and the fields within these headers were given. Each of these fields however deserves more detail. This chapter will give a detailed description of the IP and TCP protocol suites and the different fields and protocols in these suites. This chapter therefore contains a summary of various RFC's describing the different fields within the TCP/IP headers. A discussion concerning the type of data that can be measured from these headers are given. The relevant header fields that can be used to extract data from for TCP/IP QoS monitoring are also identified.

3.2 IP protocol suite

The IP protocol suite is a suite constructed of many different protocols with firstly the network layer protocols, which is given an ether-type number. Secondly transport layer protocols, which is given an IP protocol number and lastly the application layer protocols which are assigned one or more SCTP, TCP or UDP numbers. Examples of the different protocols for the different layers can be seen in table 3.2.1.

Network layer protocols	Transport layer protocols	Application layer protocols
• **ARP** (Address resolution protocol) • **RARP** (Reverse Address resolution protocol) • **DRARP** (Dynamic RARP) • **InARP** (Inverse address resolution protocol) • **IP** (Internet	• **AH** (IP authentication header) • **AX.25** • **CBT** (Core based trees) • **EGP** (Exterior gateway protocol) • **ESP** (encapsulating security payload) • **GGP** (Gateway to gateway protocol) • **GRE** (Generic routing encapsulating)	• **ACAP** (Application configuration access protocol) • **Agent X** • **APEX** (Application exchange core) • **ATMP** (Ascend tunnel management protocol) • **AURP** (Apple talk update based routing protocol)

protocol) • **IPv6** • **TP/IX** (The next internet)	• **HMP** (Host monitoring protocol) • **ICMP** (Internet control monitoring protocol) • **ICMPv6** (ICMP for IPv6) • **IDPR** (Inter domain policy routing protocol) • **IFMP** (Ipsilon flow management protocol) • **IGAP** (IGMP for the authentication protocol) • **IGMP** (Internet group management protocol) • **IP-in-IP** encapsulation • **IPPCP** (IP payload compression protocol) • **IRTP** (Internet reliable transaction protocol)	• Authentication server protocol • **BFTP** (Background file transfer protocol) • **BGP** (Border gateway protocol) • **COPS** (Common open policy service) • **CRANE** (Common reliable accounting for network element) • **DHCP** (Dynamic host configuration protocol) • **DICT** (Dictionary server protocol) • **Discard** (Discard protocol) • **EMSD** (Efficient mail submission and delivery)

Table 3.2.1 Examples of different layer protocols

These protocols coincide to produce the internetworking infrastructure of the internet, with the application layer programs typically e-mail or telnet for example and the lower layers only performing the necessary data link and link control characteristics of the transmission session [RFC 1180].

3.3 IP in detail

Figure 3.3.1 represents an IP header and the fields it contains followed by sections explaining each field in detail. Each field in the header contains specific data, which is used for connection establishment and termination. Some of these fields also contain the priority of the packet, which is used for routing purposes by the network routers. The next sections will give more information about the different header fields found in the IP header and how they could be used in determining the QoS parameters of a session.

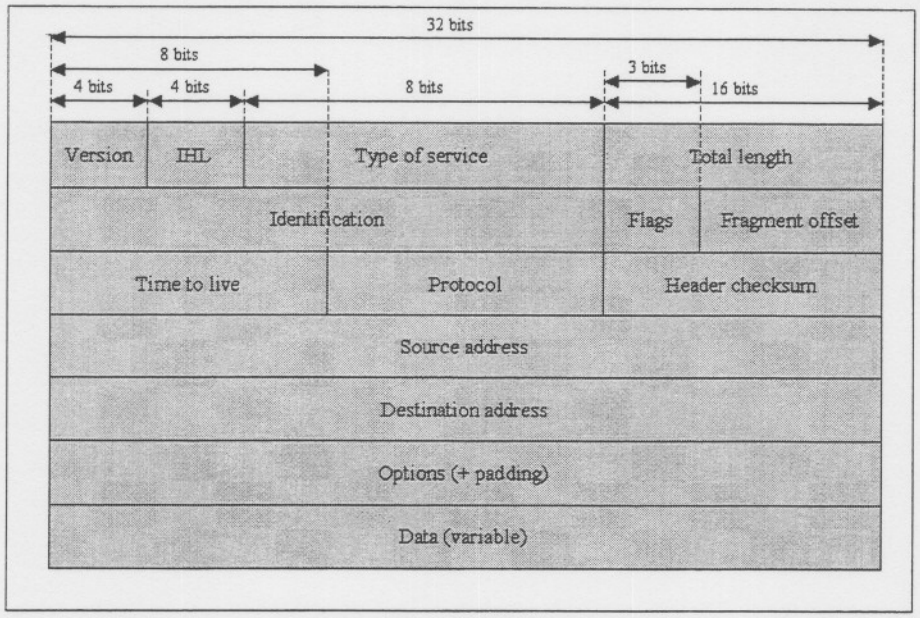

Figure 3.3.1 IP header

3.3.1 Version field

The version field indicates the current version of IP that is used and therefore specifies the format of the IP header. The Version field is 4 bits in length and contains sixteen values, these values and their meaning can be seen in table 3.3.2.

Version	Description
0	Reserved
1,2,3	-
4	IP (Internet protocol)
5	ST, St datagram mode
6	SIP, Simple Internet Protocol. SIPP, Simple Internet Protocol Plus. IPv6
7	TP/IX (The next internet)
8	PIP, The P internet protocol
9	TUBA
10,13,14	-
15	Reserved

Table 3.3.2 IP header version field contents

3.3.2 IHL (Internet header length)

The IHL field is 4 bits in total length and contains the length of the IP header in 32 bit words with the minimum valid amount of words equal to 5. Five 32-bit words will be long enough to contain all the header information except the options and data fields. The IHL field in conjunction with the IP header length is used to establish the payload length within the packet.

3.3.3. Type of service field

The type of service field is eight bits in total length and specifies the parameters of the type of service requested by the packet. This field is also at occasions utilized by the networks to define the handling of the datagrams during transport. The eight bits of this field contains five parameters, three bits of the eight are for precedence, one bit is for delay, one bit is for throughput, one bit is for reliability and the last bit is for monetary cost. The precedence field can contain a value between zero and seven depicting the priority of the packet [RFC 1349]. The values and their respective priorities can be seen in table 3.3.2 and 3.3.3. This field is used for switching purposes, therefore datagrams with higher priorities will be switched before datagrams with lower priorities.

Value	Priority
0	Routine
1	Priority
2	Immediate
3	Flash
4	Flash override
5	Critic/ECP
6	Inter-network control
7	Network control

Table 3.3.2 Precedence field values and corresponding priorities

The delay, throughput, reliability and monetary cost field values can be seen in table 3.3.3.

Value	Field BIT			
	Delay	Throughput	Reliability	Monetary cost
0	Normal delay	Normal throughput	Normal reliability	Normal cost
1	Low delay	High throughput	High reliability	Minimize cost

Table 3.3.3 Delay, throughput, reliability and monetary field values

3.3.4 Total length, identification, flag, fragment offset and time to live field

The total length field in the IP header are 16-bits in length and contains the length of the datagram. The next field in the IP header is the identification field and are 16-bits in length and are used to identify the fragments of one datagram from those of another. The originating protocol module of an Internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the network. The originating

protocol module of a complete datagram sets the MF bit to zero and the Fragment Offset field to zero. The next field in the IP header is the flag field and is constructed of three bits. The first bit is a reserved bit and the second bit is a do not fragment bit which controls the fragmentation of the datagram. The last bit is a more fragment bit, which indicates if the datagram contains additional fragments. The values of these different bits can be seen in table 3.3.4.

Value	Field bit		
	Reserved	**Do not fragment bit**	**More fragment bit**
0	Should be set to zero	Fragment if necessary	This is the last fragment
1		Do not fragment	More fragments follow this fragment

Table 3.3.4 Flag field bits description

The next IP header field is the fragment offset field, which is 13 bits in length and is used for the reassembly of the fragmented datagram. The time to live field is eight bits in length and contains a value that is used to track the lifetime of the datagram. If this field is decremented down to zero the packet is discarded. The total length field is used in conjunction with IHL and the TCP header length to calculate the payload of the data contained within the packet. These fields will therefore be used and extracted at the network entities for analysis.

3.3.5 Protocol field

The protocol field specifies the next encapsulated protocol and is eight bits in length. This field can contain 256 different encapsulating values. The different values and the protocols they encapsulate can be seen in table 3.3.5.

Value	Protocol	Value	Protocol
0	IPv6 hop-by-hop option	61	Any host internal protocol
1	**ICMP** (Internet control message protocol)	62	**CFTP**
2	**IGAP, IGMP, RGMP**	63	Any local network
3	**GGP**	64	**SATNET** and backroom **EXPAK**
4	IP in IP encapsulation	65	Kryptolan
5	**ST**	66	MIT remote virtual disk control protocol
6	**TCP**	67	Internet pluribus packet core
7	**UCL, CBT**	68	Any distributed file system
8	**EGP**	69	**SATNET** monitoring
9	**IGRP**	70	**VISA** protocol

10	BBNRCC Monitoring	71	Internet packet core utility
11	**NVP**	72	Computer protocol network architecture
12	**PUP**	73	Computer protocol heart beat
13	**ARGUS**	74	Wang span network
14	**EMCON**	75	Packet video protocol
15	**XNET**, Cross debugger	76	Backroom **SATNET** monitoring
16	Chaos	77	**SUN ND** PROTOCOL - Temporary
17	**UDP**	94	IP with IP encapsulation protocol
18	**TMux**	95	Mobile internetworking protocol
19	**DCN** measurement sub-systems	96	Semaphore communications Sec. Pro.
20	**HMP** (Host monitoring protocol)	97	Ether IP
21	Packet radio measurement	98	Encapsulation header
22	**XEROX NS IDP**	99	Any private encryption scheme
23	Trunk-1	100	**GMTP**
24	Trunk-2	101	**IFMP** (Ipsilon flow measurement protocol) IGMP for user Authentication Protocol
25	Leaf-1	102	**PNNI** over IP
26	Leaf-2	103	**PIM**
27	**RDP**	104	**ARIS**
28	**IRTP** (Internet reliable transaction protocol)	105	**SCPS**
29	**ISO** transport protocol 4	106	**QNX**
30	**NETBLT**	107	Active networks
31	**MFE** network services protocol	108	**IPPCP**, IP payload compression protocol
32	**MERIT** internodal protocol	109	**SNP**, Sitara networks protocol
33	Sequential exchange protocol	110	Compaq peer protocol
34	Third party connect protocol	111	IPX in IP
35	**IDPR**, Inter domain policy routing protocol	112	**VRRP**
36	**XTP**	113	**PGM**
37	Datagram delivery protocol	114	Any 0-hop protocol
38	**IDPR**, Control message transport protocol	115	**L2TP**
39	**TP++** Transport protocol	116	**DDX**, D-II Data exchange
40	**IL** transport protocol	117	**IATP** (Interactive agent transfer protocol)
41	IPv6 over IPv4	118	**ST**, schedule transfer
42	**SDRP**	119	**SRP**, Spectra link radio protocol
43	IPv6 routing header	120	**UTI**
44	IPv6 fragment header	121	**SMP**
45	**IDRP** (Inter domain routing protocol)	122	**SM**
46	**RSVP**	123	**PTP**
47	**GRE**	124	**ISIS** over IPv4
48	**MHRP**	125	**FIRE**
49	BNA	126	CRTP, Combat radio transport protocol

50	ESP	127	CRUDP, Combat radio user datagram
51	AH	128	SSCOPMCE
52	Integrated net layer security TUBA	129	IPLT
53	IP with encryption	130	SPS, Secure packet shield
54	NARP, NBMA address resolution protocol	131	PIPE, private encapsulation within IP
55	Minimal encapsulation protocol	132	SCTP, Stream control transmission protocol
56	TLSP using kryptonet key management	133	Fiber channel
57	SKIP	134	RSVP-E2E-IGNORE
58	ICMPv6, MLD	135 – 254	-
59	IPv6 no text header	255	Reserved
60	Destination options for IPv6		

Table 3.3.5 Protocol field values and their respective protocols

A detailed discussion the mentioned protocols within the IP protocol header field can be found in appendix B of this document.

3.3.6 Header checksum, source IP address and destination address fields

The IP header checksum field contains a 16-bit ones compliment checksum of the IP header and the IP options fields. The source IP address field contains the origin IP address of the packet and the destination IP address field contains the destination address of the packet. The destination and source addresses will be extracted by the IP probe at the network entities and will be used to determine the sessions' QoS parameters in both directions.

3.3.7 Options field

The options field can vary in length, it has a 1-bit copy field followed by a 4- bit class field and a 5-bit option field. The copy field indicates if the option should be copied into all the fragments. If the value is zero the option should not be copied into all the fragments, and if the value is one the option should be copied into all the fragments. The class field values and their descriptions can be seen in table 3.3.6 The options field is 5 bits in length and contains the option that can be copied or not into each fragment. The different values and their meanings can be seen in table 3.3.7.

Value	Description
0	Control
1	Reserved
2	Debugging and measurement
3	Reserved

Table 3.3.6 Class field values and their respective meanings

Option	Copy	Class	Value	Length	Option description
0	0	0	0	1	End of options list
1	0	0	1	1	NOP
2	1	0	130	11	Security
3	1	0	131	Variable	Loose source route
4	0	2	68	Variable	Time stamp
5	1	0	133	3.... 31	Extended security
6	1	0	134	-	Commercial security
7	0	0	7	Variable	Record route
8	1	0	136	4	Stream identifier
9	1	0	137	Variable	Strict source route
10	0	0	10	-	Experimental measurement
11	0	0	11	4	MTU probe
12	0	0	12	4	MTU reply
13	1	2	205	-	Experimental flow control
14	1	0	142	-	Experimental access control
15	0	0	15	-	
16	1	0	144	-	IMI traffic descriptor
17	1	0	145	-	Extended Internet proto
18	0	2	82	12	Traceroute
19	1	0	147	10	Address extension
20	1	0	148	4	Router alert
21	1	0	149	6.... 38	Selective directed broadcast mode
22	1	0	150	-	NSAP addresses
23	1	0	151	-	Dynamic packet state
24	1	0	152	-	Upstream multicast packet
25-31	-	-	-	-	-

Table 3.3.7 5-bit option field values and descriptions

3.3.8. Padding

The padding field can vary in length and is used as a filler to guarantee that the data starts on a 32-bit boundary. The next part of this chapter will investigate the type of data, which can be extracted from the TCP and IP headers and what kind of network statistic results can be compiled form the IP and TCP headers.

3.4 TCP in detail

TCP is normally implemented in cases where the applications require guaranteed delivery. It uses a sliding window approach, which handles both time outs as well as retransmissions. A TCP header can be seen in figure 3.4.1.

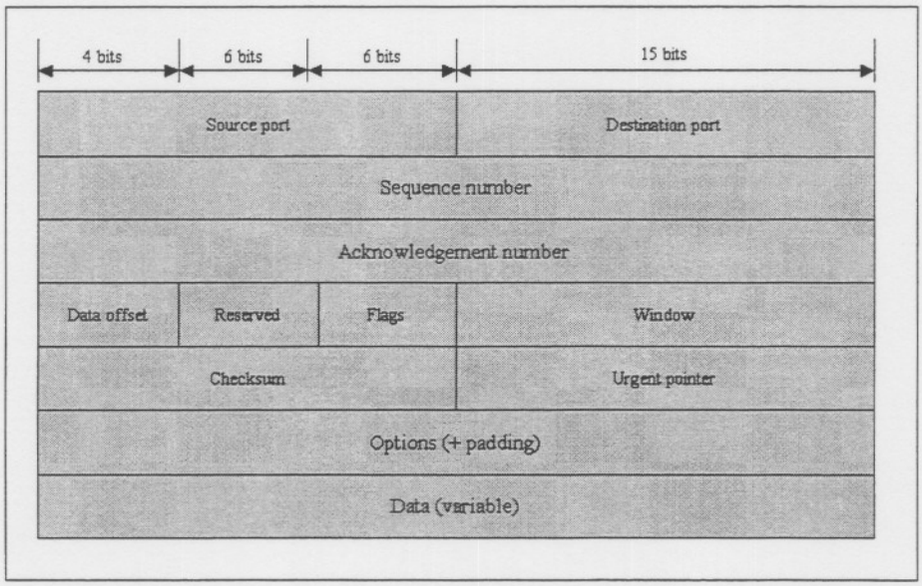

Figure 3.4.1 TCP header

3.4.1 Source and destination port, sequence number, acknowledgement number, data offset and reserved fields

The source port and destination port are both 16-bits in total length and contain the transmitting and destination stations port addresses. The sequence number field is 32-bits in length and is the first data byte in this segment. If the SYN bit is set, the sequence number is the initial sequence number and the first data byte is initial sequence number + 1. Once a connection is established with the use of the SYN bit, the acknowledgement number field contains the number of the next segment that must be transmitted.

The data offset field holds the amount of 32-bit words in the TCP header, which is used to determine the beginning of the data within the packet. The reserved field is 4-bits in total length and is set to zero. The source and destination port fields will be extracted by the probes at the different network entities and will be used in determining the sessions QoS parameters. Along with the destination and source

fields the data offset field will also be extracted and used in determining the length of the encapsulated payload.

3.4.2 ECN (Explicit notification congestion) and flag fields

TCP determines the appropriate congestion window to use by gradually increasing the window size until a packet is dropped. This causes routing queues to build up and can cause the dropping of latency sensitive flows data packets. Active queue management detects congestion before queue overflow and provides an identification of this congestion to the end nodes. Active queue management approaches use several methods to indicate congestion to the end nodes. These methods may include dropping packets for example.

But active queue management may also use the congestion experienced (CE) code-point fields in the TCP header minimizing the drop of latency sensitive data packets. This is achieved through creating two bits in the TCP header, contributing to four ECN points. The ECN capable transport code-points (10 & 01) are set by the data transmission entity to indicate that the end points of the transport protocol are ECN capable, calling them ECT(0) and ECT(1) respectively. The values and their descriptions for this field can be seen in table 3.4.1 [RFC 3168].

ECT	CE	Description
0	0	Packet not using ECN
0	1	ECT(1)
1	0	ECT(0)
1	1	Used by the router to indicate congestion to the end nodes

Table 3.4.1 TCP ECN field contents

The flag field is 6-bits in length and is used for control. The following bits are found in the TCP flag field.

Field	Description
URG	Urgent pointer valid flag
ACK	Acknowledgement number valid flag
PSH	Push function flag
RST	Reset the connection flag
SYN	Synchronize the sequence numbers
FIN	End of data flag

Table 3.4.2 TCP header flag field information

The flag field will be extracted from the TCP header at by the IP probes at the different network entities and will be used to determine the sessions QoS parameters.

3.4.3 Window and checksum fields

The window field is a 16-bit unsigned field and holds the number of data bytes beginning with the one indicated in the acknowledgement field, which the sender of this segment is willing to accept. The checksum field is 16-bits in length and holds the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. The pseudo header construction can be viewed in figure 3.4.2.

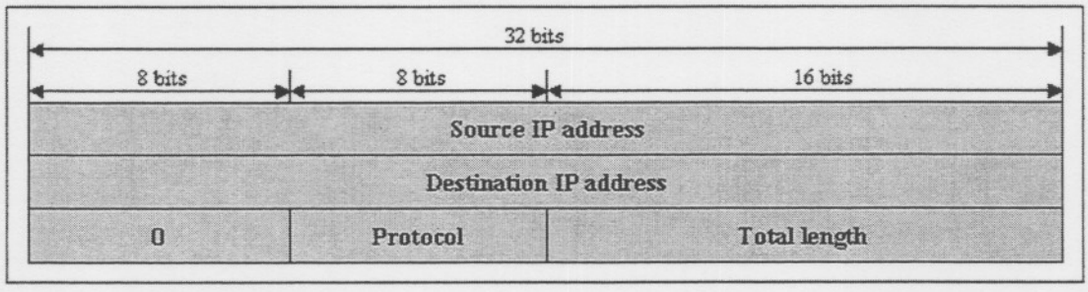

Figure 3.4.2 Pseudo header format

3.4.4 Urgent pointer and the options field

The urgent pointer is a 16-bit unsigned field and points to the sequence number of the last byte in a sequence of urgent data. The values and their descriptions used in the options field, which can be 0 – 44 bytes in length can be seen in table 3.4.3.

Kind	Length	Description	References
0	1	End of option list	[RFC 793]
1	1	No operation	[RFC 793]
2	4	Maximum segment size	[RFC 793]
3	3	Window scale factor	[RFC 1072, 1323]
4	2	SACK permitted	[RFC 2018]
5	Variable	SACK	[RFC 2018, 2883]
6	6	Echo	[RFC 1072]
7	6	Echo reply	[RFC 1072]
8	10	Timestamp	[RFC1323]

9_	2	Partial order connection permitted	[RFC1693]
10	3	Partial order service profile	[RFC1693]
11	6	CC, Connection count	[RFC1644]
12	6	CC.NEW	[RFC1644]
13	6	CC.ECHO	[RFC1644]
14	3	TCP alternate checksum request	[RFC1146]
15	Variable	TCP alternate checksum data	[RFC1146]
16		Skeeter	
17		Bubba	
18	3	Trailer checksum option	
19	18	MD5 signature	[RFC 2385]
20		SCPS capabilities	
21		Selective negative acknowledgements	
22		Record boundaries	
23		Corruption experienced	
24		SNAP	
25			
26		TCP compression filter	

Table 3.4.3 TCP options field content

The next section will discuss the relevancy of extracting header data and monitoring QoS parameters for sessions within packet switched networks as well as the method and the fields used in the TCP and IP headers to monitor these sessions' QoS parameters.

3.5 TCP and IP header QoS extractable data

The IP phenomenon has expanded at an unprecedented rate into a converged network topology where IP based video and voice communications are implemented. Implementing these services will help companies to step away from separate circuit switched telephone networks as well as separate video ISDN networks. Creating one network infrastructure, which is able to traverse data, voice and video. QoS in these converged networks is therefore an important implementation consideration.

Packet switching was originally designed as a best effort service, not distinguishing between different service needs for different types of service streams. The current networks with balanced network overhead have achieved acceptable levels of performance for most network applications. But if voice and video must be transmitted over these networks, new service parameters must be achievable to effectively transverse these types of services over the network.

3.5.2 IP network QoS parameters (Latency, end-to-end delay, packet loss and jitter)

Before measuring QoS for certain CoS we need to identify the information we need to determine QoS for these CoS. The information needed can be subdivided into the following parameters.

- Bandwidth.
- End-to-end delay.
- Jitter.
- Latency.
- Packet loss.

Bandwidth can typically be defined as the average amount of bits per second that can travel successfully through the network and is specified in kilo or mega bits per second. ***End-to-end*** delay can however be defined as the average time it takes for a packet to traverse through the network from one end point to another. With the use of end-to-end delay we can define ***jitter*** as the variation in the end-to-end delay of sequential packets. ***Packet loss*** is almost self-explanatory and is measured in percentage of packets that doesn't reach the end destination. ***Latency*** can be defined as the end-to-end delay as well as the endpoint processing time.

To maintain a good service quality, bandwidth is normally maximized while end-to-end delay, jitter and packet loss are minimized. If the delay is high there will be long pauses in the speech causing an unnatural feel to the conversation. Large jitter values will however cause packets to arrive out of sequence at their destination and will cause the video to be jittery and the audio to stutter.

QoS defines the ability to compensate for traffic characteristics without compromising the average network throughput. Network elements are therefore able to distinguish between different types of streams and treat them in an according manner. To achieve these end-to-end QoS specifications, the IETF (Internet Engineering Task Force) has created four areas for IP technologies.

- Prioritizing packets using differentiated services.
- Reservation using integrated services.

- Label switching using multi-protocol label switching.
- Bandwidth management with the use of subnet bandwidth managers.

Another approach in network infrastructures today is to implement networks, which are over engineered and therefore contain more bandwidth than is needed. This approach however is expensive and not viable in today's competitive telecommunication industry.

As a general rule of thumb when a network is designed, the maximum amount of bandwidth required for all applications including data, voice and video transfer should not exceed 75% of the available network bandwidth. Another approach to implement QoS for certain services is with the use of queuing techniques as discussed in section 2.7 of this document.

Another approach that is used in conjunction with queuing is classifying. The queuing mechanisms work on packet classifications, which can be found in the packet overhead. These services include RSVP, DiffServ and MPLS as described in section 2.7 of this document.

RSVP however is cumbersome because every device along the data flow path must be able to signal the RSVP specified QoS requirements making RSVP difficult to scale for large network topologies. In the IP header bits 9,10 and 11 are defined as the priority bits and can classify eight different types of priority which ranges from zero the lowest to 7 the highest. DiffServ however uses bits 9 to 16 in the IP header for packet prioritization. Making DiffServ more flexible with 64 different types of services.

MPLS uses a different approach for guaranteeing QoS for certain CoS. MPLS assigns a routing label to each packet based upon the packets priority and final destination. MPLS brings a number of benefits to IP based networks, which include RSVP types of QoS for example.

But is the necessary QoS parameters needed to conduct billing and service guarantees for certain CoS available from only the TCP and IP headers alone. The next section will identify the fields in the TCP and IP headers, which can be used to extract data for QoS monitoring and billing.

3.5.3 Data available form TCP/IP headers for QoS monitoring and billing

In the previous sections the TCP and IP headers were discussed in detail. Looking at the different fields in these headers. From this investigation fields can be identified which gives us ample data to monitor and bill for QoS provided for certain CoS. These fields can be extracted at the network entities from the TCP and IP headers. We can compare TCP/IP with the lower four layers of the OSI model. IP provides the routing of the packets through the network and TCP provides the management of the packets at the different hosts.

Examining the IP header, it is clear that some of the fields are irrelevant when QoS data are extracted. The fields of interest in the IP headers as mentioned at the end of each previous section are the destination IP address, source IP addresses, type of service field, IHL field and the total length field. In the IP header bits 9,10 and 11 are used for priority but if DiffServ is used bits 9 -16 are used. If the options field is investigated the next encapsulating protocol can be identified and recorded as well.

The throughput can be measured through recording the destination and host IP addresses and ports with their start and stop times and the number of bytes transmitted in every direction. This can help to calculate a throughput measure in Megabits/Second. This type of measurement however isn't that accurate because it doesn't include user time, it is therefore well defined for services such as ftp transfer but not for services where users need to make decisions such as telnet for example. Therefore performance problems are transient and normally occur at the end systems or at the access level but rarely on the backbone itself.

If we use the type of service field we can measure these throughput characteristics for the type of service used in this communication session to ensure that they are given the necessary bandwidth, minimum delay, jitter and latency that were agreed upon in the SLA with the user. It must be remembered that the probe that is going to be installed will be installed on network routers and not on end systems itself but on the routers within the network. Figure 3.5.1 represents the extraction of the relevant QoS parameters.

Therefore only the lower layers of the OSI stack will be visible to the probe to take its measurements from. If we look at a situation where a connection is established and

the transmission and receiving station transmits a request and an acknowledgement message the router will pick it up.

The router therefore listened to the whole communication setup stage and can now with the use of the destination and receiving IP addresses record the time of connection, type of connection from the type of service field, throughput for the connection by measuring start and stop times and the number of bytes transmitted. The router can also keep track of the amount of packets dropped due to corruption.

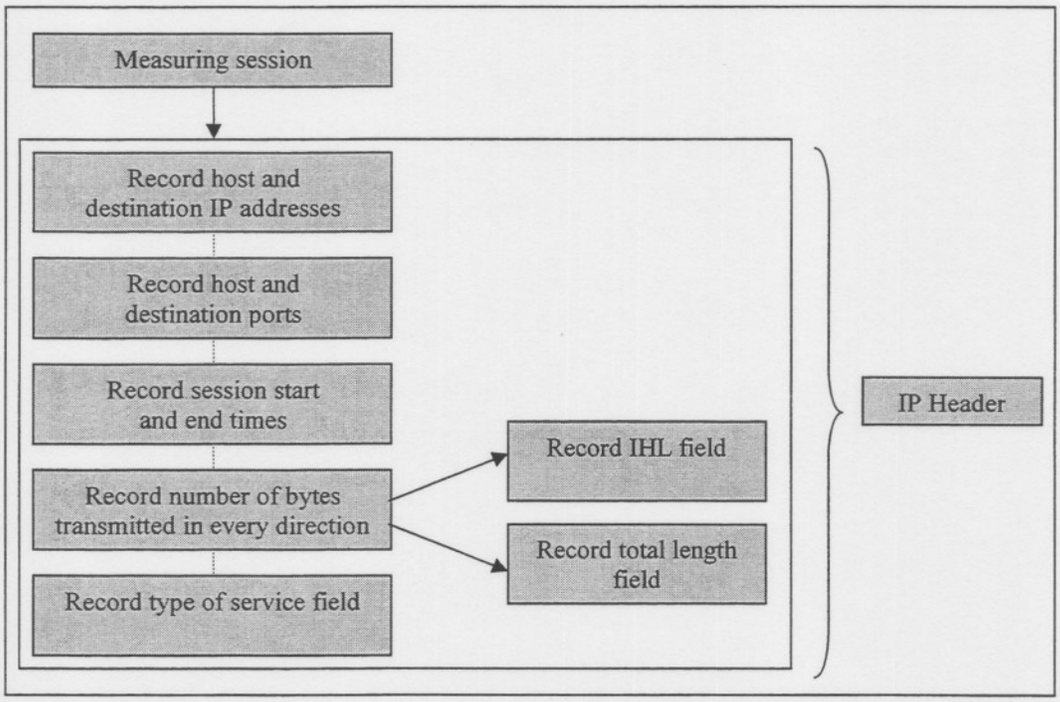

Figure 3.5.1 Extracting QoS fields from an IP header

As previously mentioned, the fields within the IP header that is of importance is the destination IP address, source IP addresses, type of service field, IHL field and the total length field and the protocol type field. Routers use IP headers for different reasons when they are stationed as firewalls between an organizations network and the Internet.

- Routers block out access by discarding all packets with IP destination addresses external to the origination.
- Routers use IP header information to block out access by discarding all packets with IP service addresses external to the organization.

If we examine TCP headers the fields of interest are the source port numbers, the destination port number, sequence and acknowledgement number fields, data offset field and the flag fields. Services are identified via port numbers and the flag fields are used to determine when the connection was started and when it was ended. The method used for extracting the data from the TCP header can be seen in figure 3.5.2.

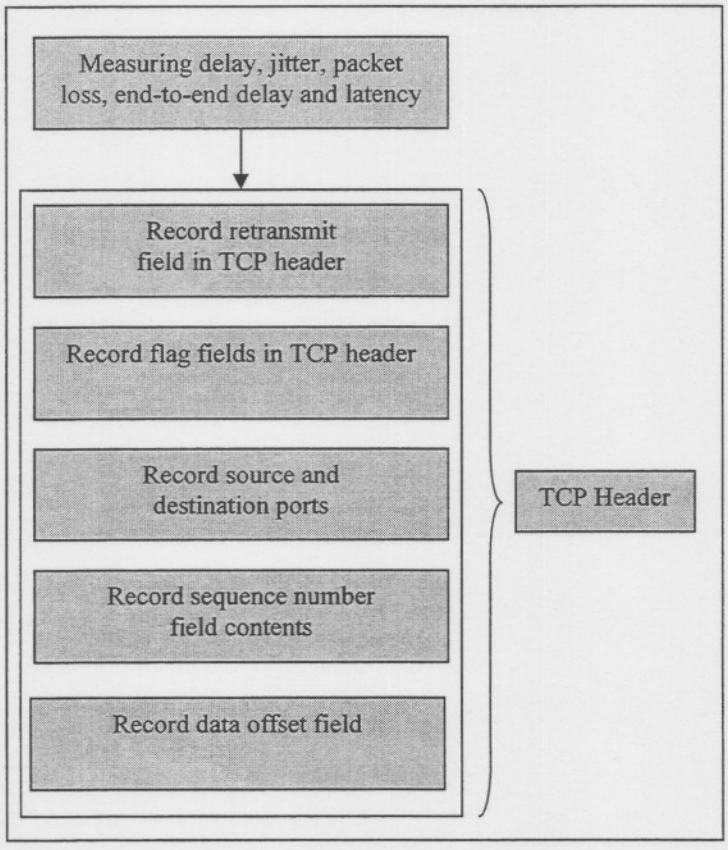

Figure 3.5.2 Extracting QoS parameters from a TCP header

These values are useful to monitor the type of service that certain flows received through the network, as well as to bill the customers based on these types of services. The information needed by the billing infrastructure are the host and destination IP addresses, the type of service, the amount of data transmitted, the time the communication session took place and if the type of service received its guaranteed on service levels. These parameters can be extracted from the IP and TCP headers. Figure 3.5.3 represents the overall extraction from the IP and TCP headers.

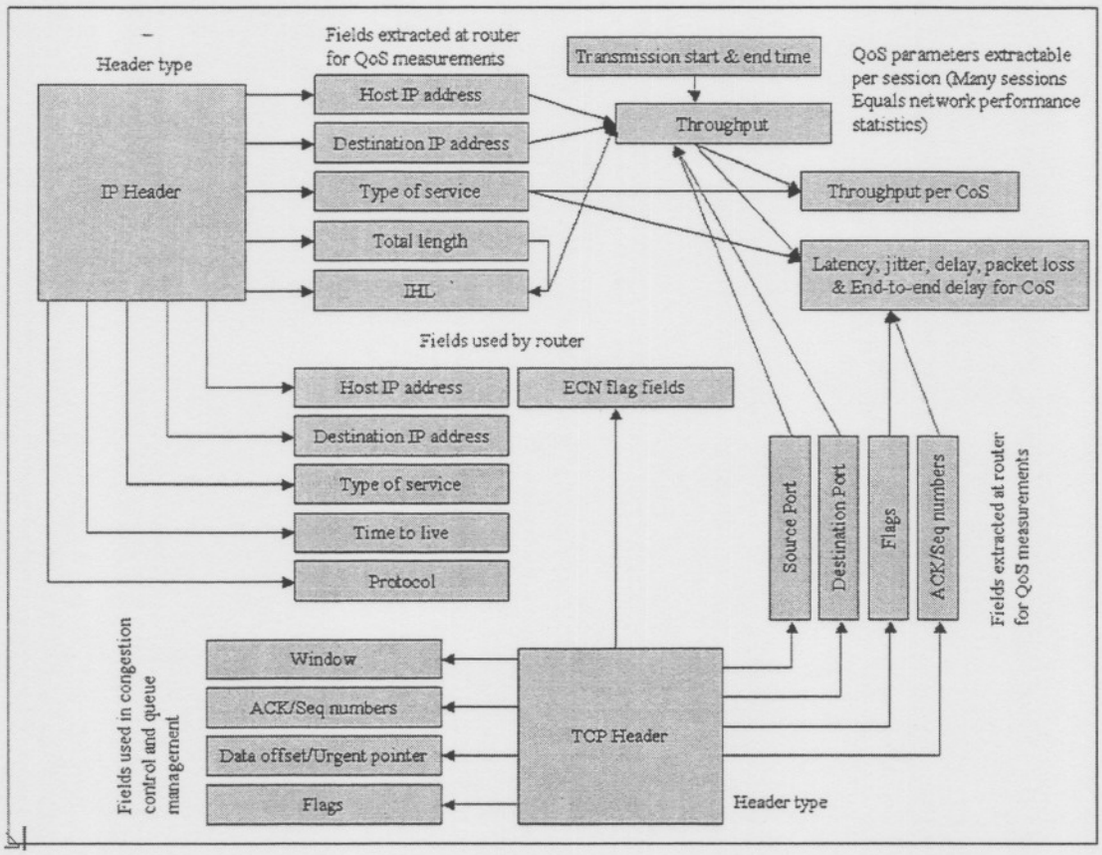

Figure 3.5.3 QoS parameter extraction from both TCP and IP headers

3.6 Conclusion

Satisfactory information about the service and the type of quality it is receiving can be extracted from the TCP and IP headers. Other services are encapsulated in the IP and TCP headers. At the routers only these exterior IP and TCP headers can be examined, therefore only the connection setup, service type, amount of packets dropped, end-to-end delay, amount of data transmitted, host and destination IP addresses and port numbers can be extracted.

If we examine purely QoS then layer three will be the layer to look at and if we purely examined CoS then layer two would be the layer to examine. In layer two separation between services occur and in layer three prioritization of these services are managed. In conclusion layer two therefore ensures local area network performance and layer three ensures wide area network performance. The optimal layer to extract header data at for QoS measurements if only TCP and IP headers are examined is the TCP header.

Chapter 4. Network simulation techniques and software

Abstract - The aim of this chapter is to give the reader an overview of the methods and techniques used in network simulation. Within this chapter different simulation packages will also be investigated and compared with each other as well as the different mathematical methods and approaches available to model systems and their relevant behavior. From these results the most appropriate simulation tool for the study will be chosen.

4.1 Introduction

Before modeling different systems, a system as well as different methodologies involved in modeling of systems needs to defined. There are different approaches that can be used when the modeling of a system is brought into consideration. In this chapter these different methodologies will be discussed in more detail as well as the different simulation packages that are available to model these systems.

4.2 System definition and modeling techniques

A **system** can be defined as a group or set of interacting and interrelated, or independent elements forming a complex whole. Such a system can be characterized by its parts and the interaction between these parts, and is in a whole defined by its purpose or function [23].

It must be remembered that a system can be constructed of multiple sub systems and these sub systems can also be constructed of different systems and so forth. The mode or the state that a system is in depends on the states of different sub systems. Thus a system can be recursive. These states are normally expressed by the values of different variables. These states can be discrete or continuous, with static systems not changing their values over time in contrast to a continuous system where the output is a function of time. If a system is chosen to be modeled what model should then be used for the system if it is a discrete or continuous system. Sometimes the physical system is available for testing, but in many cases this approach is too expensive and a model of the system must be created [23,24].

This model must then be created or built to capture some relevant properties of the system. Such a model is also simplified and reduced in complexity compared to the original system. Some models include physical models and mathematical models. Physical models are normally implemented in the aeronautical industry where a small

scaled down model of the original is built and used to perform tests on. For example a new airplane design must be tested for air profiles. A model of the airplane is created and then tested in a wind tunnel. In a mathematical approach the airplane would be modeled with the use of a series of mathematical equations concerning aerodynamics and different structure and material stress laws for example. This system is then used to predict how the real system would act, if it were subjected to the same criteria the model was subjected to.

Another approach is to use static or dynamic models, with the static approach only considering a fixed state of a system and a dynamic approach, which considers the state of the system as it evolves over time. In a continuous system the system is designed in such a manner that the state variables is a function of time and represents the system over time. In a discrete approach the variables only represent values occurring at distinct places in time [24].

Between these different times when the variables change, the variables maintain their previous values. Sometimes a continuous model isn't used for a continuous system and sometimes a discrete model isn't used for a discrete system. It all depends on the objectives that must be achieved as well as the intentions and the feasibility of the design. Another type of model is a deterministic model, in such a model the evolution of the states is completely described in such a manner that it only depends on the initial states. An example of such an instance is a set of differential equations describing the concentration of different substances in a chemical reaction.

A model where the evolution of state depends on random events in both time of occurrence or nature is called a stochastic model and can also be used to describe a system. An example of such a model of a system is a model of a highway in which the time that the cars enter the highway is described by a random variable. Thus the output of this model does not only depend on the initial state but also on the values of the random variables [23].

The same criteria exists for deterministic and stochastic models as for static and continuous models in the sense that sometimes a deterministic model is used to describe a stochastic system and the same for a stochastic system which is modeled by a deterministic model. When a system and its appropriate modeling technique

have been chosen the model can be created. But such a model must abide to some criteria to make the model a good model. These criteria include,

- An appropriate representation of the system
- The model must be as small as possible, without impeding the appropriateness of the model
- The model must be reusable for similar systems and as part of other systems.
- The system must be parameterizable.
- The system must be amendable to the investigation method, meaning that the model must work in an appropriate time with acceptable effort and with the desired accuracy [24].

Figure 4.1.1 represents a diagram distinguishing the different methods and their places within the modeling sequence.

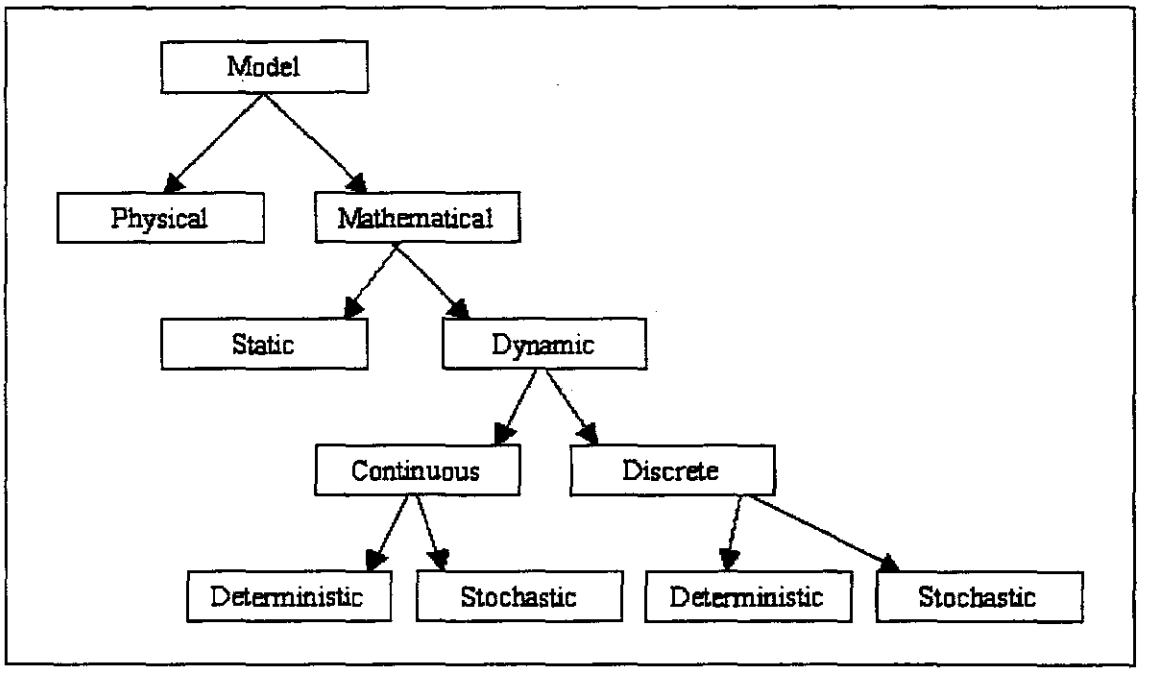

Figure 4.1.1 Model overview

Most systems have a set of parameters, which set and select the systems behavior. This set of parameters then usually serves as an input to a system evaluation. Such system evaluation methods include experiments, mathematical analysis and simulation methods. An experimental evaluation of the system normally implies that the physical system is available to do experiments on. These experiments normally include the observation of the system and the measuring of its performance. This

approach is often not feasible because of its expensive nature. Sometimes the system also doesn't yet exist for experimental evaluation. In a mathematical approach a mathematical model is required. This mathematical estimation or closed form solution of the system is normally derived from the characteristic parameters. If a closed form solution isn't obtainable then a formula can be created that can be evaluated numerically.

Sometimes this approach isn't feasible because of the complexity of the system [23]. In a simulation approach the mathematical model is needed and the operations of the given system is emulated with the use of the structure defined within the model. There are different ways to simulate dynamic, continuous models or dynamic, discrete models. Table 4.1.1 represents a short description of both the above-mentioned methods.

Dynamic, continuous models	**Dynamic discrete models**
Such a system is normally described by a set of differential equations. It also typically involves numerical solutions for these differential equations. There is no real difference to a numerically based mathematical solution.	Deterministic models • Implement the mechanisms of the model in a computer program. • Set the start parameters. • Let the program execute. • Observe the metrics that are of interest. • This approach is straightforward. Stochastic models • Implement the mechanisms of the model in a computer program. • Set the start parameters. • Let the program execute. • Observe the metrics that are of interest. • In this approach there is more questions which includes, how do one reflect mechanisms in a program if they are stochastic and not deterministic. What would be the meaning of the observations made from the results. This approach is more difficult than a deterministic approach.

Table 4.1.1 A short description of dynamic, continuous models and dynamic, discrete models

In conclusion a system can be seen as a recursive concept in which there is a choice of the abstraction level that depends on the purpose. Such a system can be constructed of many sub-systems. The global system can have a state that is determined by the sub-systems, which respectively may also have a certain state. These may be continuous or discrete, deterministic or stochastic state changes [23,24].

If such a system is to be modeled there are different types of models that can be used. These models include a physical/mathematical approach or a static/dynamic approach or a continuous discrete approach or finally a deterministic/stochastic approach. There are different investigation methods that can be used which include experiments, mathematical analysis or simulation. Now that the theoretical aspects of simulation has been covered. Different network simulation packages are compared to obtain the most appropriate simulation tool for the simulations.

In this study a mathematical approach will be used because a physical system is too expensive and not available. A dynamic approach within the mathematical model was used because the systems behavior was tested over a certain time. A continuous approach within the mathematical models was then used because the measured variables represent the system over time. Within the continuous model a stochastic approach was then used because the evolution of the states depended on random events.

4.3 Simulation packages

There are currently different simulation packages available, which, can be used to model a variety of different network topologies. This section contains tables representing different aspects of the different simulation tools. These simulation tools will be discussed on the main use of the tools, the package attributes contained within the simulator, the method of simulation and the type of simulation engine used by the network simulator.

4.3.1 The NETSIM simulation tool

	NETSIM
Use	NETSIM is an event driven network simulator for packet switched networks. It can be used to model all kinds of networks that are built up of components that transmit messages to one another.
Package attributes	This simulation package provides some component parameters, which may be changed during runtime. It also contains an event manager, I/O routines, various structural tools and a toolkit.
Simulator implementation	Links, hosts, switches and network connections are considered as components with each consisting of a class and a type that handles a set of events. These classes are stored in a doubly linked list while NETSIM is running and are presented by a data structure known as an action routine. Information that characterizes a component is classified as parameters. These parameters are normally displayed or saved in files. The simulator implements static routing with the use of components of type connection. Components that are currently available include Ethernet links, point-to-point links, switches, hosts, Purdue's implementation of TCP and a simple Poisson traffic source and packet sink. The simulator was designed for packet switched networks and therefore includes packet data types, but its data structure however is not constrained to any particular format.
Simulation engine	The simulation engine is written in C, and provides the means to schedule events. Events are inserted in an event queue and ordered according to the fire time. The event queue is therefore not a FIFO (First in first out) or LIFO (Last in first out) queue. The time is maintained in an unsigned 32-bit value, which enables the simulator to simulate a network for almost 12 hours.

Table 4.3.1 Table representing different aspects of the NETSIM simulation model

4.3.2 The NIST (National Institute of Standards and Technology) network simulation tool

	NIST
Use	NIST is an ATM network simulator, used for studying and evaluating the performance of ATM networks and is based on the previously discussed NETSIM. If the simulator is used as a planning tool it is run with various network configurations and traffic loads and produce statistics. If the network is used as a protocol analysis tool to study for example the total system effect of an ATM protocol, parameters such as protocol bandwidth, overhead and effectiveness of flow control can be viewed.
Package attributes	This simulators packaging is the same as that of NETSIM, the only difference however is that new network topologies can now be created directly via the user interface. This simulator runs on a UNIX platform.
Simulator implementation	The simulation implementation looks almost the same as the implementation for the NETSIM simulation tool. The only difference is that the components have been replaced in order to match the requirements set by ATM simulation. The available components within NIST for ATM network simulation purposes are physical links, ATM switches, Broadland terminal equipment (B-TE) and ATM applications.
Simulation engine	The simulation engine is the same as the simulation engine discussed for the NETSIM network simulator.

Table 4.3.2 Table representing different aspects of the NIST network simulation tool

4.3.3 The_CPSim network simulation tool

	CPSim
Use	CPSim is a parallel general-purpose simulation tool. This tool is commercially available and was designed by Groselj Boyan. The tool is used for parallel discrete event simulation. The model entails two methods of simulation, which includes optimistic and conservative. In the optimistic approach, the events are processed at each object as soon as they become available. In the conservative approach the events are processed at simulation time only if no more events are expected at simulation time.
Package attributes	This simulation tool does not contain a graphical interface. The designers of this simulation tool designed it to keep the performance at an optimum. This simulation tool also contains no built in functions and is therefore only intended for simulation professionals.
Simulator implementation	The simulator can be subdivided into two different sections, the CPSim libraries and the CPSim kernel. The CPSim kernel exists in two versions, one for parallel architectures and one for uniprocessors. This approach allows for portability on many platforms as well as allowing the development and debugging to be performed in the uniproccesor before its passed onto the parallel section.
Simulation engine	Event scheduling is done locally for each processor, and a FIFO approach is implemented to handle the events. Events are added to the back of the queue for processing. An object may have several input queues, but the scheduler takes the event with the shortest time stamp and processes it first. Synchronization between the different objects is achieved through the use of null events.

Table 4.3.3 Table representing different aspects of the CPSim network simulation tool

4.3.4 The_INSANE (Internet Simulated ATM Network Environment) network simulation tool

	INSANE
Use	Insane is a special purpose simulator and is used for evaluating performance of various IP over ATM policies a heterogeneous network.
Package attributes	INSANE is an object orientated event driven simulator with the core and some primitive objects written in C++. A library of Tcl scripts is supplied with the installation. Simulation and customization is therefore performed with the use of Tcl scripts, with no compilation needed for the construction of new scripts.
Simulator implementation	The simulator is single processed but can keep track of multiple simulations running on multiple machines. Applications run on a simulated IP protocol stack with two different data link layers, ATM or generic LAN. In the ATM data link layer approach a FIFO and RSCP (Rate Controlled Static Priority) method is used for queuing purposes. The IP module uses a static routing table, which is loaded at configuration time.
Simulation engine	Objects within INSANE is basically implemented a finite state machine fashion, and communicate through posting events to each other. The scheduler delivers events in their chronological order, which is based on a calendar queue. These types of queues (calendar queues) however need heavy processing.

Table 4.3.4 Table representing different aspects of the INSANE network simulation tool

4.3.5 The_NEST (Network Simulation Test-bed) network simulation tool

	NEST 2.5
Use	The main use of Nest is the simulating and prototyping of distributed algorithms and systems. This simulation tool also requires a low communication bandwidth.
Package attributes	The tools can be described as client server models with the display clients connected to the simulation server by a socket. Clients are independent programs used to create and configure a simulation model and control its execution. Creation and configuration is achieved through a GUI (Graphical User Interface) that communicates with the simulation via a TCP/IP connection. This GUI also allows the programmer to dynamically create or modify the network configuration. Because the GUI and the simulation architectures are divided it allows for CPU saving and it allows for many different users to connect and disconnect at any time.
Simulator implementation	NEST is implemented as a library of functions linked together though the users code. The simulation tool is written in C, and the simple communication facilities of NEST allows the user to only concern themselves with the higher layers of the protocol or distributed system. The topology of a network is created with the use of a set of graphical tools, with the connectivity of the network stored internally as a table.
Simulation engine	The simulation runs with a single UNIX process, and supports a lightweight process mechanism to facilitate the simulation of complex distributed systems. Each node contains its own stack of variables while global variables and dynamically allocated memory are shared between nodes. NEST is not a discrete event simulator, its simulation proceeds in a series of synchronization passes whose length can be specified by the user and modified dynamically. The node with the earliest simulated time is chosen through a round robin approach so that each node will receive the same amount of running time.

Table 4.3.5 Table representing different aspects of the NEST network simulation tool

4.3.6 The REAL (Realistic and Large) network simulation tool

	REAL 5
Use	REAL is based on the modified version of NEST 2.5 and is intended for the studying the dynamic behavior of flow and congestion control schemes in packet switched networks.
Package attributes	Real provides 30 modules written in C that emulates flow control protocols such as TCP as well as 5 scheduling disciplines such as FIFO, Fair Queuing, DEC Congestion Avoidance and Hierarchical Round Robin. The descriptions of the network and protocols-workload, are transmitted to the server through the use of an ascii representation called NetLanguage.
Simulator implementation	REAL is a rewritten version of NEST 2.5, with the main advantages of being faster, more general and cleaner. Outing is static and is based on Dijkstra's shortest path algorithm, with each node being a node, router or a sink.
Simulation engine	Refer to the previously discussed simulation engine section of NEST 2.5.

Table 4.3.6. Table representing different aspects of the REAL network simulation tool

4.3.7 The NS2 (Network simulator) network simulator tool

	NS version 2 (Network Simulator)
Use	NS2 is an object orientated event driven network simulator and is mostly used for small-scale simulations of queuing algorithms, transport protocol congestion control and multicast related work.
Package attributes	NS implements key protocol modules for unicast and multicast routing, reservation, transport and session protocols. NS is written in C++ and provides a compiled class hierarchy of objects written in C++ and an interpreted class hierarchy of objects written in Otcl.
Simulator implementation	Controlling, configuring and operating of the simulations are provided through the Otcl class simulator. The class provides the procedure to manage and create the desired network topology as well as to initialize the network and to choose the desired scheduler. The topology is created by the user through the use of Otcl standalone classes, links and nodes that provides some simple primitives. The function of the nodes is to receive a packet examine it and to map it to

-	the relevant outgoing interfaces. These nodes are composed of classifier objects, with each classifier concentrating on a certain part of the packet. The end points of the network where packets are received or transmitted is modeled by agents, NS currently supports various TCP agents, CBR, UDP, and other protocols such as RTP, RTCP and SRM. It must be remembered that NS don't include protocols for ATM networks. NS links are characterized in terms of delay and link bandwidth. They are built from a sequence of connector's objects. The data structure representing a link is composed of a queue of connector objects, its head, and type of link, time to live and an object that processes link drops. Connectors receive the packets, perform a certain function, and then transmit the packet to the next connector or to the drop object. Various kinds of links are supported, e.g. point-to-point, broadcast and wireless. The output buffers attached to a link in a router are modeled by queues. In NS, queues are considered as part of a link. NS allows the simulation of various queuing and packet scheduling disciplines within its C++ classes, which includes drop-tail FiFO queuing, RED buffer management, CBQ (priority and round robin), WFQ, Stochastic Fair Queuing (SFQ) and Deficit Round Robin (DRR). In order to analyze results, NS provides classes to trace each individual packet as it arrives, departs or is dropped, and to record any kind of counts, applied on all packets or a per-flow basis. The trace can be set or unset as desired by the user.
Simulation engine	NS's simulation engine is a single threaded engine and is extensible, programmable and configurable. Events within NS are described by a firing time and a handler function. The type of event scheduler used to drive the simulation can be chosen among the four presently available event schedulers: a simple linked-list (default), heap, calendar queue and a special type called real-time. Each one is implemented using a different data structure. The simple linked-list scheduler provides a list of events kept in time-

–	order from the earliest to the latest. This requires scanning the list to find the appropriate entry upon insertion or deletion. The entry at the head is always executed first. Entries with the same simulated time are extracted according to their order in the list. The heap scheduler code is borrowed from the NETSIM simulator. In the calendar queue scheduler implementation, events with the same "month/day" of multiples "year" are recorded in one "day". The real-time scheduler is still under development and is currently a subclass of the list scheduler. It is well suited when events arrive with a relatively slow rate.

Table 4.3.7 Table representing different aspects of the NS network simulation tool

4.3.8 The OPNET (Optimized Network Engineering Tools) simulation tool

	OPNET (Optimized Network Engineering Tools)
Use	OPNET is a commercial simulation tool and is used to simulate ATM networks as well as to validate the protocols used for various layers. OPNET can also be used for simulating packet switched radio networks.
Package attributes	OPNET can be subdivided into a few parts, which is the OPNET Modeler, OPNET Planner, Model Library and the Analysis Tool. Some of the features included in this generic simulator are an event-driven scheduled simulation kernel, integrated analysis tools for interpreting and synthesizing output data, graphical specification of models and a hierarchical object-based modeler. The OPNET modeler is intended for modeling, simulating, and analyzing of large communication networks. The modeling methodology of OPNET is organized in a hierarchical structure. At the lowest level, process models are structured as a finite state machine. State and transitions are specified graphically using state-transition diagrams whereas conditions that specify what happens within each state are programmed with a C-like language called Proto-C. Those processes, and built-in modules in OPNET (source and destination modules, traffic generators and queues for example) and are then configured with menus and organized into data flow diagrams that represent nodes

-	using the graphical node editor. Using a graphical network editor, nodes and links are selected to build up the topology of a communication network. The analysis tool provides a graphical environment to view and manipulate data collected during simulation runs. Results can be analyzed for any network element. While the OPNET planner is an application that allows administrators to evaluate the performance of communications networks and distributed systems, without programming or compiling. The modeling libraries are included in the OPNET planner and modeler and contain protocols and analysis environments. The ATM library contains 16 process models that implement the functions of ATM, the AAL and the IP interface for example. It provides for buffer management, congestion control, segmentation and reassembly, modeled explicitly or analytically. While the IP library contains interfaces to other protocols such as ATM or Ethernet for example.

Table 4.3.8 Table representing different aspects of the OPNET network simulation tool

The next simulation tool was chosen as the most appropriate simulation tool and will therefore be discussed in more detail. This simulation tool was chosen due to its ability to allow the programmer to physically manipulate the OSI layer at the different network layer entities.

4.3.9 CNET simulation tool

	CNET
Use	CNET is a network simulator which enables experiments with various data-link, network layer, routing and transport layer networking protocols in networks consisting of any combination of point-to-point links and Ethernet segments [21].
Package attributes	The CNET network simulation package is organized around the OSI reference model's idea of layering. CNET simulates the application

	and physical layers of the OSI seven layer reference stack leaving the programmer with the task of writing code for the remaining 5 layers. An advantage of CNET is that it automatically introduces corrupted and lost frames into the physical layer. CNET also provides a graphical representation of the network under execution and permits a number of attributes of the simulated network to be adjusted and modified in real time while the network are being simulated. [18]. CNET either displays the entire network under Tcl/Tk or runs rather less visually on an ASCII terminal. Under Tcl/Tk CNET provides the previously stated graphical representation of the network and permits the user to modify network attributes in real time during network simulation [21].
Simulator implementation	CNET is written and maintained by Professor C. McDonald from the University of Australia and currently only executes on several forms of UNIX (Linux (ELF only), DEC-OSF (v4.0), SunOS 4.1.x, Solaris 2.x, and SGI IRIX-5) [19]. CNET simulates each node's application layer by generating messages intended for delivery at other nodes. When such a node's application layer has a message ready for transmission, it calls one of the functions written by the programmer. The programmers code must then deliver all the messages to the intended destinations applications layers in an uncorrupted and ordered fashion. The communication between the different layers is however accomplished through an event based programming system [18]. CNET's simulation model is based on the OSI reference model. The next section will discuss CNET's simulation model in more detail.

Table 4.3.9 CNET network simulation tool attributes

4.4 CNET simulation model

From the above tables it can clearly be seen that the simulation tool of choice is CNET. This is due to the fact that CNET lets the programmer manipulate the different layers within the OSI stack. The prerequisite of this study is to extract data from the various layers within the OSI stack for TCP/IP transmission. This tool will therefore suite the investigation the best. The next sections will give more detail about the CNET simulation tool.

4.4.1 The CNET simulation model

In the CNET model, nodes are connected either by point-to-point links or Ethernet segments. These nodes however may be a router or a host, with hosts acting as workstations having an application layer for message generation and receiving. Hosts never generate messages for themselves, and routers never generate messages at all. Figure 4.4.1 represents the simulation model used by CNET for simulation purposes.

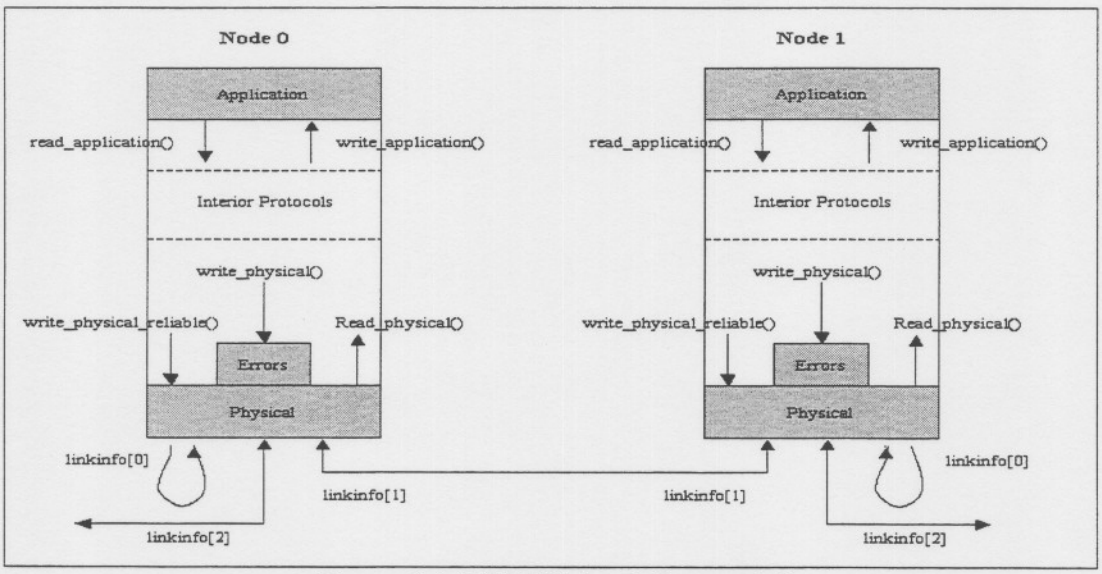

Figure 4.4.1 CNET simulation model

When a network is simulated within CNET, each node from 2 to hundreds are connected to each other via one or more physical point-to-point links or Ethernet segments. Links to each node are numbered from 0 to the physical number of links the node contains. Link number 0 of every node is dedicated as the loop back link, which simply delivers anything transmitted to it straight back to the host. If the nodes attributes must be accessed during runtime, it could be done via the elements of the C structure variables linkinfo[1], linkinfo[2], and so forth.

Each separate node is provided with an application layer and a physical layer, with a hidden error layer that resides above the physical layer and corrupts the transmitted data frames according to some specified probabilities. In conjunction to this the nodes initially have no knowledge of the network e.g. how many other nodes there are in the network or attributes of any other nodes or links.

The interior protocols are written by the programmers, which could also rewrite the application and physical layers if it is needed. These written protocols may be as complex or as simple as the programmer desires, and should follow a layered approach to isolate distinct responsibilities. Examples of these distinct responsibilities are.

- If the simulated network only consists of 2 nodes, only a single layer between the application layer and the physical layer are needed (this layer is usually termed the data link layer). This protocol will therefore simply have the responsibility to move frames reliably across the single bi-directional link between the two nodes.
- If the simulated network consists of more than two nodes, an additional layer between the data link layer and the application layer are needed. This layer is normally termed the network layer, and is responsible for managing packet routing, congestion and flow control and message fragmentation for example.
- If the nodes in the network may crash or the links between these nodes may be severed an additional layer between the network layer and the application layer is needed to ensure that old packets that have been transmitted and are currently floating around within the network between crashes are not considered as part of the current communication sequence. The above-mentioned layer is termed the session layer and provides the above-mentioned criteria.
- In a network where limited bandwidth must be considered as well as network security another additional layer between the session layer and the application layer must be inserted. This layer is termed the presentation layer and normally provides compression and/or encryption of the data before it is transmitted.

If a message is generated in CNET the message is generated within the application layer, the application layer will then inform the interior protocol layers that a message has been generated and it acquires delivery. The interior protocols don't care about the contents of the message it only treats it as a block of bytes. The physical layer that is accessible via each node's link number 1 number is then used to deliver the message. The entire transmission can be seen in figure 4.4.2.

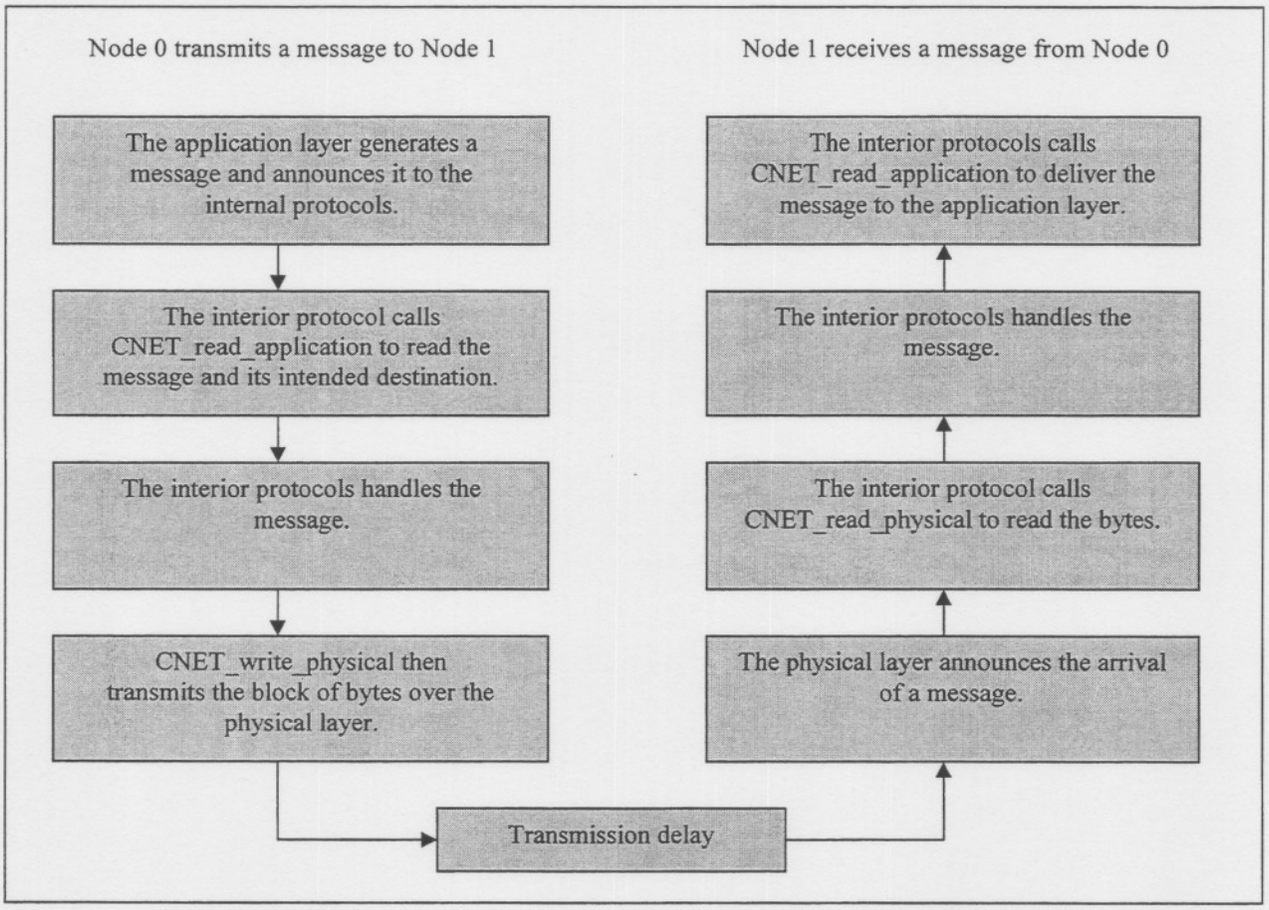

Figure 4.4.2 Node 0 transmits a message to Node 1

If a network topology is created in CNET, that uses the above discussed simulation model. Such a network topology can be created with the use of certain CNET topology file attributes. The next section discusses these attributes in more detail.

4.4.2 CNET topology files attributes

When networks are simulated in CNET the attributes and connections in these networks are defined within a topology file. These attributes include global attributes,

node attributes and link attributes with the global attributes affecting the execution of he entire simulation. Table 4.4.1 gives a description of some of the most commonly used attributes along with their data types, meaning and an example of each attribute [18,21].

Global attribute	Data type	Meaning	Example
bgimage	String	This global attribute provides the name of a GIF-format image file to be centered on the simulation's main window. The image file is sought via the CNETPATH environment variable if necessary.	bgimage = "Picture.gif"
drawframes	Boolean	The drawframe attribute requests that frames traversing the Physical Layer should be drawn under certain conditions. By specifying a special event handler, data link layer protocols in a 2-node network may request that their frames be drawn using different colors and lengths.	drawframes = true
showcostperbyte	Boolean	The showcostperbyte attribute requests that each link's costperbyte attribute value be displayed on the simulation's main window over each link. If the showcostperframe attribute is used the showcostperbyte attribute are overridden.	showcostperbyte = false
showcostperframe	Boolean	The showcostperframe attribute requests that each link's costperframe attribute value be displayed on the simulation's main window over each link. If the showcostperframe attribute is used the showcostperbyte attribute will be overridden.	showcostperframe = true
tracefile	String	This attribute requests that the trace	

-		of execution be mirrored in the named file when the –t option is given. Tracefiles can grow very large very quickly.	

Table 4.4.1 Global attributes

If the node and link attributes are defined before any nodes are defined, then they are considered as global attributes unless they are defined locally within a node or a link definition. This can be seen in figure 4.4.3 where the default messagerate is declared as 500ms, making it the default for all the nodes. But node Perth is given its own messagerate of 1000ms locally within the nodes definition. Table 4.4.2 and table 4.4.3 gives a brief description of the different node and link attributes and their respective data types, meanings and a brief example of each attribute [21].

```
compile = "stopandwait.c"

bgimage = "australia.gif"          } Global attributes
drawframes = true

messagerate = 500ms,               }
propagationdelay = 700ms,          } Link and node attributes
probframecorrupt = 3,              }

host Perth
{
ostype = "palm"
x = 100, y = 100
messagerate = 1000ms  ---------->  Local link/node attribute

link to Melbourne
}

host Melbourne
{
ostype = "linux"
east of Perth

link to Perth
(probframeloss = 2)
}
```

Figure 4.4.3 Example of a topology file

Node attribute	Data type	Meaning	Examples
address	Integer	The address attribute gives the unique address of each node.	address = 238
compile	String	This compilation string is used to declare the source file names containing the protocols for each node (This attribute locally overrides the –C option).	Compile= "protocol.c stats.c-lm"
messagerate	Time	This attribute dictates the rate at which the application layer can generate new messages for delivery.	messagerate = 10000usec messagerate = 2s
minmessagesize	Bytes	This attribute shows the minimum size of the messages generated by the application layer	minmessagesize = 100bytes
maxmessagesize	Bytes	This attribute shows the maximum size of the messages generated by the application layer. This attribute is bounded by MAX_MESSAGE_SIZE	maxmessagesize = 200bytes
nodemtbf	Time	This attribute gives the expected time between node hardware failures.	nodemtbf = 60000s
nodemttr	Time	This attribute gives the expected time taken to repair hardware failures.	nodemttr = 5000s
ostype	String	This attribute depicts the name of the operating system currently running on the node (only used to set the node's icon).	ostype = "linux"
outputfile	String	This attribute depicts the outputfile for each node. When	outputfile = "output"

		this attribute is used as a global attribute then outputfile is used as a filename prefix according to the –o option. When the attribute is used locally the outputfile indicates the entire filename.	
rebootargs	String	This attribute provides one or more white spaced separated command line arguments, which is passed to the EV_REBOOT handler. If it is used locally it overrides any arguments passed on by CNET's own command line.	rebootargs = "-fast –nonstats"
rebootfunc	String	The ANSI-C reboot function to call when the node reboots. If used locally it overrides the –R option.	rebootnode = "reboot_function"
trace	Boolean	This attribute is a Boolean, which indicates if an event tracing is required. It must be remembered that this attribute overrides the –t option.	trace = true
winopen	Boolean	This attribute requests that a node's window must be opened on startup.	winopen = false
winx, winy	Integer	Screen coordinates of the node's window under Tcl/Tk.	winx = 100, winy = 200
x, y	Integer	Coordinates of either a node's icon or the left hand end of an Ethernet segment on the main window.	x = 80, y = 120

Table 4.4.2 Node attributes

The second entry in the above table is the compile attribute which is a compilation string used to declare the source file names containing the protocols for each node. Because CNET must dynamically link compiled versions of protocols at run time, CNET performs all the necessary compilation and linking.

Protocolls are neither compared or linked by the programmer. Invoking CNET with a valid topology file will perform all necessary compilation and linking before commencing the simulation. CNET performs the rudimentary actions of make(3), compiling and linking files if the required target does not exist or is out-of-date with respect to source files. Strings are used to declare the location (filenames) of the source and shared object codes for the Application, "Central" and Physical layers used in each simulation. These strings may be provided on the command line, via the -A, -C, and –P options. The compilation string to compile the "Central" layers may also be specified with the compile node attribute in the topology file [21].

In their simplest form, compilation strings may present just a single C source file name, such as "protocol.c". If necessary, CNET will compile the file protocol.c into the object file protocol.o and then link this file to form the final shared object protocol.CNET. This final shared object file will then be used to provide the code for each node's relevant layer(s). In its more complex form, a compilation string may also include compilation switches, a number of source file names, and linker switches.

Each source file is compiled (if necessary) to create its object file, and all object files are then linked together to form a single shared object. The shared object's name is then derived from the first source file found. The embedded switches -l and -L are recognized as (assumed to be) linker switches with all other switches assumed to be preprocessor and compiler switches [21].

CNET introduces errors into the physical layer of the simulated network. The physical layer delivers frames between the nodes on unreliable, bi-directional links. These initial values of the link attributes (global or per-node) may be specified within the CNET topology file. Some of these link attributes may be adjusted while the simulation is running. Table 4.4.3 gives an overview of the most common link attributes used in CNET.

Link attribute	Data type	Meaning	Examples
bandwidth	Datarate	The bandwidth over a specific link.	bandwidth = 56Kbps
costperbyte	Cents	The cost per byte over this specific link.	costperbyte = 1c
costperframe	Cents	The cost per frame over this specific link.	costperframe = 5c
linkmtbf	Time	The expected time between link failures.	linkmtbf = 60000s
linkmttr	Time	The expected time to repair such a link failure.	linkmttr = 5000s
probframecorrupt	Probability	The probability that a frame will be corrupted over such a link.	probframecorrupt = 3 /* 1 in 8 */
probframeloss	Probability	The probability that a frame will be totally lost over such a link.	probframeloss = 4 /*1 in 16 */
propagationdelay	Time	The propagation delay over this specific link.	propagationdelay = 200usecs

Table 4.4.3 Link attributes

CNET employs an event driven programming style implying that execution proceeds when CNET informs protocols that an event of interest has occurred. No event will however be delivered when a node crashes, pauses or suffers a hardware failure. While an event handler is executing it will not be interrupted by the arrival of another event. To receive an event, event handlers must first be registered to receive incoming events with a call to CNET_set_handler.

CNET's Application layer

The responsibility of the application layer is to generate messages that will be transmitted to other nodes application layers. The destination nodes to which these messages will be transmitted are identified via their network addresses and not their node numbers. When the application layer receives a message for delivery it will read the message into a buffer supplied by the written protocol. If the read is

successful_then the address of the destination node and the actual length of the message will be filled in.

When a protocol are written it will typically need to restrict, or throttle, the generation of messages for certain destination nodes. This can easily be achieved through using the CNET_enable_application and CNET_disable_application function with a single parameter to identify the destination address to throttle.

Messages for all nodes accept the node from which the messages are transmitted may also be created and transmitted through the above-mentioned functions. The application layer has several functions which could be used when a protocol are created that uses the capabilities of the application layer. The following four functions acquire attention.

- **int** Cnet_read_application(CnetAddr *dest, **char** *msg, **int** *len);

In this function len must point to an integer, which indicates the maximum number of bytes that may be copied into msg. On the return len will point to another integer indicating the number of bytes that has been copied into msg. The network address of the required destination node is copied into destaddr.

- **int** Cnet_write_application(**char** *msg, **int** *len);

This function passes a number of bytes, pointed to by msg up to the application layer, with len pointing to an integer which indicates the number of bytes that must be taken from msg. And on return len will then point to an integer, which will indicate the number of bytes that has been accepted by the application layer.

- **int** Cnet_enable_application(CnetAddr destaddr);

This function permits the application layer to generate messages to the node specified by the indicated network address. If the network address is ALLNODES then the messages will be generated for all the nodes within the network accept the transmitting node.

- **int** Cnet_disable_application(CnetAddr destaddr);

When this function is used the application layer will be prevented to generate messages to the node indicated to by the network address. If the network address is ALLNODES the application layer will be prevented to transmit messages to any node within the network.

CNET's physical layer

The responsibility of the physical layer is to deliver data frames between nodes, these nodes may be connected via point-to-point links or Ethernet segments. Each physical link is numbered from 1 to its total number of links as previously explained in the CNET simulation model section. The physical layer will randomly drop data frames on the point-to-point links but on the loopback links as well as the Ethernet segments.

When a data frame must be transmitted over the physical link the data frame is written to the physical layer with the Cnet_write_physical function. When the Cnet_write_physical function is used, the length of the frame must be specified. CNET will on the return from the function indicate how many bytes the physical layer accepted. When CNET informs the destination node that a frame has arrived, the handler for EV_PHYSICALREADY should read that frame.

On the return from a successful call to Cnet_read_physical, the written protocol is informed on which link the frame arrived and the length of the frame. There are two additional physical layer functions provided to assist in the debugging of multi-layered protocols. These functions are Cnet_write_physical_reliable and Cnet_write_physical and are almost identical except that frames transmitted with Cnet_write_physical_reliable will not be dropped or corrupted. The Cnet_write_direct also bypasses all the physical layer errors and instructs a message to be transmitted directly to the destination node. The most commonly used functions used for the physical layer can be summarized as follows.

- **int** Cnet_write_physical(**int** link, **char** *frame, **int** *len);

This function passes a number of bytes pointed to by frame down to the physical layer which will then attempt to transmit these bytes to the link indicated to by link. In special cases the link may transmit a frame to itself via the loopback link. Again len must point to an integer, which indicates the number of bytes that must be taken from the frame, and on return it will indicate to an integer showing the number of bytes that was accepted by the physical layer.

- **int** Cnet_write_physical_reliable(**int** link, char *frame, **int** *len);

This function is identical to the Cnet_write_physical function. The only difference is that a reliable data link is provided. Therefore no frames will be dropped or corrupted by the physical layer.

- **int** Cnet_write_direct(CnetAddr destaddr, **char** *msg, **int** *len);

This function is very similar to Cnet_write_physical_reliable except that the network address of the required destination may be specified. Messages transmitted with this function are transmitted over link 1. If the destination is BROADCAST the message will be transmitted to all the nodes except the sending node. When this function is used the message is delivered in 1 msec, regardless of all the hops, propagation delays and bandwidth within the network.

- **int** Cnet_read_physical(**int** *link, **char** *frame, **int** *len);

This function accepts the specified maximum of bytes from the physical layer and places them in the address that is pointed to by frame. In this case len points to an integer indicating the maximum number of bytes that may be copied into frame. On return, len will point to an integer indicating the number of bytes taken from the physical layer and link will point to an integer indicating on which link the bytes were received.

4.5 Conclusion

CNET allows the programmer manipulate the seven layers of the OSI stack. This fact enables the programmer to create a protocol and extract the necessary data at the different layers in the OSI stack. The criteria to which the simulation tool must accord can be summarized as follows.

- The simulation tool must allow the programmer to manipulate the seven layers of the OSI stack.
- The simulation must be able to implement the TCP/IP protocol.

Therefore CNET suites the simulation criteria the best and will be used for simulation purposes. The next section explains the problem methodology that will be used in the network simulation study.

Chapter 5. Problem methodology

Abstract – This chapter will give the reader the network topologies that will be used along with the different tests that will be performed to determine the data available to probe in the network. The results obtained through implementing the steps will be presented in chapter 6 of this document.

5.1 Introduction

The aim of the methodology is to give answers to two fundamental questions that were identified in chapter 1 of this document.

1. How do the probe locations within the OSI layers influence the measurements?
2. How do the probe locations in the network influence the measurements?

Now that the simulation tool has been chosen, the network simulations are implemented and data is extracted and saved for different probe locations. The problem methodology followed will deduce answers for the above-mentioned questions. From the section 3 it is clear that layer 2 information gives CoS information and layer 3 gives QoS information.

5.2 Problem methodology

The method that was implemented to deduce answers for the above-mentioned questions were as follows:

1. The network protocols were created and validated, which were followed by the creation of the different network topologies. The written code are then validated with existing examples.
2. Tests were performed on the simulation topologies.
3. Data were extracted from different locations within the network and saved for analysis. The extracted data was then analyzed and conclusions were drawn. The method used can be seen in Figure 5.2.1 followed by a definition of each step.

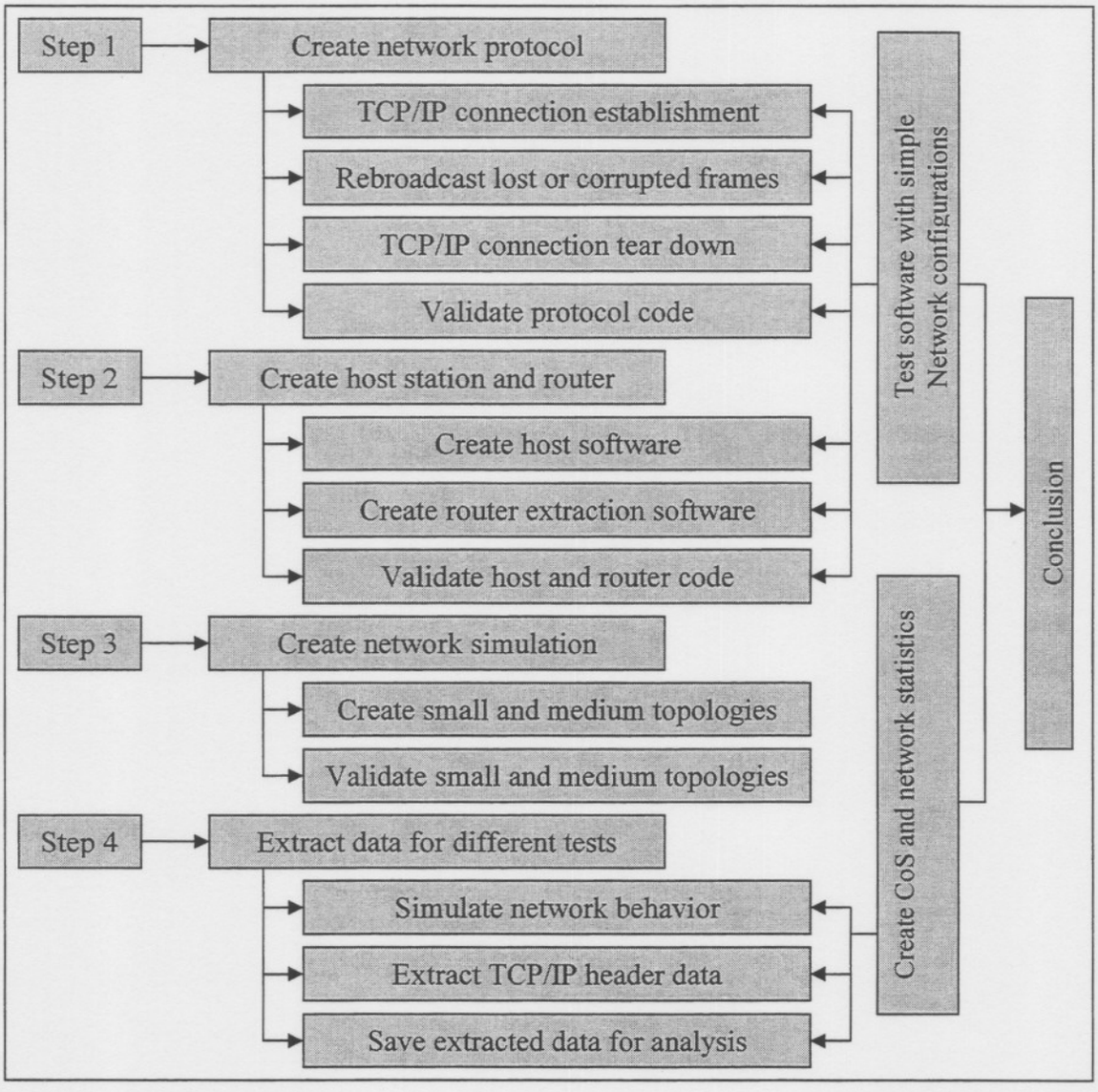

Figure 5.2.1 Methodology used

5.3.1 Step 1: Create network protocols

A protocol is needed to simulate traffic flow on the different network topologies. The protocol must exhibit some distinct features, which includes.

- The necessary headers should be added to the data packet at each OSI layer.
- If a packet arrives at a router, the router should extract the data from the first three layers of the OSI stack headers and save the data to a file.

- A host should generate a message, attach the desired headers and transmit it to a destination, which in turn should dismantle the packet do CRC checks and if necessary transmit an ACK message to the transmitting station.
- A TCP/IP protocol replica should therefore be created, which will be used to transmit data from one node to another through a series of routers.
- It must also be remembered that the nodes and routers are independent and can only communicate with each other through the physical medium.

A graphical representation of the protocols network layer functionality can be seen in figure 5.3.1 along with a graphical representation of the protocols data link layer functionality in figure 5.3.2.

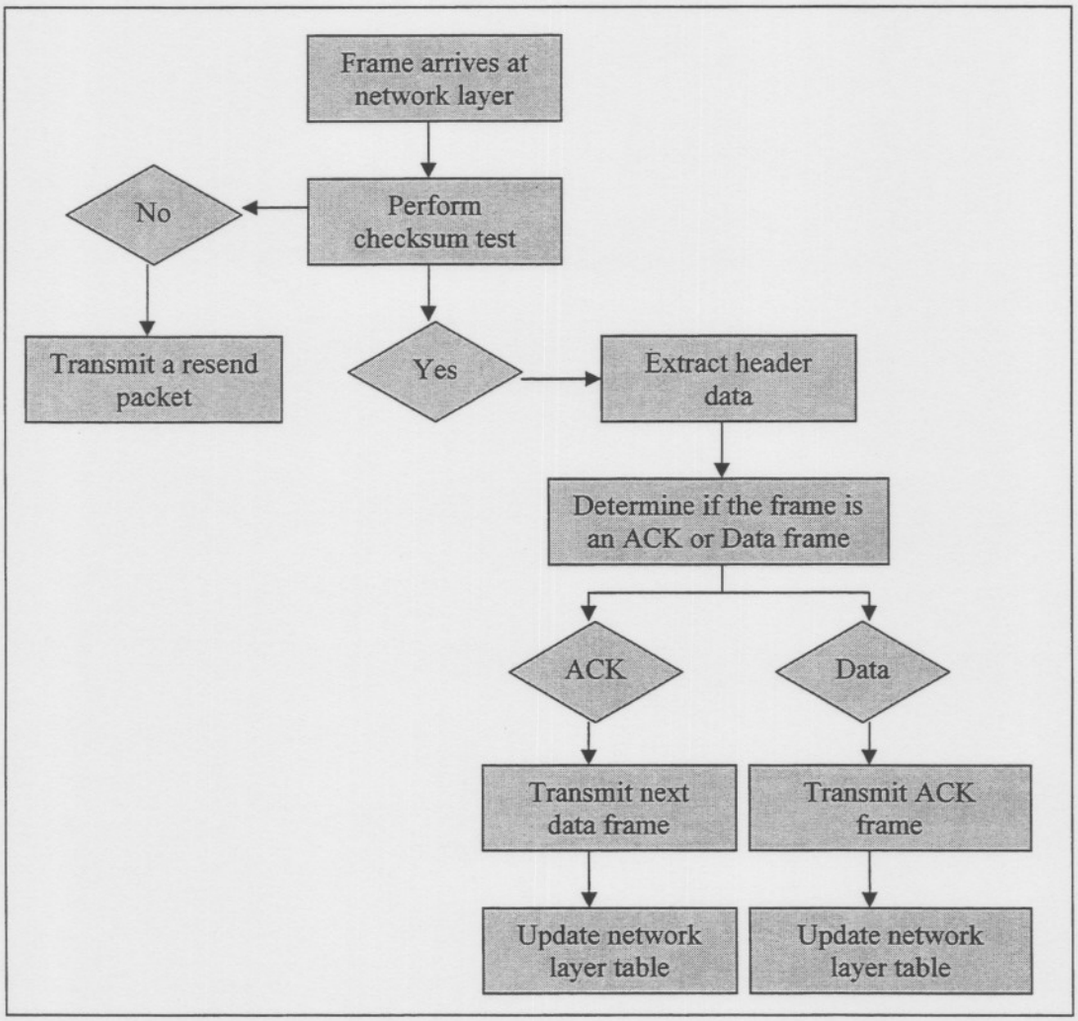

Figure 5.3.1 Network layer flow diagram

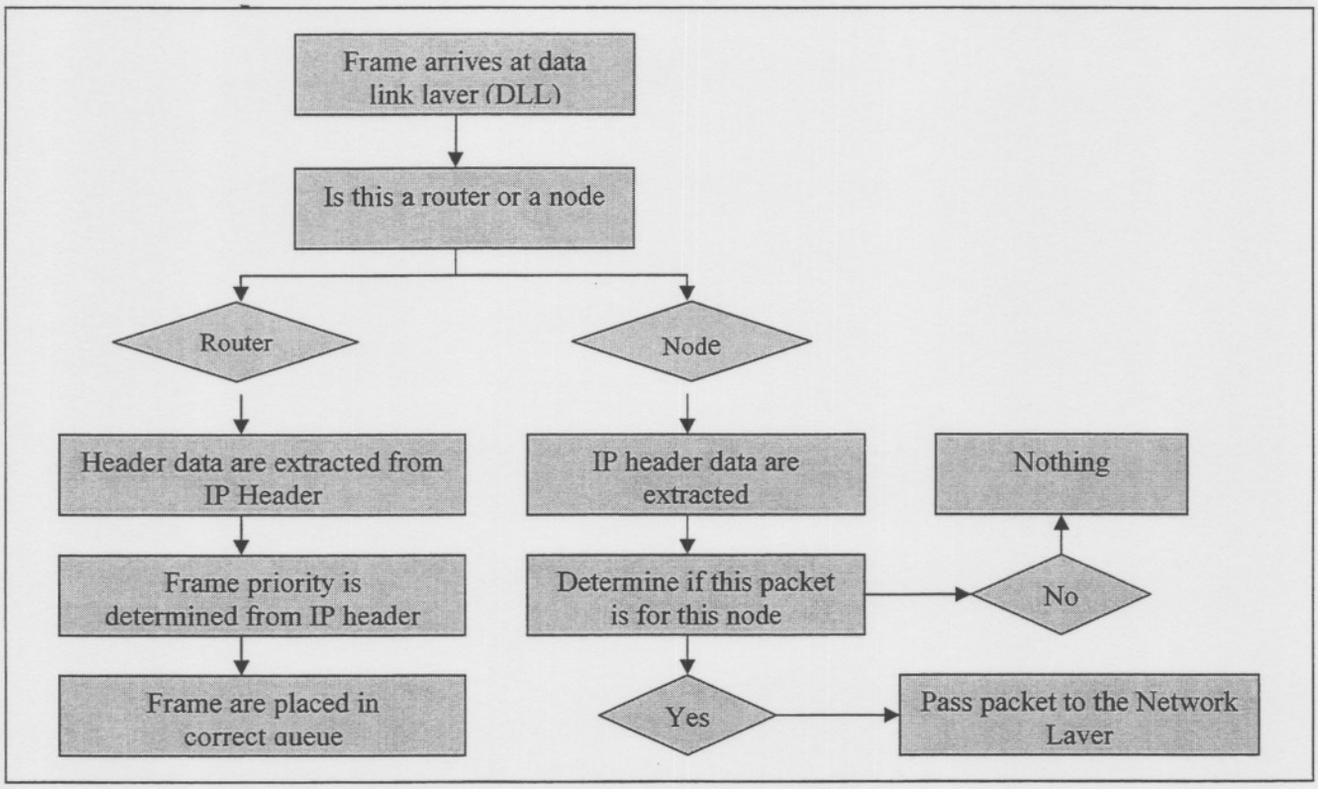

Figure 5.3.2 Data link layer flow diagram

After the protocol had been created the necessary software needed to extract the data from the TCP/IP headers at the routers and the nodes must be created. The next section explains the host and router software.

5.3.2 Step 2: Create host, station and router software

Code is needed to extract the header data from the TCP/IP headers at the routers within the network. Code is also needed at the nodes to handle the incoming frames accordingly. The main features of the router code can be classified as follows and can be seen in figure 5.3.5.

- If a frame arrives at a router the router should extract the necessary data from the TCP/IP headers and store it in a database.
- The time to live parameter should be analyzed and the frame should be discarded if necessary.
- Check destination address and retransmit the frame on the correct link.

The main features of the host software can be classified as follows and can be seen in figure 5.3.3.

- Check if frame is destined for this node. If not discard the frame, else pass frame up to the next layer
- Check if frame is an acknowledgement or data frame.
- If frame is an acknowledgement frame, transmit the next data frame. If the frame is a data frame transmit an acknowledgement frame.

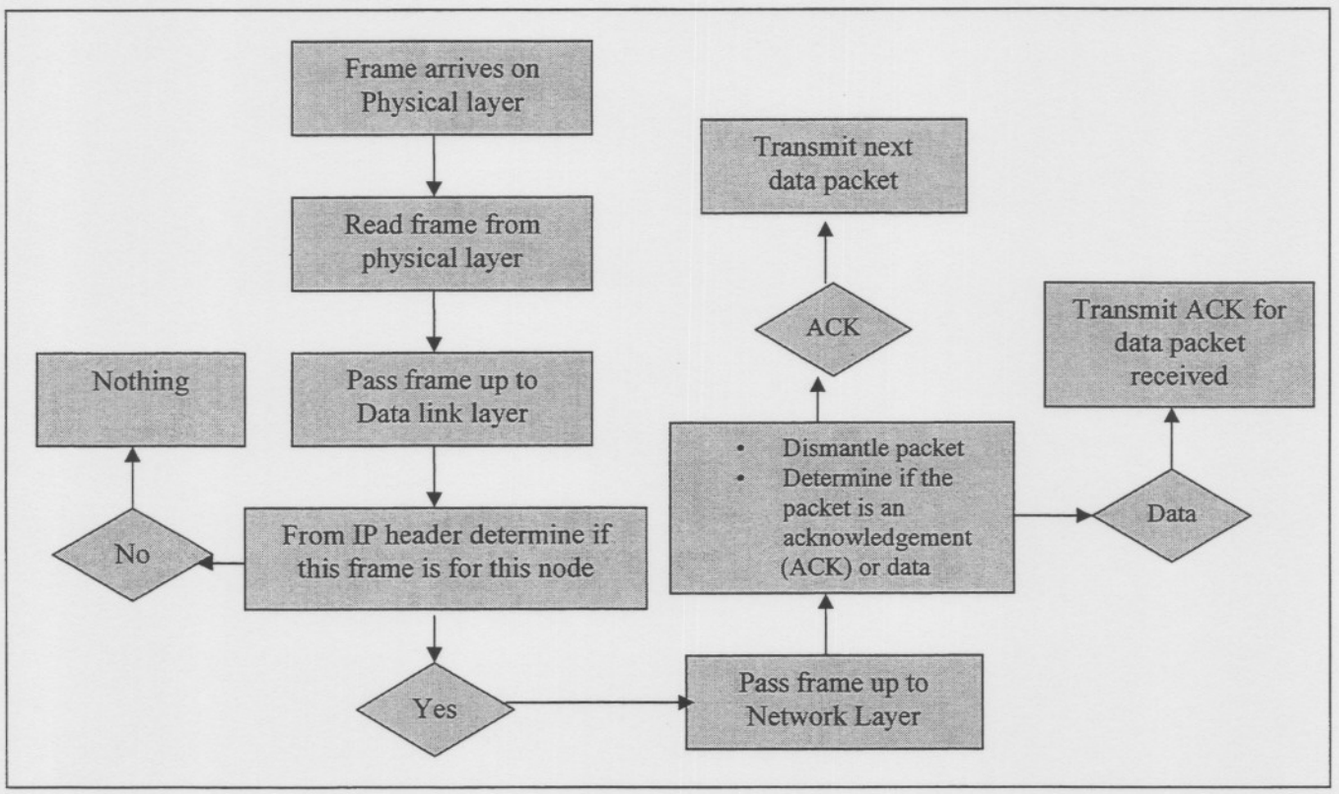

Figure 5.3.3 Host software layout

The router stores the header information in files for analysis. The format that was programmed in which the data will be saved in can be seen in figure 5.3.4 followed by a definition of the fields in table 5.3.1.

R	Time	Packet Number	Source address	Destination address	Data	ACK	TCP destination port	TCP host port	Size

Figure 5.3.4 Data format

Symbol	Meaning
R	A positive value of 1 indicates that a frame has been transmitted by the router/node while a negative value of –1 indicates that the frame has been received by the router/node.
Time	The time the frame was received or transmitted from the router/node.
Packet Number	Indicates the number of the frame transmitted.
Source Address	Source address, from which the frame originated.
Destination Address	Destination address for which the frame was intended.
Data	A positive value of 1 indicates that the frame contains data.
Acknowledgement	A positive value of 1 indicates that the frame contains an acknowledgement.
TCP destination port	The address of the TCP destination port.
TCP source port	The address of the TCP source port.
Size	Frame size.

Table 5.3.1 Data format explanation

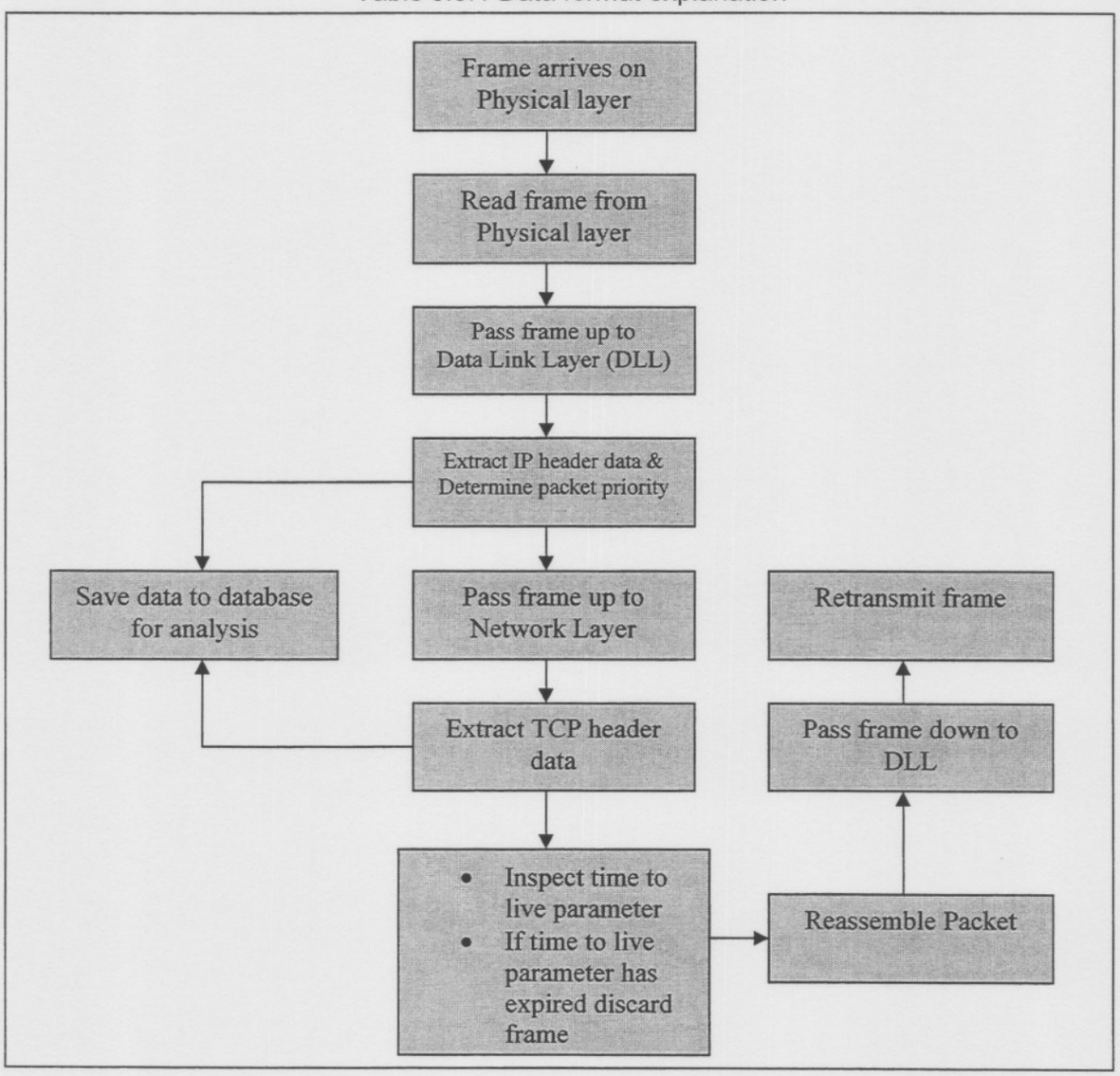

Figure 5.3.5 Router flow diagram

The protocol code and the host and router software can now be used on implemented simulation topologies. The next section shows the topologies that will be used for simulation. The written software were validated with examples from literature.

5.3.3 Step 3: Create network simulation topologies

When a simulation network is constructed it is of cardinal importance that the network must simulate the kind of network topology that currently exists and on which the IP probes must be installed. If this criterion isn't adhered to, the simulation will be of no use.

The exact layout and construction of the telecommunication company's networks aren't available. Therefore two random topologies will be used. Firstly a simple network topology and secondly a medium network topology.

The small network simulation topology is constructed of 2 core routers, 2 edge routers and 2 nodes (Figure 5.3.6). These nodes may consist of other small networks and node topologies. The medium network topology can be seen in figure 5.3.7, and consists of 3 edge routers, 3 core routers and 5 nodes.

These topologies will be used in conjunction with the protocol, host and router software to simulate the extraction of IP network statistics. The next section will give more information about the different tests that will be performed on these simulation networks.

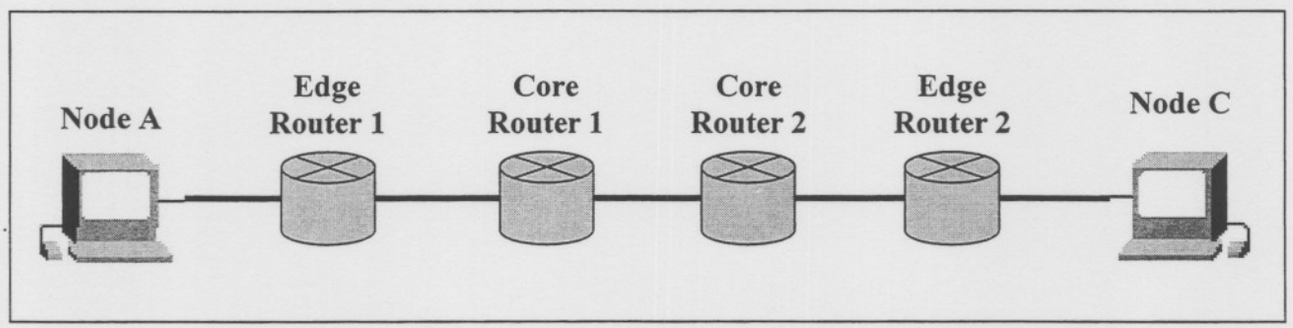

Figure 5.3.6 Small network topology

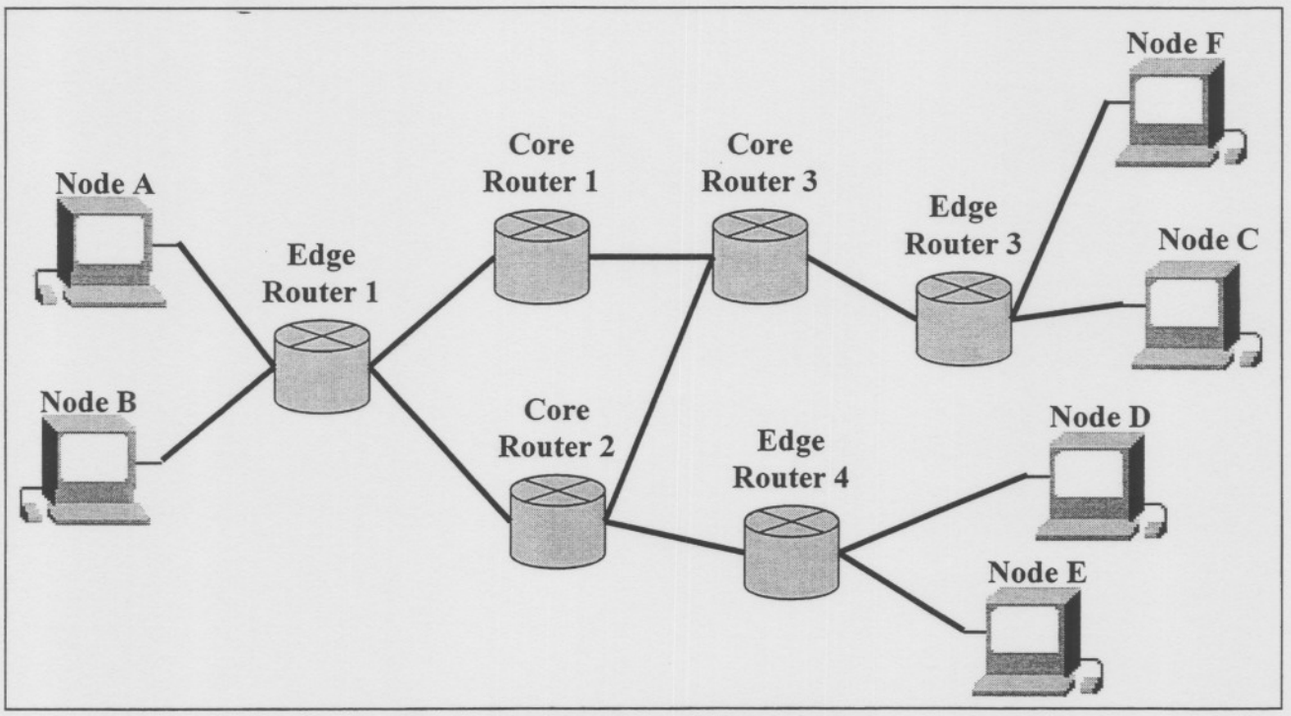

Figure 5.3.7 Medium network topology

5.3.4 Step 4, Extract data for different tests

Different tests will be performed on the network to simulate traffic flow. Sessions will be established between nodes and data will be transmitted from one node to another. The information in the TCP and IP headers will be monitored and saved for analysis. Sessions will be created on both the small and the medium networks.

There are a total of four tests that will be performed and documented. These tests include tests with packet loss, on medium and small networks as well as tests performed with dedicated links on the medium network. Descriptions of these tests can be seen in the following tables.

Test 1 (Small network session without loss)	
Test setup	• A small network topology is used (2 core routers, 2 edge routers, 2 nodes (hosts)). • A session is established between Node A and Node C. • The TCP/IP header data are extracted from the headers at the nodes, core routers and edge routers. • The extracted header data are stored in files for analysis. • The edge-to-edge delay is kept constant while the edge-to-host delay is varied. Therefore the latency will be varied and the jitter between the edge routers will be constant.
Objective	The aim of the test is to determine the data available to probe in a small network with no packet loss. Different measurements will be made, which includes: • Session jitter perceived towards node A and node C at the routers and nodes. • Session jitter perceived between the edge routers (Edge-to-edge). • Session jitter perceived between the nodes (End-to-end). • The throughput in each direction will also be calculated at each router and node. The jitter is varied dramatically to see if these changes can be monitored. The throughput will therefore be low because of the large end-to-end delay variations. The same applies for the other tests.
Extracted parameters	• TCP/IP header data at every node and router.

Table 5.3.2 Test 1 explanation

Test 2 (Small network session with loss)	
Test method	• A small network topology is used (2 core routers, 2 edge routers, 2 nodes (hosts)). • A session is established between Node A and Node C. • The TCP/IP header data are extracted from the headers at the nodes, core routers and edge routers. • The extracted data are stored in files for analysis. • The edge-to-edge delay is kept constant while the edge-to-host delay is varied. These variations may vary from a few ms (minimum 40 ms) up to a few seconds. Therefore the latency will be varied and the jitter between the edge routers will be constant. • In this session packets are lost on the links. These variations may vary from a few ms (minimum 40 ms) up to a few seconds. The throughput at the different network elements will therefore be different as well as the perceived jitter.
Objective	The aim of the test is to determine the data available to probe in a small network with packet loss. Different measurements will be made, which includes: • Session jitter perceived towards node A and node C at the routers and nodes. • Session jitter perceived between the edge routers (Edge-to-edge). • Session jitter perceived between the nodes (End-to-end). • The session jitter at each router will also differ due to packet loss. • The throughput in each direction will also be calculated at each router and node. The throughput will however differ at the various network elements due to packet loss.
Extracted parameters	Data will be extracted from the TCP/IP headers and saved for analysis.

Table 5.3.3 Test 2 explanation

Test 3 (Medium network sessions no loss flooding)	
Test method	• A medium network topology is used (3 core routers, 3 edge routers, 6 nodes (hosts)). • The network is flooded and two sessions are established. Session 1 is between node A and node E and session 2 is between node B and node C. • The TCP/IP header data are extracted from the headers at the nodes, core routers and edge routers. • The extracted data are stored in files for analysis. • The edge-to-edge delay is kept constant while the edge-to-host delay is varied. These variations may vary from a few ms (minimum 40 ms) up to a few seconds. Therefore the latency will be varied and the jitter between the edge routers will be constant.
Objective	The aim of the test is to determine the data available to probe in a flooded medium network. Different measurements will be made, which includes. • Session jitter perceived towards node A, node B, node C, node D, node E and node F at the routers and nodes for the implicated sessions. • Session jitter perceived between the edge routers for the implicated session (Edge-to-edge). • Session jitter perceived between the nodes (End-to-end) for the implicated session. • The throughput in each direction will also be calculated at each router and node for each session.
Extracted parameters	The TCP/IP header data are extracted from the headers at each node and router and saved for analysis.

Table 5.3.4 Test 3 explanation

Test 4 (Medium network sessions with dedicated links)	
Test method	• A medium network topology is used (3 core routers, 3 edge routers, 6 nodes (hosts)). • Three communication sessions are established, with each communication session having a different priority level. Dedicated paths (VPN's) are established between the nodes over the network. Session 1 is between node A and node E. Session 2 is between node B and node C and session 3 is between node D and node F. • The TCP/IP header data are extracted from the headers at the nodes, core routers and edge routers. • The extracted data are stored in files for analysis. • The edge-to-edge delay is kept constant while the edge-to-host delay is varied. These variations may vary from a few ms (minimum 40 ms) up to a few seconds. Therefore the latency will be varied and the jitter between the edge routers will be constant.
Objective	The aim of the test is to determine the data available to probe in a medium network running VPN's. Different measurements will be made, which includes. • Session jitter perceived towards node A, node B, node C, node D, node E and node F at the routers and nodes for the implicated sessions. • Session jitter perceived between the edge routers for the implicated session (Edge-to-edge). • Session jitter perceived between the nodes (End-to-end) for the implicated session. • The throughput in each direction will also be calculated at each router and node for each session.
Extracted parameters	The TCP/IP header data are extracted from the headers at each node and router and saved for analysis.

Table 5.3.5 Test 4 explanation

Through the use of these tests, the ability to monitor the TCP/IP headers for different types of sessions, different amounts of sessions and sessions in different topologies can be investigated.

5.3.5 Step 5, Create QoS statistics & Step 6, Draw conclusion

After the tests have been completed the extracted results will be analyzed and the sessions QoS parameters will be evaluated. These results will then be used to comment on the validity of only using TCP and IP headers to measure a session's QoS parameters. A conclusion will be drawn about the effectiveness of the measurements based on the probe locations as well as the OSI level the data are extracted on.

5.3 Conclusion

From the above-mentioned methodology enough data can be collected to answer the two fundamental questions mentioned in this chapter's introduction.

1. How do the probe locations within the OSI layers influence the measurements?
2. How do the probe locations in the network influence the measurements?

The aim of the methodology is to determine the data available to probes at different locations and OSI levels within a TCP/IP network. The next chapter gives the results of the mentioned methodology, as well as a discussion of the results that were obtained.

Chapter 6. Results

Abstract – This chapter contains the results and a discussion of the results obtained from the tests discussed in chapter 5 (Problem methodology) of this document. A final conclusion on the results can be found in chapter 7 of this document.

6.1 Introduction

Chapter five gave an overview of the different tests that was performed on the small and medium networks. Chapter six gives the results obtained from these tests as well as a discussion of the results.

6.2 Monitoring a session without loss in a small network

Figure 6.2.1 represents the network topology that was used in the test. The total end-to-end minimum delay between the nodes is 250ms. This value is calculated from the minimum delay values given to the links. The values used on the different links can be seen in table 6.2.1. The average jitter, packet loss, end-to-end delay and throughput measured between the network elements towards node A and Node C can be seen in table's 6.2.2 and 6.2.3. Packet lengths were varied according to a normal distribution between 4kb and 10kb with packet sizes taken at random. A definition of the normal distribution can be seen in Appendix B of this document. Jitter is defined as the variation in the time between packets arriving at the nodes.

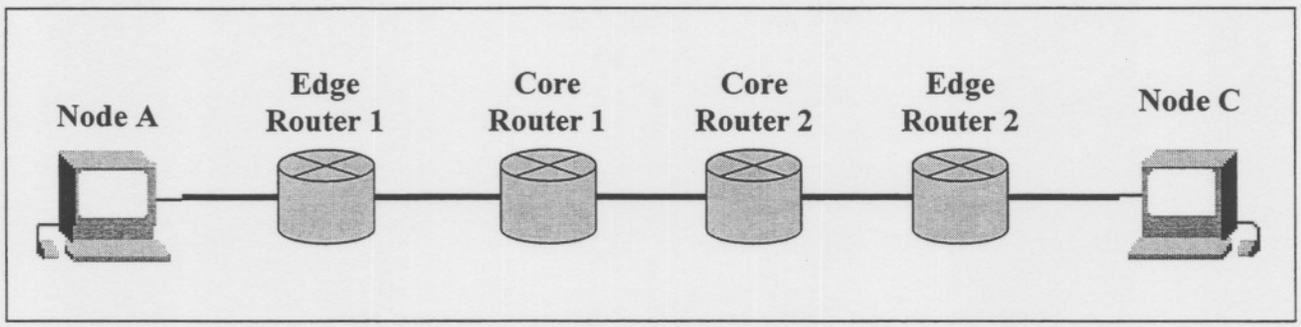

Figure 6.2.1 Small network topology

Link	Bit rate	Line Delay
Node A – Edge Router 1	10 Mbps	40 ms
Edge Router 1 – Core Router 1	15 Mbps	15 ms
Core Router 1 – Core Router 2	15 Mbps	15 ms
Core Router 2 – Edge Router 2	15 Mbps	15 ms
Edge Router 2 – Node C	10 Mbps	40 ms

Table 6.2.1. Link delay information

Table's 6.2.2 and 6.2.3 represents the results obtained from test 1 for a small network with no packet loss.

	Edge1-to Edge2	Node A-to-Node C
Throughput	4,9 kb/s	4,9 kb/s
Average delay	0 ms	4 ms
Average Jitter	0 ms	1.3 ms
Packets lost	0	0

Table 6.2.2 Measured router details towards node C

	Edge1-to Edge2	Node C-to-Node A
Throughput	4,9 kb/s	4,9 kb/s
Average Delay	0 ms	3.9 ms
Average Jitter	0 ms	1.4 ms
Packets lost	0	0

Table 6.2.3 Measured session details towards node A

The maximum and minimum end-to-end delay for both directions of the session can be seen in table 6.2.4. The average jitter differs between the end-to-end and edge-to-edge calculations. This can be explained by the fact that the edge-to-end minimum delays aren't included in the edge-to-edge measurements. The internal core delays weren't varied as well. The average delay for the node-to-node delays also varies because the delay on the edge-to-user links was varied at random. The same applies for all the other tests performed on the network.

	Towards node C	Towards node A
Maximum	9.3 ms	2.4 ms
Minimum	2.6 ms	1.2 ms

Table 6.2.4 Maximum and minimum end-to-end delays

In the above table the minimum values aren't 0 ms, this is due to the fact that the delays on the edge-to-user links were varied from 0 ms upwards with a normal distribution between 0 and 10ms.

The results obtained supports the results obtained in the paper study. From the table it is clear that the throughput, average end-to-end delay, average jitter, maximum and minimum delay and packet loss could be measured between the nodes within the packet loss network using only TCP and IP header data. In chapter 3 a paper study was performed to determine the QoS data available from the TCP/IP headers. In chapter 5 a methodology was presented. Software were created and validated and the results are displayed in chapter 6. The results obtained supports the results obtained within the paper study.

6.3 Monitoring a session with loss in a small network

The same network topology displayed in figure 6.2.1 was used in this test with the same end-to-end minimum delay between the nodes (250ms, table 6.2.1). Table's 6.3.1 and 6.3.2 represents the results obtained from test 2 for a small network with packet loss.

	Edge1-to Edge2	Node A-to-Node C
Throughput	3.5 kb/s	3.5kb/s
Average Delay	0 ms	1 ms
Average Jitter	0 ms	0.7 ms
Packets lost	0	113 of 162 (70%)

Table 6.3.1 Measured router details towards node C

	Edge2-to Edge1	Node C-to-Node A
Throughput	1.4 kb/s	1.8 kb/s
Average Delay	0 ms	3 ms
Average Jitter	0 ms	2.1 ms
Packets lost	0	0

Table 6.3.2 Measured session details towards node A

The test was performed in a fashion where packets from Node C towards Node A were dropped at a high rate and packets from Node A towards Node C were not dropped. These packets were also dropped before the edge router to determine if

these session parameters could be measured through extracting only TCP and IP header data. The maximum and minimum end-to-end delay for both directions of the session can be seen in table 6.3.3. As previously mentioned the delay values were chosen random from a normal distribution between 0 and 10ms

	Towards node C	Towards node A
Maximum	2.3 ms	5.4 ms
Minimum	0.6 ms	1.2 ms

Table 6.3.3 Maximum and minimum end-to-end delays

The extracted data showed that the constructed session's independent differences could be measured through extracting the data from the TCP and IP headers. The maximum and minimum roundtrip delays, throughput, jitter and packet loss were determined for the session as well. These results support the suspected theoretical results.

6.4 Monitoring multiple sessions through flooding on a no loss medium sized network

In this test multiple communication sessions were established between nodes over a medium network (Figure 6.4.1) and maintained through flooding. The TCP/IP packet headers received at each router and node were recorded and analyzed.

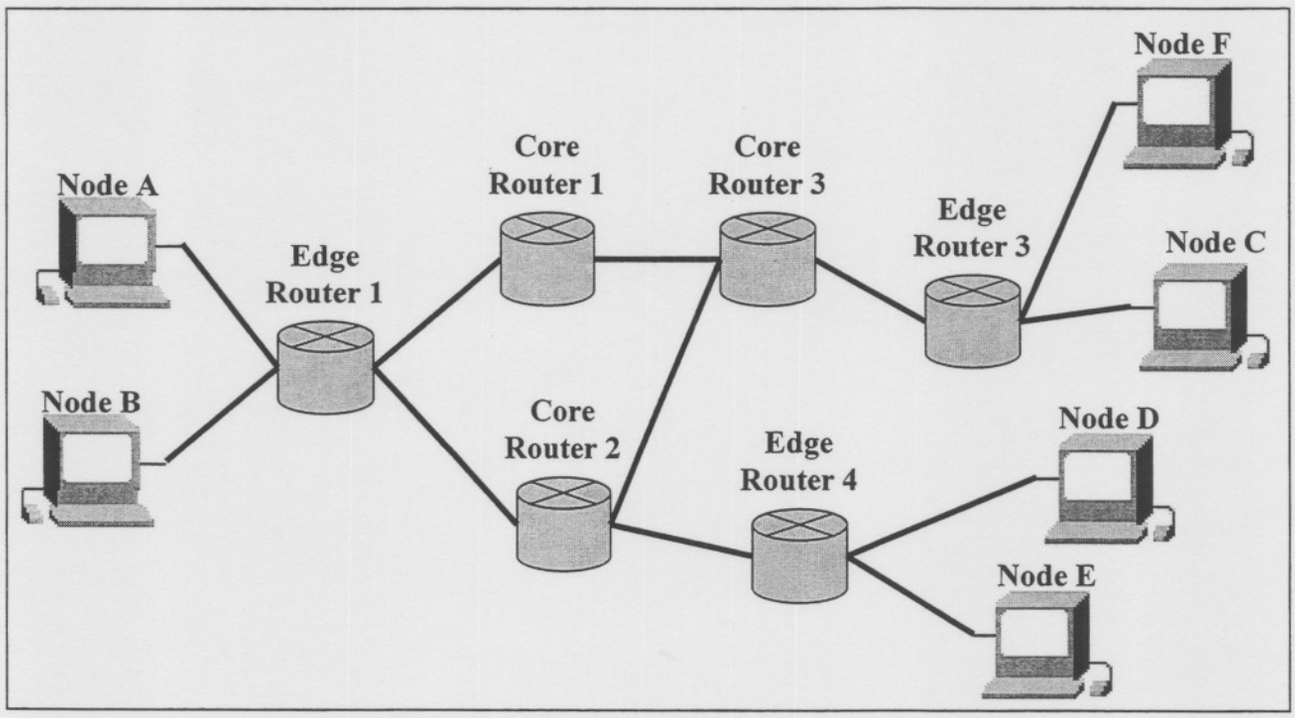

Figure 6.4.1 Medium network topology

The different link properties of the links in the medium network topology can be found in table 6.4.1 followed by the different session information for the session between Node A and Node E in tables 6.4.2 and 6.4.3.

Link	Bit rate	Line Delay
Node A – Edge Router 1	10 Mbps	40 ms
Node B – Edge Router 1	10 Mbps	40 ms
Edge Router 1 – Core Router 1	15 Mbps	20ms
Edge Router 1 – Core Router 2	15 Mbps	20ms
Core Router 1 – Core Router 3	15 Mbps	20ms
Core Router 2 – Core Router 3	15 Mbps	20ms
Core Router 2 – Edge Router 4	15 Mbps	20ms
Core Router 3 – Edge Router 3	15Mbps	20ms
Edge Router 4 – Node D	10 Mbps	40 ms
Edge Router 4 – Node E	10 Mbps	40 ms
Edge Router 3 – Node C	10 Mbps	40 ms
Edge Router 3 – Node F	10 Mbps	40 ms

Table 6.4.1 Medium network topology link information

	Node A-to-Node E	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	3.33 kb/s	5.15 kb/s	5.15 kb/s	3.44 kb/s
Average Delay	390 ms	223 ms	225 ms	224 ms
Average Jitter	1.2 ms	1.3 ms	1.4 ms	1.3 ms
Packets lost	0	0	0	0

Table 6.4.2 Measured session details towards node E from Node A

	Node E-to-Node A	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	3.33 kb/s	8.31 kb/s	8.31 kb/s	6.67 kb/s
Average Delay	392 ms	223 ms	222 ms	221 ms
Average Jitter	1.4 ms	0.9 ms	1.2 ms	1.3 ms
Packets lost	0	0	0	0

Table 6.4.2 Measured session details towards node E from Node A

The same tables can be constructed for the other session. From these tables it is clear that the throughput values differ dramatically because of packets traversing

through the network and double counting due to packets traversing through certain routers before being discarded due to their hop-rate reaching a maximum. This also needs lots of processing time making the extraction of data from a flooded network complex and not accurate. The next test investigated a VPN approach to maintaining sessions.

6.5 Medium network sessions with dedicated links

In this test multiple communication sessions were established between nodes over a medium network (Figure 6.4.1) with link delays shown in table 6.4.2. Frames were transmitted over dedicated links (VPN's). Dedicated paths were established between node A and node E, node B and Node C and Node F and Node D. Packets were dropped in these sessions towards node E, Node C and Node F before the packets entered the edge routers. The measured session details can be seen in table's 6.5.1 to 6.6.6.

	Node A-to-Node E	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	3.49 kb/s	------	2.45 kb/s	------
Average Delay	2 ms	------	0 ms	------
Average Jitter	0.8 ms	------	0 ms	------
Packets lost	13	-------	0	------

Table 6.5.1 Measured session details towards node E from Node A

	Node E-to-Node A	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	1.59kb/s	------	1.51 kb/s	-------
Average Delay	1 ms	------	0 ms	------
Average Jitter	0.9 ms	------	0 ms	------
Packets lost	2	------	0	------

Table 6.5.2 Measured session details towards node A from Node E

-	Node B-to-Node C	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	4.95 kb/s	3.69 kb/s	------	------
Average Delay	3 ms	0 ms	------	------
Average Jitter	1.1 ms	O ms	------	------
Packets lost	7	0	------	------

Table 6.5.3 Measured session details towards node C from Node B

	Node C-to-Node B	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	2.42 kb/s	2.41 kb/s	------	------
Average Delay	1 ms	0 ms	------	------
Average Jitter	0.8 ms	0 ms	------	------
Packets lost	2	0	------	-------

Table 6.5.4 Measured session details towards node B from Node C

	Node F-to-Node D	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	2.74 kb/s	------	------	2.71 kb/s
Average Delay	4 ms	------	------	0 ms
Average Jitter	2.1 ms	------	------	0 ms
Packets lost	4	------	------	0

Table 6.5.5 Measured session details towards node D from Node F

	Node D-to-Node F	Edge1-to-Edge3	Edge1-to-Edge4	Edge3-to-Edge4
Throughput	4.977 kb/s	------	------	2.69 kb/s
Average Delay	2 ms	------	------	0 ms
Average Jitter	1.1 ms	------	------	0 ms
Packets lost	25	------	------	0

Table 6.5.6 Measured session details towards node F from Node D

The values for the average jitter, packet loss and throughput could be measured for the different sessions. This approach yielded a much quicker calculation time due to the fact that double counting was eliminated and fixed routes were used to traverse the packets through the network.

6.6 Conclusion

From the results it is clear that the optimal position to extract QoS data from only TCP/IP headers over a network are at the edge routers. It is also clear that if data are extracted at the edge routers the latency of the sessions as well as the end-to-end delay of the session within the network can be monitored. The data extracted from the edge routers should however be used in conjunction with each other for the VPN session flowing through the routers to determine the sessions QoS parameters.

In chapter 3 a theoretical study showed that QoS data could be extracted from layer 2 and layer 3 within the OSI stack. In chapter 5 a methodology was developed to extract TCP/IP (Layer2 and layer 3) header data from packets within a packet switched network. Chapter 6 represented the results from the methodology developed in chapter 5 of this document. These results also showed that QoS statistics could be extracted form layer 2 and layer 3 within the network. These results supports the results found in chapter 3.

Chapter 7. Conclusion and Recommendations

Abstract – In this chapter a conclusion is drawn from the results obtained from the study. This chapter will also discuss further study fields originating from this study as well as shortcomings of this particular study.

7.1 Introduction

The objective of this study was to investigate the extent to which a communication sessions QoS parameters can be measured through only extracting TCP/IP header data. The effect on these measurements depending on the point of data extracting within the network as well as the OSI stack was also investigated.

In this chapter a summary of the conclusions from this study will be presented and final conclusions will be drawn and recommendations for future work will be made.

7.2 Summary

Installation of telecommunication networks is very expensive, therefore before such an expensive network can be installed it must be certain that the network will perform to its predefined and intended specifications. Even applications and data probes running on these networks must be reliable and fault free when they are installed. Faulty applications may cause downtime and imply financial losses to the telecommunication company.

Modeling and emulation of these networks and network applications provides a reliable and more cost effective solution to telecommunication companies worldwide than installing networks with over engineered bandwidth or debugging network and network applications in real time.

Different simulation packages exist to model these networks and protocols running on these networks. These different simulation packages have their advantages and disadvantages. In this study a network simulation package had to be used that could simulate a network as well as the protocols on these networks. The simulation package also had to be able to allow the programmer to manipulate the OSI stack within the different network apparatus.

A comparison between the different simulation packages was therefore performed and the most appropriate simulation package was chosen. The chosen simulation package was used to simulate different network topologies and extract data from within the network and from different layers of the OSI stack.

The data obtained was then used to answer the two main questions of the study.

1. How does the probe locations within the OSI layers influence the measurements?
2. How does the probe locations in the network influence the measurements?

7.3 Conclusions

7.3.1 OSI stack

Chapter three of this document showed that satisfactory information about the service and the type of quality it is receiving can be extracted from the TCP and IP headers. Other services are encapsulated in the IP and TCP headers. At the routers only these exterior IP and TCP headers can be examined, therefore only the connection setup, service type, number of packets dropped, end-to-end delay, amount of data transmitted, host and destination IP addresses and port numbers can be extracted.

It was also found that layer three gives information concerning the QoS and layer two gives information concerning the CoS. It was also determined that in layer two separation between services occurred and in layer three prioritization of these services were managed.

In conclusion layer two therefore ensures local area network performance and layer three ensures wide area network performance. Enough data can be extracted from the TCP/IP headers to monitor the sessions QoS parameters. The optimal position for the data probes will be at layer three of the OSI stack.

7.3.2 Probe placement

Chapter six of this document shows the results of probing the network at different locations for different network topologies and methods of session establishment and session maintaining.

The simple network tests showed that communication sessions' QoS parameters could be measured for a session with no packet loss as well as a session with loss. The optimal position to extract the data from was from the edge routers of the network.

These results were supported by the results from the test of the medium network, where it was shown that the sessions QoS parameters were optimally extracted from the edge routers for sessions established and maintained through using virtual private network technologies and protocols.

In conclusion the most appropriate placement of the probes are at the edge routers with the most appropriate amount of data extracted for virtual private sessions through these edge routers.

In the next section a few recommendations for future work originating from this study will be made.

7.4 Proposed solution

Agents should be installed on the routers and network elements where data are to be extracted for the use in QoS and CoS calculations. These agents then collect data from the elements and store it in the MIB (Management information base) variable file. A network management system will then pole these elements to retrieve this MIB variable information. This information will then be used to give a representation of the type of traffic flowing through these routers and the importance they are receiving. Delay and latency can be determined through using for example the proxy ping command on a CISCO router between a destination and the router transmitting the ping.

7.5 Recommendations for future work

There are some further developments that can be performed from this study. These developments may include simulating larger networks with more complex protocols as well as simulating the same tests in other simulation packages such as NS2 for example.

Further developments may include coding probes for OSI level data extraction in other simulation packages such as NS2.

References

[1] Perumalla Kalya May 2002, Georgia Institute of Technology, "Flexible, Efficient, Backplane-based network emulation", http://www.cc.gatech.edu/computing/pads/nms/

[2] Breslau Lee, Estrin Deborah, Fall Kevin May 2000, "Advances in network simulation", http://nwfusion.com/, (Web page accessed – February 2003)

[3] Telkom SA Ltd, "Request for information (Network Based IP Probe)", RFI200/02, 10/2003

[4] Pawlikowski K, Joshua Jeong H.D, Ruth Lee J.S, "ON CREDIBILITY OF SIMULATION STUDIES OF TELECOMMUNICATION NETWORKS", IEEE Communications Magazine, January 2001

[5] Stallings William, "Data & Computer Communications", 3rd Edition, Prentice Hall, 1996

[6] Peter Loshin, "TCP/IP clearly explained", 3rd Edition, Academic Press San Diego, 1992

[7] Goldman James E, "Applied Data Communications: A business Orientated Approach", Wiley: New York, 1998

[8] Shay William A, "Understanding Data Communication & Networks", 2nd Edition, Brooks/Cole, 1994

[9] "Introduction to WAN technologies", Cisco Systems, www.cisco.com, "Web page accessed – April 2002"

[10] Fred Halsall, "Data Communications, Computer Networks and OSI", Addison Wesley, 1988

[11] W. Richard Stevens, "TCP/IP illustrated", Addison Wesley, c1994-c1996

[12] Jian Zhao, Hossam Hassanein, Jieyi Wu, Guanqun Gu, "End-to-end QoS routing framework for differentiated services networks", Computer Communications, 2002

[13] S.A. Hussain, A. Marshall, "An agent based control mechanism for WFQ in IP networks, Control Engineering Practice, 2003

[14] A. Bak, W. Burakowski, F. Ricciato, S. Salsano, H. Tarasiuk, " A framework for providing differentiated QoS guarantees in IP based network", Computer Communications, Edition 26 2003, pg 327 - 337

[15] Loshin Peter, TCP/IP clearly explained, San Diego: Academic Press, 3rd edition, 1999

[16] James E Goldman, Applied data communications: a business orientated approach, 2nd Edition, John Wiley & Sons, New York, 1998

[17] Uyless D. Black, Computer networks: protocols, standards and interfaces, Englewood cliffs, N.J. Prentice Hall, c 1993

[RFC 768] Postel J, "User Datagram Protocol", 28 August 1980

[RFC 792] Postel J, "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", September 1981

[RFC 793] Postel, J, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", September 1981.

[RFC 827] Rosen Eric C, "Exterior Gateway Protocol (EGP) – DARPA Internet Program Protocol Specification", October 1982

[RFC 888] Rosen Eric C, "STUB – Exterior Gateway Protocol – BBN Communications", January 1984

[RFC 890] Postel J, "Exterior Gateway Protocol Implementation Schedule", February 1984

[RFC 904] Mills DL, "Exterior Gateway Protocol Formal Specification", April 1984

[RFC 911] Kirton P, "EGP Gateway Under UNIX 4.2", 22 August 1984

[RFC 938] Miller T, "Internet Reliable Transaction Protocol Functional and Interface Specification", February 1985

[RFC 975] Mills DL, "Autonomous Confederations", February 1986

[RFC 988] Deering SE, "Host Extensions for IP Multicasting", July 1986

[RFC 1004] Mills DL, "A Distributed Protocol – Authentication Stream", April 1987

[RFC 1009] Braden R, Postel J, "Requirements for Internet Gateways", June 1987

[RFC 1054] Deering S, "Host Extensions for IP multicasting", May 1988

[RFC 1072] Jacobson V, Braden R, "TCP Extensions for Long-Delay Paths", October 1988

[RFC 1092] Rekhter J, "EGP and Policy Based Routing in the New NSFNET Backbone", February 1989

[RFC 1112] Deering S, "Host Extensions for IP Multicasting", August 1989

[RFC 1122] Braden R, "Requirements for Internet Hosts – Communication Layers - IETF", October 1989

[RFC 1146] Zweig J, Patridge C, "TCP Alternate Checksum Options - BBN", March 1990

[RFC 1156] McCloghrie K, Rose M, "Management Information Base for Network Management of TCP/IP based internets", May 1990

[RFC 1158] Rose M, "Management Information Base for Network Management of TCP/IP based internets MIB II", May 1990

[RFC 1180] Socolofsky T, Kale C, "A TCP/IP tutorial", January 1991

[RFC 1240] Shue C, Haggerty W, Dobbins K, "OSI Connectionless Transport Services on top of UDP Version: 1", June 1991

[RFC 1323] Jacobson V, Braden R, Borman D, "TCP Extensions for High Performance", May 1992

[RFC 1347] Callon R, "TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing", June 1992

[RFC 1349] Almquist P, "Type of Service in the Internet Protocol Suite", July 1992

[RFC 1644] Braden R, "T/TCP – TCP Extensions for Transactions Functional Specification", July 1994

[RFC 1693] Connoly T, Amer P, Conrad P, "An Extension to TCP : Partial Order Service", November 1994

[RFC 1791] Sung T, "TCP Over IPX Networks With Fixed Path MTU", April 1995

[RFC 1812] Baker F, "Requirements for IP Version 4 Routers", June 1995

[RFC 1853] Simpson W, "IP in IP Tunneling", October 1995

[RFC 1940] Estrin D, Li T, Rekhter Y, Varadhan K, Zappala D, "Source Demand Routing: Packet Format and Forwarding Specification (Version 1)", May 1996

[RFC 2003] Perkins C, "IP Encapsulation within IP", October 1996

[RFC 2018] Mathis M, Mahdavi J, Floyd S, Romanow A, "TCP Selective Acknowledgement Options", October 1996

[RFC 2189] Ballardie A, "Core Based Trees (CBT version 2) Multicast Routing – Protocol Specification --", September 1997

[RFC 2201] Ballardie A, "Core Based Tress (CBT) Multicast Architecture", September 1997

[RFC 2236] Fenner W, "Internet Group Management Protocol", November 1997

[RFC 2385] Heffernan A, "Protection of BGP Sessions via the TCP Signature Option", August 1998

[RFC 2460] Deering S, Hinden R, "Internet Protocol, Version 6 (IPv6) Specification", December 1998

[RFC 2883] Floyd S, Mahdavi J, Mathis M, Podolsky M, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", July 2000

[RFC 3168] Ramakrishnan K, Floyd S, Black D, "The Addition of Explicit Congestion Notification (ECN) to IP", September 2001

[RFC 3488] Wu I, Eckert T, "Cisco Systems Router-port Group Management Protocol (RGMP)", February 2003

[18] Ondich Jeff, Carleton College, Northfield, MN 55057, "Getting started with CNET", http://www.cs.uwa.edu.au/, (Web page accessed – October 2002)

[19] Chris McDonald, "The CNET Network Simulator", http://www.cs.uwa.edu.au/cnet/, (Web page accessed – October 2002)

[20] Breslau Lee, Estrin Deborah, Fall Kevin May 2000, "Advances in network simulation", http://nwfusion.com/, (Web page accessed – February 2003)

[21] Chris McDonald, "An introduction to the CNET network simulator", http://www.cs.uwa.edu.au/, (Web page accessed – October 2002)

[22] Chung Jae, Claypool Mark, WPI Worcester Polytechnic Institute – Computer Science, "NS by Example", http://perform.wpi.edu/NS/, (Web page accessed – January 2003)

[23] Kalyan Perumalla, Georgia Tech., CoC, "Dynamic network backplane architecture project ", http://www.cc.gatech.edu/computing/pads/nms/architecture.html, (Web page accessed – January 2003)

[24] Law Kelton, "Simulation modeling and analysis", 3rd edition, Prentice Hall, April 2000

[25] K.C. Claffy, G. Miller, and K. Thompson, "The nature of the beast: recent traffic measurements from an Internet backbone," Proc. INET'98, Geneva, Switzerland, July 1998

[26] W. Jiang and H. Schulzrinne, "QoS measurement of Internet real-time multimedia services," Technical Report CUCS-015-99, Department of Computer Science, Columbia University, December 1999

[27] V. Markovski, "Simulation and Analysis of Loss in IP Networks," Simon Fraser University, 2000

[28] J.F. Ceballos-Mejia, "Design and implementation of a modeling tool for multicast networks", University of Erlangen-Nuremberg, 2000

[29] Falko Dressler, "QoS considerations on IP multicast services", Proceedings to International Conference on Advances in Infrastructure for Electronic Business, Education, Science, and Medicine on the Internet, 2002

[30] A. Hussein, A Abou-Zeid, M. Azizoglu, S. Roy, "Stochastic modeling of a single TCP/IP session over a random loss channel", "DIMACS series in discrete mathematics and theoretical computer science", 1991

[31] B. Chen, "Simulation and analysis of QoS service parameters in IP networks with real time traffic", Simon Fraser University, May 2002

[32] F. Dressler, "An approach for QoS measurements in IP multicast networks, MQM – Multicast Quality Monitor", University of Erlangen-Nuremberg, Germany

Appendix A (Network technology overview)

A.1 FDM and TDM

A.1.1 Frequency division multiplexing (FDM)

FDM is used when the useful bandwidth of the transmission medium exceeds the required bandwidth of the signals that needs to be transmitted. A multiplexer accepts the signals from multiple sources, each with a certain bandwidth. These signals are then combined into a signal with a much larger bandwidth. This signal is then transmitted over the transmission medium and de-multiplexed at the receiving end into the different signals it was composed of [5,6].

This is achieved through modulating each individual signal onto a different carrier frequency with the carrier frequencies sufficiently separated to assure that the bandwidths of the signals don't overlap. Firstly the available bandwidth of the transmission medium is divided into separate ranges or channels. Each of these channels then corresponds to one of the multiplexers input signals. Secondly a carrier signal is defined for each of these channels, and the channel signal and the carrier signal are then modulated to create a new signal. In the third step these different independent signals are combined together to create a more complex signal that is transmitted to the receiving station [5,6].

The above-mentioned method can be seen in figure A.1.1, where a number of signals $[m_i(t), i = 1, n]$ must be multiplexed onto the same transmission medium. These signals are modulated onto a carrier frequency f_i, with the resulting modulated signals summed to produce a composite baseband signal $m_b(t)$. The international FDM standards for voice transmission can be seen in table A.1.1.

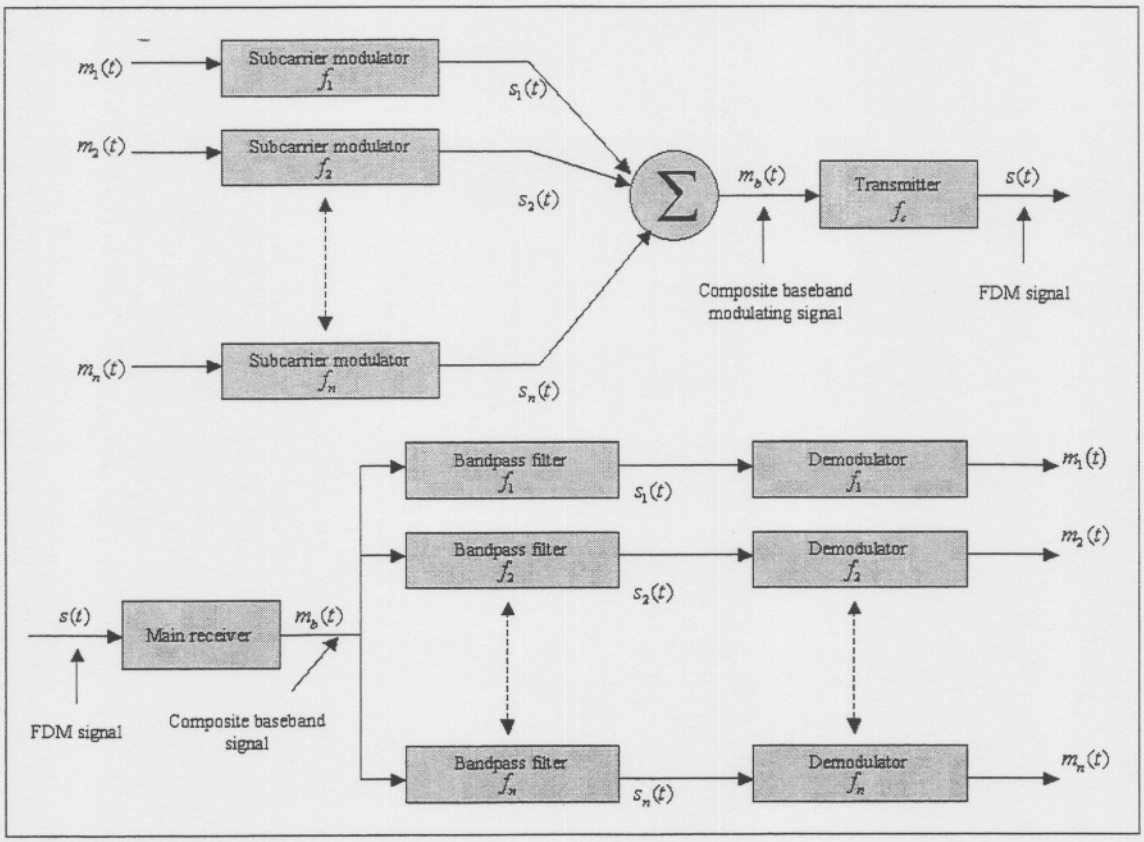

Figure A.1.1. Transmitter and receiver sections of a FDM system

Number of voice channels	Bandwidth	Spectrum	ITU-T
12	48 kHz	60-108 kHz	Group
60	240 kHz	312-552 kHz	Supergroup
300	1.232 MHz	812-2044 kHz	Mastergroup
600	2.52 MHz	564-3084 kHz	
900	3.872 MHz	8.516-12.388 MHz	Supermaster group
3600	16.984 MHz	0.564-17.548 MHz	
10800	57.443 MHz	3.124-60.566 MHz	

Table A.1.1 International FDM carrier standards

A.1.2 Time division multiplexing (TDM)

TDM is used when the achievable data rate of the medium exceeds the data rate of digital signals that needs to be transmitted. Multiple digital signals can be transmitted on a transmission medium through interleaving portions of each time signal in time. This method can be seen in figure A.1.2 where a number of signals $[m_i(t), i = 1, n]$ must be multiplexed onto the same transmission medium. The incoming data are briefly buffered; with each buffer typically one bit or one character in length. The

buffers are then scanned sequentially to form a composite digital data stream $m_c(t)$. The data rate of $m_c(t)$ must at least equal sum of the data rates of $m_i(t)$. The international TDM standards for voice transmission can be seen in table A.1.2.

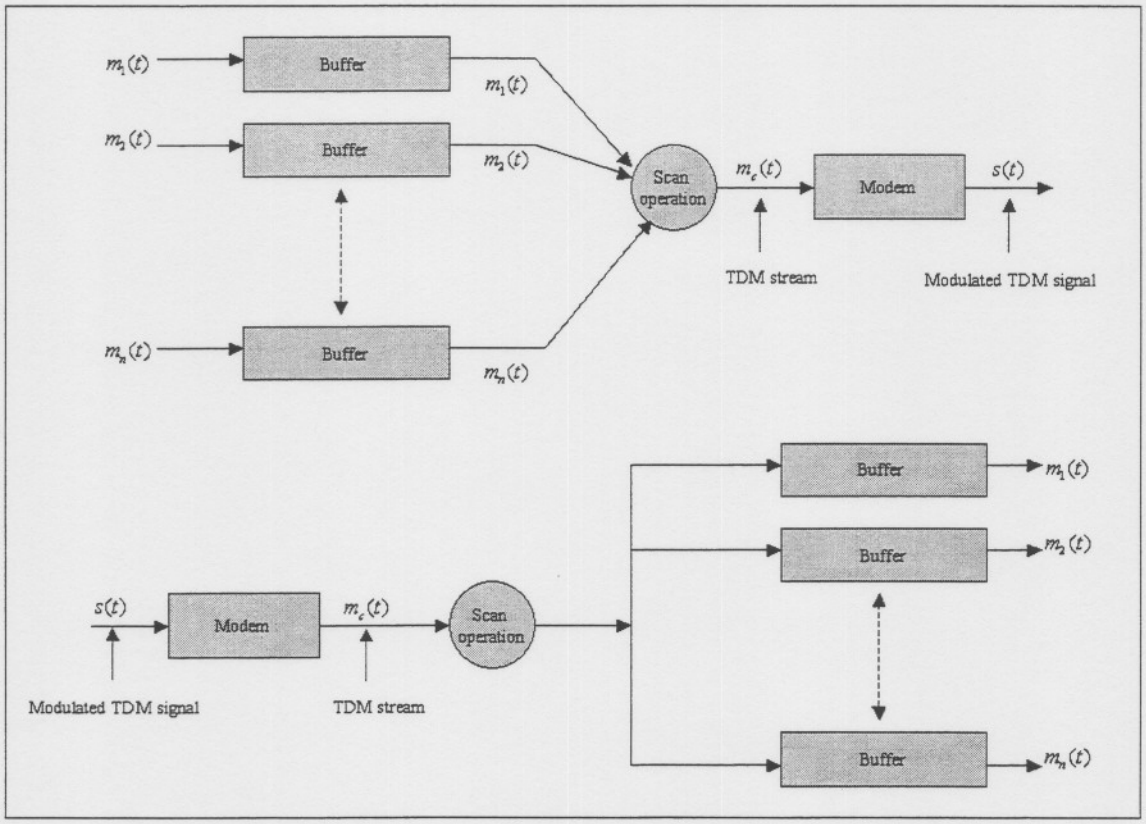

Figure A.1.2. Transmitter and receiver sections of a typical TDM section

Level	Number of voice channels	Data rate (Mbps)
1	30	2.048
2	120	8.448
3	480	34.368
4	1920	139.264
5	7680	565.148

Table A.1.2. International TDM standards

A.2 Long distances calls over a circuit switched network

When a long distance call is made the through a circuit switched network a path must normally be routed through more than one switch and trunk. The strategy when developing such a network however is to use as little as possible equipment and to still provide enough efficiency to handle more than the expected load.

Originally this was performed through the use of a tree structured routing strategy. If a call was made, a path was constructed by starting at the calling subscriber, tracing up the tree to the first common node, and then tracing down to the tree to the called subscriber. This approach was not very dynamic and could not handle and adapt to changing conditions. A dynamic approach is an approach that can adapt to current traffic conditions and can make decisions based on the current status of the network.

A method of creating a dynamic routing system is to use multiple pre-defined routes, enabling the switch to choose the desired route according to the current status of the network. Different sets of routes may also be used at different times of the day or week making the system even more dynamic and increasing efficiency. For example in table A.2.1 the different routes can be seen for figure A.2.1.

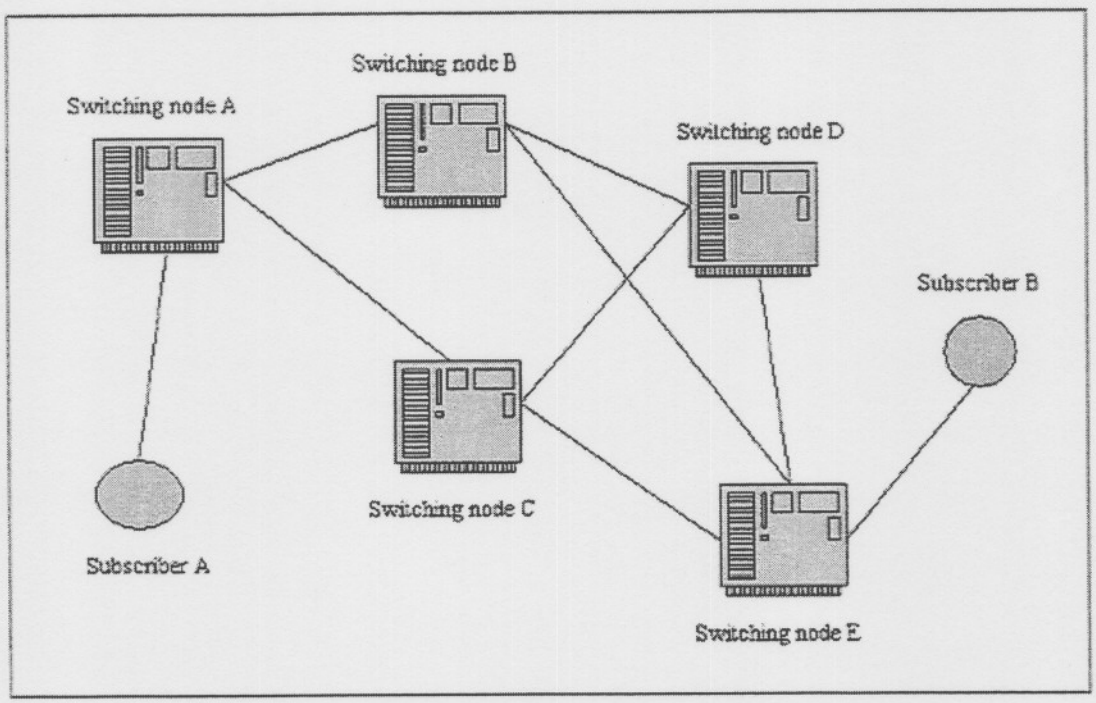

Figure A.2.1 Switched network topology

Time of day	First desired route	Second desired route	Third desired route	Fourth desired route
8 am – 8 pm	A – C – E	A – B - E	A – B – D - E	A – C – D – E
8 pm – 8 am	A – C – E	A – C – D – E	A – B – D - E	A –B – E
Weekends	A – C – E	A – B – D – E	A – B - E	A – C – D – E
Public holidays	A – C – E	A – B - E	A – C – D - E	A – B – D - E

Table A.2.1 Routing table for figure A.2.1

If a switch is broken the next switch will be tried in the order that they are listed within the table and according to the time of day or week. When a such connection are created control signals must be used, these control signals can be subdivided into four categories with each independent category contributing to a certain part within the establishment of the connection.

The first category is the supervisory category, which provides the mechanism for obtaining the resources for the connection. The second category is the address category, which provides the mechanisms for identifying the subscribers participating in the connection. The third category provides the mechanisms for call information and fourthly the last category is the network management category, which provides status signals.

These signals could be passed on the same channel used for the transmission or it could be passed on a separate channel between nodes for control signaling. The latter is called common channel signaling and the other method is called inchannel signaling. When an inchannel approach is taken the control signals can be transmitted in a different frequency range than the data/voice signal (Out of band signaling) or on the same frequency as the data/voice (Inband signaling). Common channel signaling is more powerful than inchannel signaling due to the fact that inchannel signaling has a higher connection establishment time as well as limited information transfer rate [5,6].

The most used common channel-signaling standard is the Signaling System Number 7 (SS7) scheme. The purpose of SS7 is to provide an internationally standardized, general-purpose common channel signaling system with the following primary characteristics.

- SS7 is optimized for use in digital telecommunication networks in conjunction with digital stored program-control exchanges, utilizing 64 kbps digital channels.
- SS7 is designed to meet present and future information transfer requirements for call control, remote control, management and maintenance as well as to be a reliable means for the transfer of information in the correct sequence without loss or duplication.
- SS7 is suitable for operation over analog channels and at speeds below 64 kbps and is also used on point-to-point terrestrial and satellite links [5].

SS7 is therefore a four-layer protocol (figure A.2.2) with the bottom three layers constituting to message transfer and the top layer constituting to call control, message formats and maintenance.

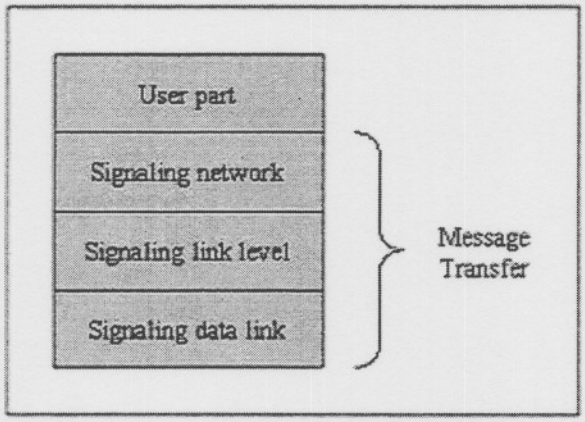

Figure A.2.2. Four layers of SS7

The lowest layer (Signaling data link layer) provides all the physical and electrical specifications and provides a 64 kbps full duplex transmission. The signaling link layer provides reliable communications between two consecutive points within the network and the signaling network layer provides reliable message transfer between these two signaling end points [6].

A.3 Packet switched network routing techniques

There are different routing techniques used when data are transmitted through a packet switched network. These routing techniques include fixed routing, flooding, random routing and adaptive routing. In a fixed routing topology a single permanent route is configured for each source-destination pair of nodes within the network. Fixed routing could be implemented through the use of routing tables or matrixes for

each source destination pair. The routing tables for an example network such as in figure A.3.1 can be seen in tables A.3.1 to A.3.4.

As an example bridge B2 contains routing tables for both LAN 1 and LAN 2. Thus if the bridge receives a frame from LAN 1 it determines the destination address and looks up the appropriate route in the table for L1. If the destination address is B through F the bridge forwards the frame to L2. If the message is destined for A the bridge will keep the frame in L1. The same applies for bridge 3, with the bridge forwarding any frame from L2 to L3 that is destined for C, D or F. But in the event of the frame destined for A, B or E the bridge will not forward the frame. This approach of routing packets through the network is referred to as fixed routing [5,6].

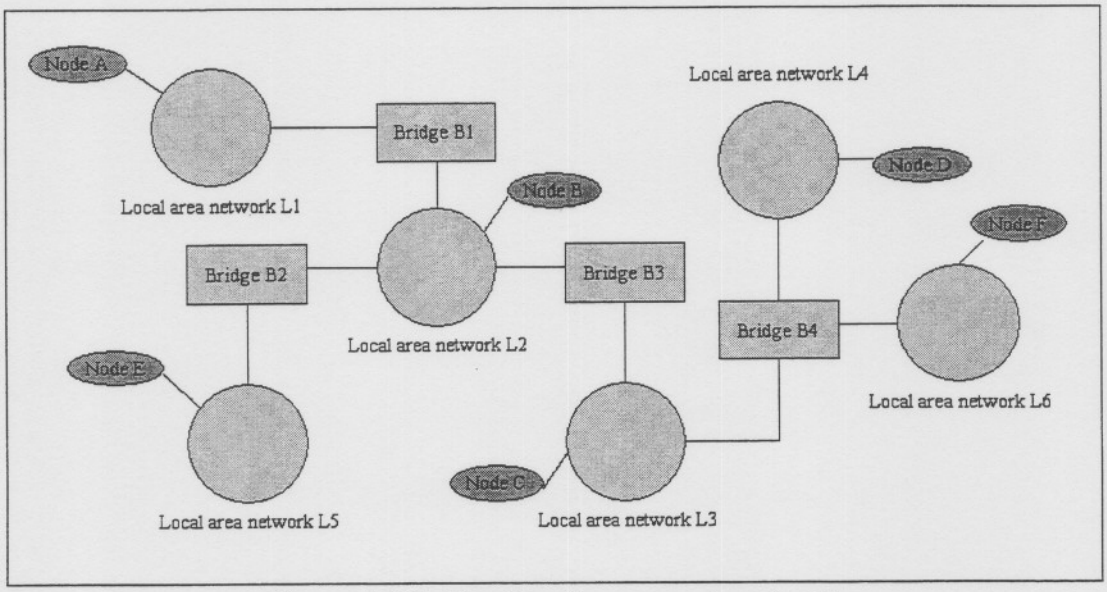

Figure A.3.1. Fixed routing example network

Source LAN 1		Source LAN 2	
Destination	**Next LAN**	**Destination**	**Next LAN**
A	--	A	L1
B	L2	B	--
C	L2	C	--
D	L2	D	--
E	L2	E	--
F	L2	F	--

Table A.3.1. Routing table for bridge B1

Source LAN 2		Source LAN 5	
Destination	Next LAN	Destination	Next LAN
A	--	A	L2
B	--	B	L2
C	--	C	L2
D	--	D	L2
E	L5	E	--
F	--	F	L2

Table A.3.2. Routing table for bridge B2

Source LAN 2		Source LAN 3	
Destination	Next LAN	Destination	Next LAN
A	--	A	L2
B	--	B	L2
C	L3	C	--
D	L3	D	--
E	--	E	L2
F	L3	F	--

Table A.3.3. Routing table for bridge B3

Source LAN L3		Source LAN L4		Source LAN L6	
Destination	Next LAN	Destination	Next LAN	Destination	Next LAN
A	--	A	L3	A	L3
B	--	B	L3	B	L3
C	--	C	L3	C	L3
D	L4	D	--	D	L4
E	--	E	L3	E	L3
F	L6	F	L6	F	--

Table A.3.4. Routing table for bridge B4

The next approach to routing is flooding, and works on the principle that the frame is transmitted by the receiving node to all of its neighbors for distribution. This kind of routing entails some remarkable properties. All routes between the source and the destination are tried, compensating for a broken link. Because all of these links are tried, at least one of the packets will arrive at the intended destination with a minimum hop rate.

This service can therefore be used to transmit emergency messages because this service is very robust, and all nodes on the network are directly or indirectly contacted. This type of routing may also be used to set up a virtual circuit for a transmission that requires low jitter and delay for example. The disadvantage of this

approach however is the fact that it generates a high traffic load, which is directly proportional to the connectivity of the network.

To ensure that the frames don't stay on the network to long, a maximum hop count could be implemented and the packet could be discarded if this count reaches zero. The receiving node must also discard multiple copies of the same frame. An example of this approach can be seen in figure A.3.2 where Node A transmits a frame to all of its neighbors and its neighbors then transmits the same frame to all of their neighbors and so forth until the frame arrives at Node F. The frames will be discarded after a certain amount of hops [5,6].

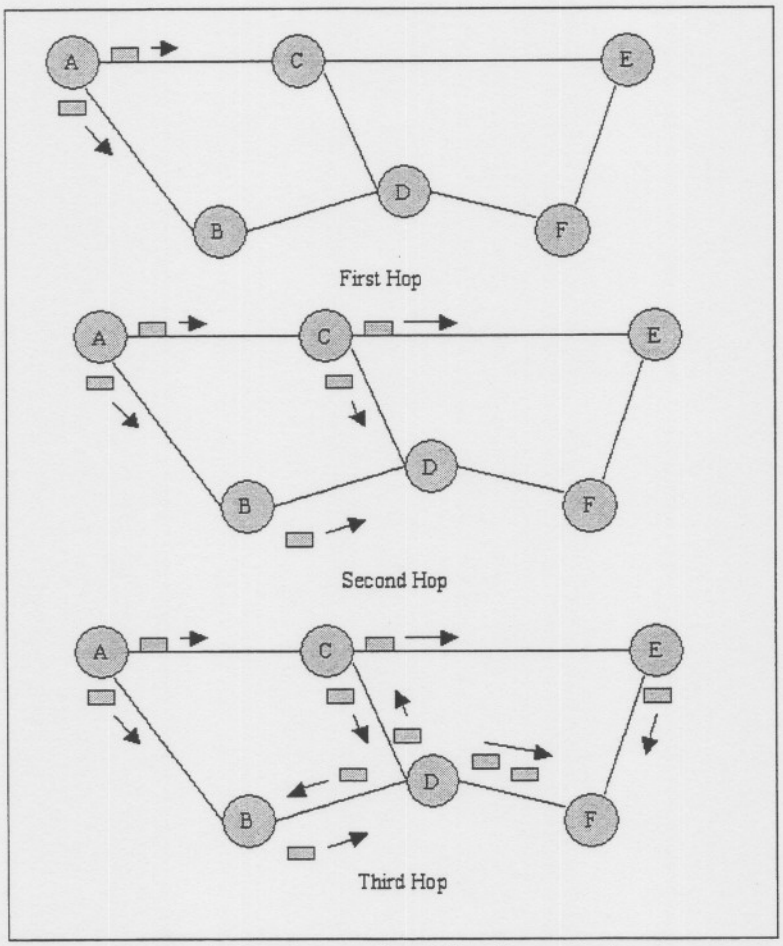

Figure A.3.2. Example of flooding a network

The next routing approach that will be discussed is random routing and works on the principle that each node only selects one outgoing path for every frame. This approach has the robustness of flooding but decreases the traffic load the network is subjected to. The outgoing link is chosen at random excluding the incoming link. This technique could be enhanced through assigning a probability to each link and to then

choose a link on its probability value. This type of routing also subjects the network to a high traffic load, but not as high as flooding.

A much better approach to routing will be to route the packets in accordance to network changes. This means that packets will be routed to different nodes depending on the current conditions of the network. These conditions may include failure of certain links as well as congestion on others. The routing should then be adapted to these circumstances. The methods used to route packets on the above mentioned criteria are called adaptive routing. For this approach to work, the different nodes within the network must exchange information, thus the processing burden of each node increases.

This approach however is much more efficient than flooding or random routing because it improves network performance as seen by the network user and it aids in congestion control. The node may route the incoming packet to the outgoing link with the shortest queue. This can be improved through transmitting the packet into the right direction as well. Thus if we represent the Queue length with Q and each link's bias with B the best transmission would be if Q + B could be minimized [5,6].

A.4 IP addressing

As with any other network-layer protocol, the IP addressing scheme is integral to the process of routing IP datagrams through an inter-network. Each IP address has specific components and follows a basic format. These IP addresses can be subdivided and used to create addresses for sub-networks. Each host on a TCP/IP network is assigned a unique 32-bit logical address that is divided into two main parts, the network number and the host number.

The network number identifies a network and must be assigned by the Internet Network Information Center (InterNIC) if the network is to be part of the Internet. An Internet Service Provider (ISP) can obtain blocks of network addresses from the InterNIC and can itself assign address space as necessary. The host number identifies a host on a network and is assigned by the local network administrator.

The 32-bit IP address is grouped eight bits at a time, separated by dots, and represented in decimal format (known as *dotted decimal notation*). Each bit in the octet has a binary weight (128, 64, 32, 16, 8, 4, 2, 1). The minimum value for an octet

is 0, and the maximum value for an octet is 255. The construction of such an IP address can be seen in figure A.4.1.

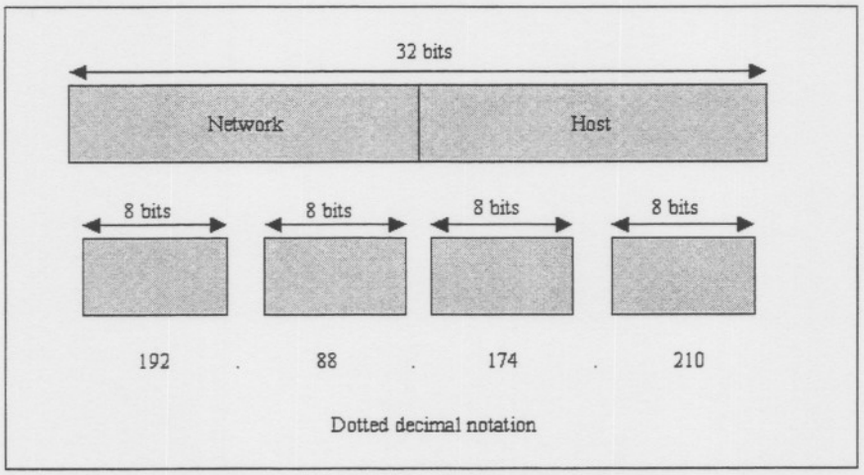

Figure A.4.1. IP address construction

IP addressing supports five different address classes: A, B, C, D, and E. Only classes A, B, and C are available for commercial use. Table A.4.1 gives some reference information about the five different IP address classes. Under the Format column N represents the network number and H represents the host number. Figure A.4.2 illustrates the construction of the commercial IP addresses.

IP address class	Format	Purpose	High Order Bits	Address Range	Network/Host number of bits	Maximum hosts
A	N.H.H.H	Few large organizations	0	1.0.0.0 – 126.0.0.0	7/24	$16777214(2^{24}-2)$
B	N.N.H.H	Medium size organizations	1,0	128.1.0.0 – 191.254.0.0	14/16	$65543(2^{16}-2)$
C	N.N.N.H	Relatively small organizations	1,1,0	192.0.1.0 – 223.255.254.0	22/8	$245(2^{8}-2)$
D	N/A	Multicast groups	1,1,1,0	224.0.0.0 – 239.255.255.255	N/A not for commercial use	N/A
E	N/A	Experimental	1,1,1,1	240.0.0.0 – 254.255.255.255	N/A	N/A

Table A.4.1. Reference information about the five different IP address classes

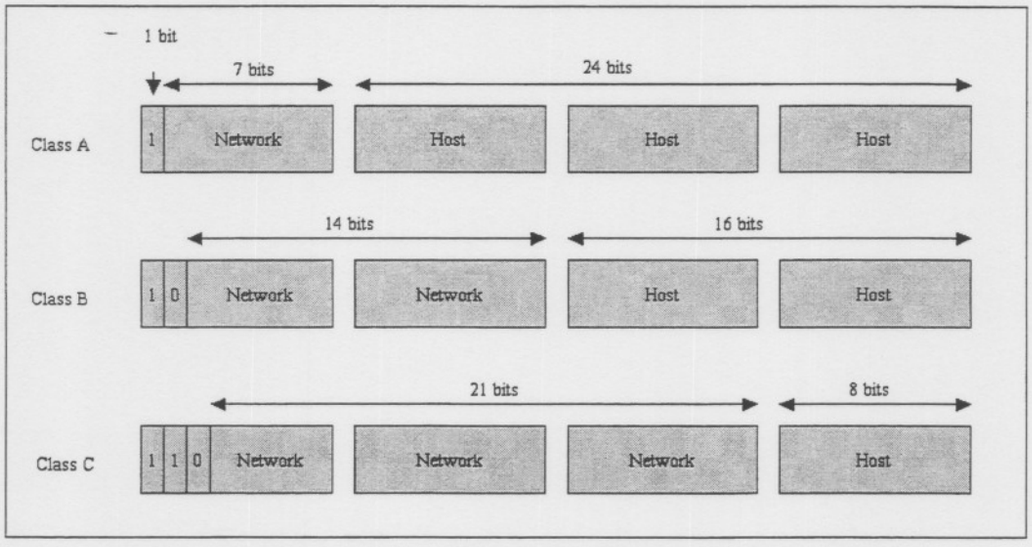

Figure A.4.2. Construction of the different IP addresses

The class of address can be determined easily by examining the first octet of the address and mapping that value to a class range in the following table. In an IP address of 172.31.1.2, for example, the first octet is 172. Because 172 falls between 128 and 191, 172.31.1.2 is a Class B address. Table A.4.2 summarizes the range of possible values for the first octet of each address class.

Address Class	First Octet in Decimal	High Order Bits
Class A	1 Ð 126	0
Class B	128 Ð 191	1 0
Class C	192 Ð 223	1 1 0
Class D	224 Ð 239	1 1 1 0
Class E	240 Ð 254	1 1 1 1

Table A.4.2. A range of possible values for the first octet of each address class

A.5 IP subnet addressing

IP networks can be divided into smaller networks called sub-networks (or subnets). Sub-netting provides the network administrator with several benefits, including extra flexibility, more efficient use of network addresses, and the capability to contain broadcast traffic (a broadcast will not cross a router). Subnets are under local administration. As such, the outside world sees an organization as a single network and has no detailed knowledge of the organization's internal structure.

A given network address can be broken up into many sub-networks. For example, 172.16.1.0, 172.16.2.0, 172.16.3.0, and 172.16.4.0 are all subnets within the network

171.16.0.0. (All 0's in the host portion of an address specifies the entire network.) An address is created by "borrowing" bits from the host field and designating them as the subnet field. The number of borrowed bits varies and is specified by the subnet mask. Figure A.5.1 shows how bits are borrowed from the host address field to create the subnet address field.

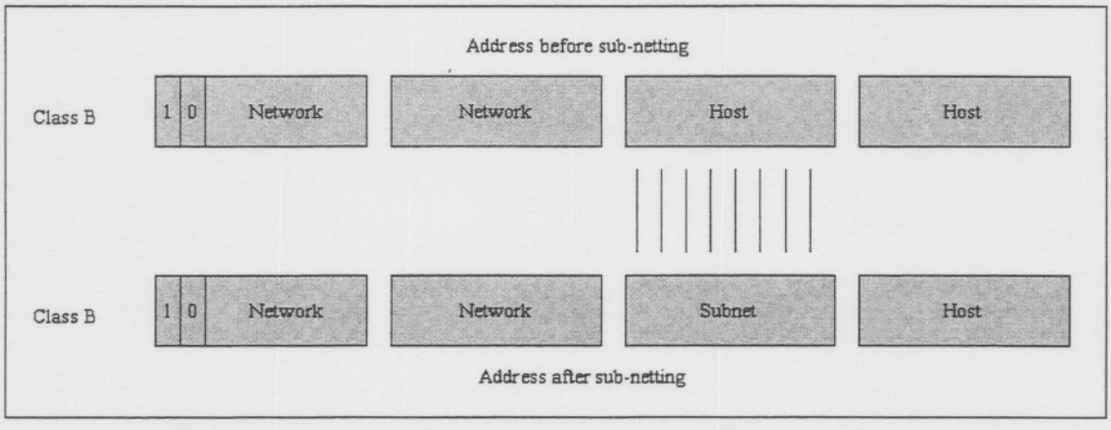

Figure A.5.1. Class B address before and after sub-netting

Subnet masks use the same format and representation technique as IP addresses. The subnet mask, however, has binary 1's in all bits specifying the network and sub-network fields, and binary 0's in all bits specifying the host field. Figure A.5.2 illustrates a sample subnet mask [5,6,11].

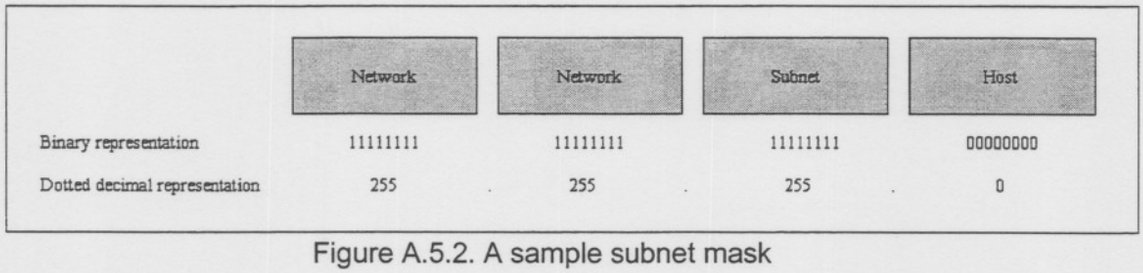

Figure A.5.2. A sample subnet mask

The default subnet mask for a Class B address that has no sub-netting is 255.255.0.0, while the subnet mask for a Class B address 171.16.0.0 that specifies eight bits of sub-netting is 255.255.255.0. The reason for this is that eight bits of sub-netting or 28 – 2 (1 for the network address and 1 for the broadcast address) = 254 subnets possible, with 28 – 2 = 254 hosts per subnet.

The subnet mask for a Class C address 192.168.2.0 that specifies five bits of sub-netting is 255.255.255.248.With five bits available for sub-netting, 25 – 2 = 30 subnets possible, with 23 – 2 = 6 hosts per subnet. The reference charts shown in

table A.5.1 and table A.5.2 can be used when planning Class B and C networks to determine the required number of subnets and hosts, and the appropriate subnet mask.

Number of bits	Subnet Mask	Number of subnets	Number of hosts
2	255.255.192.0	2	16832
3	255.255.224.0	6	8190
4	255.255.240.0	14	4094
5	255.255.248.0	30	2046
6	255.255.252.0	62	1022
7	255.255.254.0	126	510
8	255.255.255.0	254	254
9	255.255.255.128	510	126
10	255.255.255.192	1022	62
11	255.255.255.224	2046	30
12	255.255.255.240	4094	14
13	255.255.255.248	8190	6
14	255.255.255.252	16382	2

Table A.5.1. Class B sub-netting reference table

Number of bits	Subnet Mask	Number of subnets	Number of hosts
2	255.255.255.192	2	62
3	255.255.255.224	6	30
4	255.255.255.240	14	14
5	255.255.255.248	30	6
6	255.255.255.252	62	2

Table A.5.2. Class C sub-netting reference table

The router performs a set process to determine the network (or more specifically, the sub-network) address. First, the router extracts the IP destination address from the incoming packet and retrieves the internal subnet mask. It then performs a logical AND operation to obtain the network number. This causes the host portion of the IP destination address to be removed, while the destination network number remains. The router then looks up the destination network number and matches it with an outgoing interface. Finally, it forwards the frame to the destination IP address.

For two machines on a given network to communicate, they must know the other machine's physical (orMAC) addresses. By broadcasting Address Resolution Protocols (ARP's), a host can dynamically discover the MAC-layer address corresponding to a particular IP network-layer address. After receiving a MAC-layer address, IP devices create an ARP cache to store the recently acquired IP-to-MAC

address mapping, thus avoiding having to broadcast ARPS when they want to re-contact a device. If the device does not respond within a specified time frame, the cache entry is flushed. In addition to the Reverse Address Resolution Protocol (RARP) is used to map MAC-layer addresses to IP addresses. RARP, which is the logical inverse of ARP, might be used by diskless workstations that do not know their IP addresses when they boot. RARP relies on the presence of a RARP server with table entries of MAC-layer-to-IP address mappings [5,6,11].

A.6 Internet routing

Internet routing devices traditionally have been called gateways. In today's terminology, however the term gateway refers specifically to a device that performs application-layer protocol translation between devices. Interior gateways refer to devices that perform these protocol functions between machines or networks under the same administrative control or authority, such as a corporation's internal network. These are known as autonomous systems.

Exterior gateways perform protocol functions between independent networks. Routers within the Internet are organized hierarchically. Routers used for information exchange within autonomous systems are called interior routers, which use a variety of Interior Gateway Protocols (IGP's) to accomplish this purpose.

The Routing Information Protocol (RIP) is an example of an IGP. Routers that move information between autonomous systems are called exterior routers. These routers use an exterior gateway protocol to exchange information between autonomous systems. The Border Gateway Protocol (BGP) is an example of an exterior gateway protocol.

IP routing protocols are dynamic implementing dynamic routing calls for routes to be calculated automatically at regular intervals by software in routing devices. This contrasts with static routing, where routes are established by the network administrator and do not change until the network administrator changes them. An IP routing table, which consists of destination address/next hop pairs, is used to enable dynamic routing. An entry in this table, for example, would be interpreted as follows; to get to network 172.31.0.0, send the packet out Ethernet interface 0 (E0).

IP routing specifies that IP datagrams travel through inter-networks one hop at a time. The entire route is not known at the onset of the journey, however. Instead, at each stop, the next destination is calculated by matching the destination address within the datagram with an entry in the current node's routing table.

Each node's involvement in the routing process is limited to forwarding packets based on internal information. The nodes do not monitor whether the packets get to their final destination, nor does IP provide for error reporting back to the source when routing anomalies occur. This task is left to another Internet protocol, the Internet Control-Message Protocol (ICMP) [11]. A detailed description of the ICMP protocol can be seen in appendix A of this document.

A.7 TCP Congestion Control

As network or Internet conditions change, a static retransmission time is likely to be either too long or too short. Accordingly, virtually all TCP implementations attempt to estimate the current round trip delay by observing the pattern of delay for recent segments, and then set the timer to a value somewhat greater than the estimated round trip delay.

There exists different methods to calculate the round trip delay which includes a simple average approach, exponential average approach and RTT variance estimation for example. In a simple average approach the average of the observed round trip delays are taken over a number of segments. This method can be expressed with equation A.7.1 [5,6].

$$ARTT(K+1) = \frac{1}{K+1}\sum_{i=1}^{K+1} RTT(i) \quad (A.7.1)$$

RTT(i) = Round trip time observed for the *i*th transmitted segment.

ARTT(K) = Average round trip time for the first K segments.

In an exponential approach more weight within the calculation is given to more recent delay values. This method can be expressed with equation A.7.2.

$$SRTT(K+1) = \alpha \cdot SRRT(K) + (1-\alpha) \cdot RTT(K+1) \quad (A.7.2)$$

SRTT(K) = Smoothed round trip time estimate, with SRTT(0) = 0.

The above equation thus predicts the next value on the basis of a time series of past values. A value for α smaller than one is chosen, this gives the advantage that the average will quickly reflect a rapid change in the observed quantity. The disadvantage however is a jerky average value for quick fluctuations in the delay times that settles back to a relatively constant value.

In cases where the round trip time exhibits a relatively high variance the RTT variance estimation (Jacobson's Algorithm) can be used. Three points of high variance can be identified.

1. If the data rate on the TCP connection is relatively low, then the transmission delay will be relatively large compared to propagation time and the variance in delay due to variance in IP datagram size will be significant. Thus, the SRTT estimator is heavily influenced by characteristics that are property of the data and not of the network.
2. Internet traffic load and conditions may change abruptly due to traffic from other sources, causing abrupt changes in RTT.
3. The peer TCP entity may not acknowledge each segment immediately because of its own processing delays and because it exercises its privilege to use cumulative acknowledgements.

The complete algorithm proposed by Jacobson can be expressed as follows.

$$
\begin{aligned}
&SRTT(K+1) = (1-g)\cdot SRTT(K) + g\cdot RTT(K+1)\\
&SERR(K+1) = RTT(K+1) - SRTT(K)\\
&SDEV(K+1) = (1-h)\cdot SDEV(K) + h\cdot |SERR(K+1)| \quad (A.7.3)\\
&RTO(K+1) = SRTT(K+1) + f\cdot SDEV(K+1)
\end{aligned}
$$

RTO(K) = retransmission timer.

When this approach is used two other factors must be kept in mind. Firstly, what RTO value must be used on a retransmitted segment? Which can be solved through the use of the RTO back-off algorithm. The RTO back-off algorithm dictates that the TCP source must increase its RTO each time the same segment is retransmitted. A simple technique for implementing RTO back off is to multiply the RTO of the segment with a constant value for each transmission [5,6].

And secondly which round trip samples should be used as input to the Jacobson's algorithm, which can be solved through the use of Karn's algorithm. If a segment times-out and the segment has to be re-transmitted and an acknowledgement is subsequently received, there are two possibilities.

1. Firstly, this is the ACK of the first transmitted segment.
2. Secondly, this is the ACK of the second transmitted segment.

The problem however is how to determine which ACK has been received and which value to use. If the first scenario is true and the value is used along with the Jacobson's algorithm it will yield an unnecessary high value for SRTT. Using the RTT from the second scenario will cause the RTO and SRTT values to be too small. Karn's algorithm however solves this scenario with the use of the following rules.

1. Firstly, the measured RTT for a retransmitted segment should never be used to update the SRTT and SDEV values.
2. Secondly, The back-off RTO value must be calculated with the use of the exponential RTO back-off method.
3. And thirdly, use the back-off RTO value for succeeding segments until an acknowledgement arrives for a segment that has not been re-transmitted.

Appendix B (IP level protocol discussion)

B.1 ICMP (Internet control message protocol)

The Internet protocol is used for host-to-host datagram services in a network, which is constructed from smaller networks and so forth. These networks are connected to each other with gateways. These gateways communicate with each other through the GGP (gateway-to-gateway protocol). If a gateway communicates with a host the ICMP is used. ICMP therefore operates as a higher-level protocol using IP.

ICMP are utilized in many situations, which includes for example when a datagram cannot reach its destination or when the gateway does not have the buffering capacity and must drop a datagram. Another use is in the scenario where the gateway can direct the host to transmit the datagrams via a shorter route. The main purpose of these control messages is therefore to provide feedback of the communication environment.

ICMP therefore provides error reporting, flow control and first hop-gateway direction for example. ICMP can however not be used to make IP more reliable because there are still no guarantee that a datagram will be delivered or that a control message will be returned. Reliability is therefore left to higher layer protocols, which use IP for communications. An ICMP header can be seen in figure B.1.1 The header is 32 bits in total length with an 8-bit Type field, 8-bit code field and a 16-bit ICMP header checksum field.

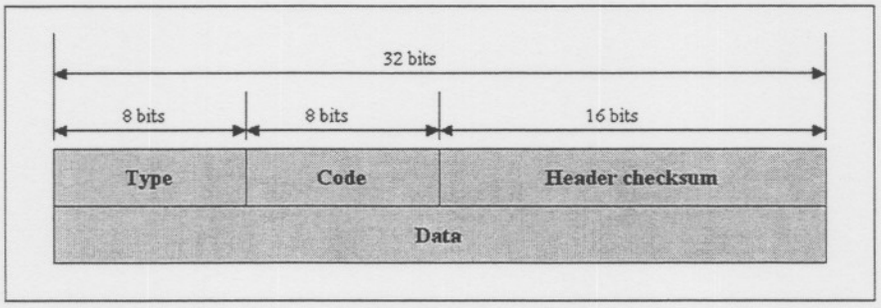

Figure B.1.1 ICMP header construction

The type field specifies the format of the ICMP message, with the code field extending the type field specification. These fields are both 8-bits in total length, meaning that they can both contain 256 values. The checksum field holds the ICMP header checksum, which is a one's compliment of the one's compliment sum of the

ICMP message starting with the type field. The information contained within the type field can be seen in table B.1.1 [RFC 792].

Type	Description	Type	Description
0	Echo reply message	17	Address mask request message
1	Reserved	18	Address mask reply message
2	Reserved	19	Reserved for security
3	Destination unreachable message	20-29	Reserved for robustness experiments
4	Source quench message	30	Trace-route message
5	Redirect message	31	Conversion error message
6	Alternate host address message	32	Mobile host redirect message
7	-	33	IPv6 Where-are-you message
8	Echo request message	34	IPv6 I-am-here message
9	Router advertisement message	35	Mobile registration request message
10	Router solicitation message	36	Mobile registration reply message
11	Time exceeded message	37	Domain name request message
12	Parameter problem message	38	Domain name reply message
13	Timestamp request message	39	SKIP algorithm discovery protocol
14	Timestamp reply message	40	Photuris, security failures message
15	Information request message	41-255	Reserved
16	Information reply message		

Table B.1.1 ICMP type field contents

B.2 ICMP (Internet Control Message Protocol)

ICMP is documented in the [RFC 792] document and is a required protocol tightly integrated with IP. ICMP messages, delivered in IP packets, are used for out-of-band messages related to network operation or miss-operation. Of course, since ICMP uses IP, ICMP packet delivery is unreliable, so hosts can't count on receiving ICMP packets for any network problem.

ICMP's generate several kinds of useful messages, including Destination Unreachable, Echo Request and Reply, Redirect, Time Exceeded, and Router Advertisement and Router Solicitation. If an ICMP message cannot be delivered, no second one is generated. This is to avoid an endless flood of ICMP messages when a router sends an ICMP destination-unreachable message. An ICMP unreachable message means that the router is unable to send the package to its final destination. The router then discards the original packet. There can be two reasons why a

When sending a multicast IP datagram, a host transmits it to a local network multicast address which identifies all neighboring members of the destination host group. If the group has members on other networks, a multicast agent becomes an additional recipient of the local multicast and relays the datagram to agents on each of those other networks, via the Internet gateway system. Finally, the agents on the other networks each transmit the datagram as a local multicast to their own neighboring members of the destination group [RFC 988].

In a level 2 implementation full support for IP multicasting allows a host to create, join and leave host groups, as well as transmit IP datagrams to host groups. It requires implementation of IGMP and extensions of the IP and local network service interfaces within the host. Figure B.3.1 is an illustration of the IGMP header, which is 128 bits in total length. The 32 bits include an 8-bit type field, an 8-bit code field, a 16-bit header checksum field, a 32-bit identifier field, a 32-bit group header field and a 32-bit access key field.

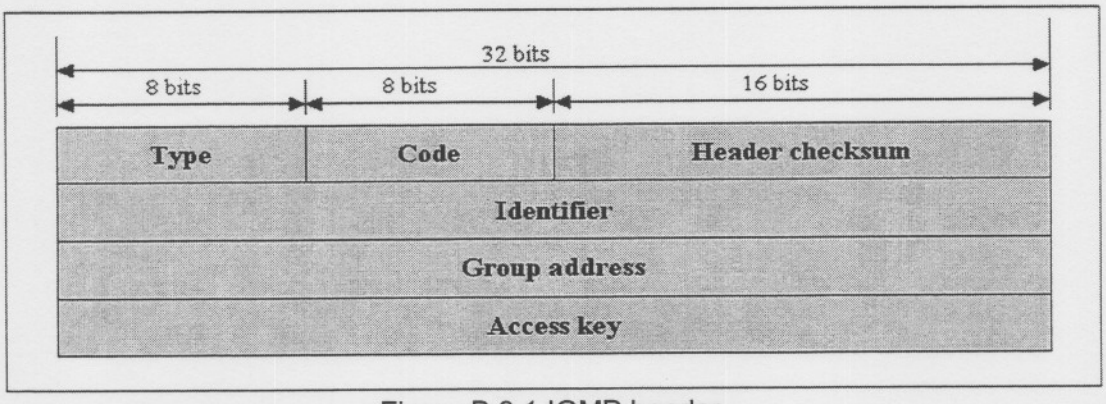

Figure B.3.1 IGMP header

The information contained in the type field can be seen in table B.3.1.

Type	Description
1	Create group request
2	Create group reply
3	Join group request
4	Join group reply
5	Leave group request
6	Leave group reply
7	Confirm group request
8	Confirm group reply

Table B.3.1 IGMP type field contents

In a create group request this field will indicate if the new host group is public or private. In all the other request messages this field will be set to zero. A zero field value indicates that the new host group is public and if the field value is one the new host group is private. If the message is a reply message the code field will indicate the outcome of the request. The values of the code field for a reply message can be seen in table B.3.2.

Code	Description
0	Request granted
1	Request denied, no resources
2	Request denied, invalid code
3	Request denied, invalid group address
4	Request denied, invalid access key
5-255	Request pending, retry in this many seconds

Table B.3.2 Code field for a reply message scenario

The checksum field contains the 16-bit one's compliment of the one's compliment sum of the IGMP message starting with the type field. The identifier field contains zero for a confirm group request message. In all other request messages the field contains a value to distinguish the request from other requests by the same host. In a reply message the field contains the same value as the corresponding request message.

In a create group request message, the group address field contains zero. In all other request messages, the group address field contains a host group address. In a create group reply message, the group address field contains either a newly allocated host group address (if the request is granted) or zero (if denied). In all other reply messages, the group address field contains the same host group address as in the corresponding request message. In a create group request message, the access key field contains zero.

In all other request messages, the access key field contains the access key assigned to the host group identified in the group address field (zero for public groups). In a create group reply message, the access key field contains either a non-zero 64-bit number (if the request for a private group is granted) or zero. In all other reply

messages, the access key field contains the same access key as in the corresponding request.

B.4 IGMP version1 and version 2

As previously mentioned, IGMP is an integral part of IP and it is required to be implemented by all hosts conforming to level 2 of the IP multicasting specification. IGMP messages are encapsulated within IP datagrams with an IP protocol number 2. Multicast routers transmit host membership query messages to discover which host groups have members on their attached local networks.

These queries are addressed to the all host address group (224.0.0.1) and carry an IP time-to-live of 1. A host responds to a query by generating host membership reports, which reports each host's group to which they belong on the network interface from which the query was received. Two techniques are used to avoid implosion of concurrent reports as well as to reduce the total amount of reports that are transmitted.

- Firstly, when a host receives a query, rather than sending a report immediately, it starts a report delay timer for each of its group memberships on the network interface of the incoming query. Each timer is set to a different, randomly chosen value between zero and D seconds. When a timer expires, a report is generated for the corresponding host group. Thus, reports are spread out over a D second interval instead of all occurring at once.
- Secondly, a report is transmitted with an IP destination address equal to the host group address being reported, and with an IP time-to-live of 1, so that other members of the same group on the same network can overhear the report. If a host hears a report for a group to which it belongs on that network, the host stops its own timer for that group and does not generate a report for that group. Thus, in the normal case, only one report will be generated for each group present on the network, by the member host whose delay timer expires first. Note that the multicast routers receive all IP multicast datagrams, and therefore need not be addressed explicitly. It must also be remembered that the routers don't have to know which hosts belong to a group, only that at least one host belongs to a group on a particular network.

There are two exceptions to the behavior described above. Firstly, if a report delay timer is already running for a group membership when a query is received, that timer is not reset to a new random value, but rather allowed to continue running with its current value. Secondly, a report delay timer is never set for a host's membership in the all- hosts group (224.0.0.1), and that membership is never reported. If a host uses a pseudo-random number generator to compute the reporting delays, one of the host's own individual IP addresses should be used as part of the seed for the generator. This is done to reduce the chance of multiple hosts generating the same sequence of delays.

A host should confirm that a received report has the same IP host group address in its IP destination field and its IGMP group address field, to ensure that the host's own report is not cancelled by an erroneous received report. A host should quietly discard any IGMP message of type other than host membership query or host membership report. Multicast routers transmit queries periodically to refresh their knowledge of memberships present on a particular network. If no reports are received for a particular group after some number of queries, the routers assume that that group has no local members and that they need not forward remotely originated multicasts for that group onto the local network.

Queries are normally transmitted infrequently (no more than once a minute) so as to keep the IGMP overhead on hosts and networks very low. However, when a multicast router starts up, it may issue several closely-space queries in order to quickly build up its knowledge of local memberships. When a host joins a new group, it should immediately transmit a report for that group, rather than waiting for a query, in case it is the first member of that group on the network.

To cover the possibility of the initial report being lost or damaged, it is recommended that it be repeated once or twice after short delays. (A simple way to accomplish this is to act as if a query had been received for that group only, setting the group's random report delay timer.

B.4.1 Group address binding

The binding of IP host group addresses to physical hosts may be considered a generalization of the binding of IP unicast addresses. An IP unicast address is statically bound to a single local network interface on a single IP network. An IP host

group address is dynamically bound to a set of local network interfaces on a set of IP networks. Therefore an IP host group address is not bound to a set of IP unicast addresses. The multicast routers do not need to maintain a list of individual members of each host group. For example, a multicast router attached to an Ethernet only needs to associate with a single Ethernet multicast address with each host group having local members, rather than a list of the member's individual IP or Ethernet addresses.

B.4.2 Host group addresses

Host groups are identified by class D IP addresses, class D IP addresses are addresses with "1110" as their high-order four bits. Class E IP addresses are addresses with "1111" as their high-order four bits, and are reserved for future addressing modes. In Internet standard "dotted decimal" notation, host group addresses range from 224.0.0.0 to 239.255.255.255. The address 224.0.0.0 are not assigned to any group, and 224.0.0.1 are assigned to the permanent group of all IP hosts (including gateways). This address is used to communicate with all multicast hosts on the directly connected network. There is no multicast address (or any other IP address) for all hosts on the total Internet. The addresses of other well-known, permanent groups are to be published in "Assigned Numbers".

IGMP is used between hosts and gateways on a single network to establish hosts' membership in particular multicast groups. The gateways use this information, in conjunction with a multicast routing protocol, to support IP multicasting across the Internet. At this time, implementation of IGMP is optional, meaning that without IGMP, a host can still participate in multicasting local to its connected networks.

A host should support local IP multicasting on all connected networks for which a mapping from Class D IP addresses to link-layer addresses has been specified. Support for local IP multicasting includes transmitting multicast datagrams, joining multicast groups and receiving multicast datagrams, and leaving multicast groups. This implies support for all of [RFC 1112] except the IGMP protocol itself, which is optional. IGMP provides gateways that are capable of multicast routing with the information required to support IP multicasting across multiple networks. At this time, multicast-routing gateways are in the experimental stage and are not widely available.

For hosts that are not connected to networks with multicast-routing gateways or that do not need to receive multicast datagrams originating on other networks, IGMP serves no purpose and is therefore optional. However, the rest of [RFC 1112] is currently recommended for the purpose of providing IP-layer access to local network multicast addressing, as a preferable alternative to local broadcast addressing. It is expected that IGMP will become recommended at some future date, when multicast-routing gateways have become more widely available. [RFC 1122]

If IGMP is not implemented, a host should still join the "all- hosts" group (224.0.0.1) when the IP layer is initialized and remain a member for as long as the IP layer is active. Joining the "all-hosts" group will support strictly local uses of multicasting, e.g., a gateway discovery protocol, even if IGMP is not implemented. The mapping of IP Class D addresses to local addresses is currently specified for the following types of networks:

- Ethernet/IEEE 802.3.
- Any network that supports broadcast but not multicast, addressing: all IP Class D addresses map to the local broadcast address.
- Any type of point-to-point link (e.g., SLIP or HDLC links): no mapping required. All IP multicast datagrams are transmitted as-is, inside the local framing.

Mappings for other types of networks will be specified in the future. A host should provide a way for higher-layer protocols or applications to determine which of the host's connected network or networks support IP multicast addressing. An IGMP version 1 header can be seen in figure B.4.1, which is followed by an explanation of the different fields.

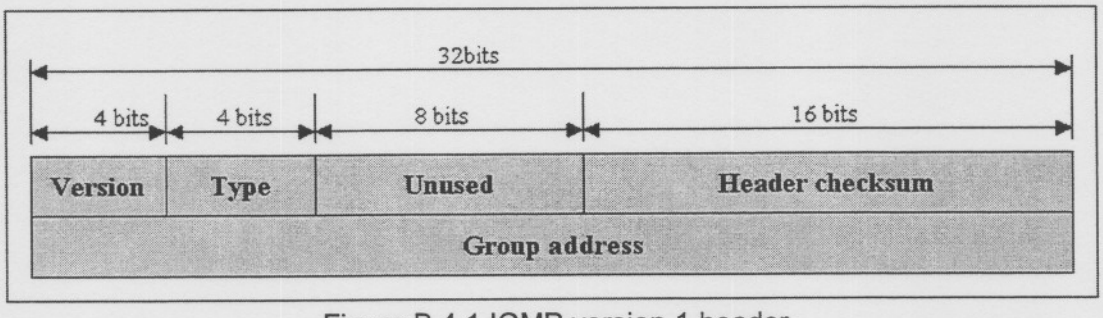

Figure B.4.1 IGMP version 1 header

The version field contains the version of ICMP that are used and in this instance it will contain a 1. The type field is 4 bits in total length and contains a 1 for host

membership query, 2 for host membership report and 3 for DVMRP. The unused field is set to zero when the IGMP packet are transmitted and ignored when the packet is received. The checksum field contains the 16-bit one's complement of the one's complement sum of the 8-byte IGMP message. In a host membership query message, the group address field is set to zero when transmitted and ignored when received. In a host membership report message, the group address field holds the IP host group address of the group being reported.

IGMP version 2 allows group membership termination to be quickly reported to the routing protocol, which is important for high bandwidth multicast groups and/or subnets with highly volatile subgroups. IGMP messages are encapsulated in IP datagrams, with an IP protocol number of 2. All IGMP messages described in this document are transmitted with an IP time to live one, and contain the IP router alert option in their IP headers. The router alert option lets the routers examine the packet more closely. By including the router alert option in the IP header of its protocol message, RSVP can cause the message to be intercepted while causing little or no performance penalty on the forwarding of normal data packets.

Multicast routers use IGMP version 2 to learn which groups has members on each of their attached physical networks. A multicast router keeps a list of multicast group memberships for each attached network, and a timer for each membership. "Multicast group memberships" means the presence of at least one member of a multicast group on a given attached network, not a list of all of the members. When a host receives a general query, it sets delay timers for each group (excluding the all-systems group) of which it is a member on the interface from which it received the query.

When a router receives a report, it adds the group being reported to the list of multicast group memberships on the network on which it received the report and sets the timer for the membership to the group membership interval. When a host joins a multicast group, it should immediately transmit an unsolicited version 2-membership report for that group, in case it is the first member of that group on the network When a host leaves a multicast group, if it was the last host to reply to a query with a membership report for that group, it should transmit a leave group message to the all-routers multicast group (224.0.0.2). Figure B.4.2 is an illustration of the IGMP version 2 header followed by an explanation of the fields [RFC 2236].

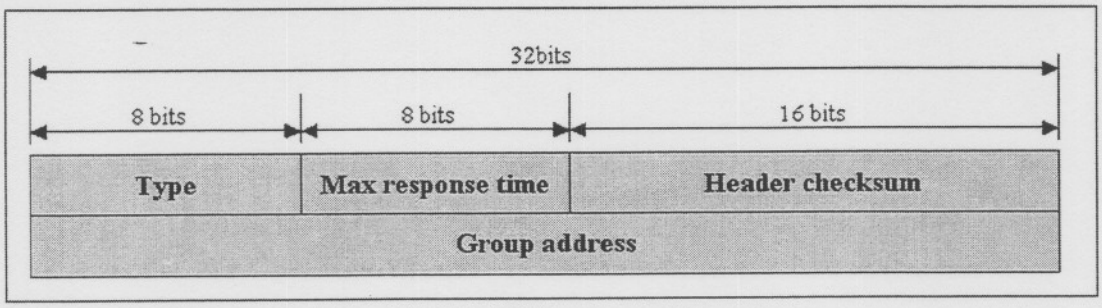

Figure B.4.2 IGMP version 2 header

The type field is 8-bits in total length and contain information about the type of IGMP message this is. The contents of this field can be seen in table B.4.1.

Type	Description
0x11	Group membership query, general or specific. General query is used to learn which groups have members on an attached network. A group specific query is used to learn if a particular group has any members on their attached network. These two types of messages are differentiated through their group addresses.
0x12	IGMPv1 membership report
0x13	DVMRP
0x14	PIMv1
0x15	Cisco trace messages
0x16	IGMPv2 membership report
0x17	IGMPv2 leave group
0x1E	Multicast trace route report
0x1F	Multicast trace route
0x22	IGMPv3 membership report
0x24	Multicast router advertisement
0x25	Multicast router solicitation
0x26	Multicast router termination
0xF0-0xFF	Experimental

Table B.4.1 IGMPv2 type field contents

The maximum response time field is also 8-bits in total length and is only used in membership query messages. The field then specifies the maximum allowed time before transmitting a responding report in units of 1/10 second. If any other message is transmitted this field is set to zero and ignored by the receivers. The IGMP checksum field is 16-bits in total length and contains the 16-bit one's compliment of the one's compliment sum of the IGMP message starting at the type field.

The group address field is 16-bits in total length and contains a zero value if a group address is transmitted and the group address that is being queried if a group specific query is transmitted. If a membership report or leave group message is transmitted

this field holds the IP multicast group address of the group that is being reported or left. A variant of IGMPv2 that adds user authentication is IGAP. IGAP enables an IP multicast service provider to authenticate requests to join a specific multicast group based on user information.

B.5 RGMP (Router group port management protocol)

RGMP was developed by Cisco systems to restrict multicast packet forwarding in switches to routers where the packets may be needed. This protocol contains a 64 – bit header that can be seen in figure B.5.1. RGMP was designed for use on backbone switched network where multiple high-speed routers are interconnected.

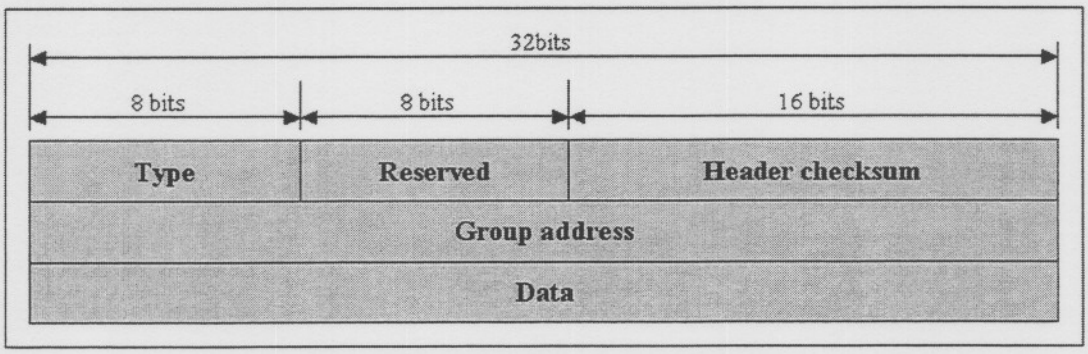

Figure B.5.1 RGMP header construction

The type field is 8-bits in total length and contains information that would identify the RGMP message. If the field had a 0xFC value the message would be a Leave group message, if the value were 0xFD the message would be a join group message, if the value were 0xFE the message would be a bye message and if the value were 0xFF the message would be a hello message.

The checksum field is 16-bits in total length and uses the same algorithm to determine the checksum as IGMPv3 uses. The group address is 32-bits in total length and in a RGMP hello or bye message the field is set to zero. In a RGMP join or leave message the field will hold the IPv4 multicast group address of the group being joined or left. The reserved field is set to zero for all transmissions [RFC 3488].

B.6 GGP (Gateway to gateway protocol)

This protocol is part of the TCP/IP protocol suite and its main task is to route datagrams and other gateway tasks through the network. The GGP packet construction can be seen in figure B.6.1 followed by an explanation of the different

fields. GGP was designed and implemented by BBN for the first experimental Internet gateways.

GGP is based on a minimal hop algorithm it therefore requires a global convergence of the routing tables after a change in network topologies or connectivity. Each gateway transmits a GGP routing update to its neighbors, which includes an entry for each known network [RFC 1009,792,1812].

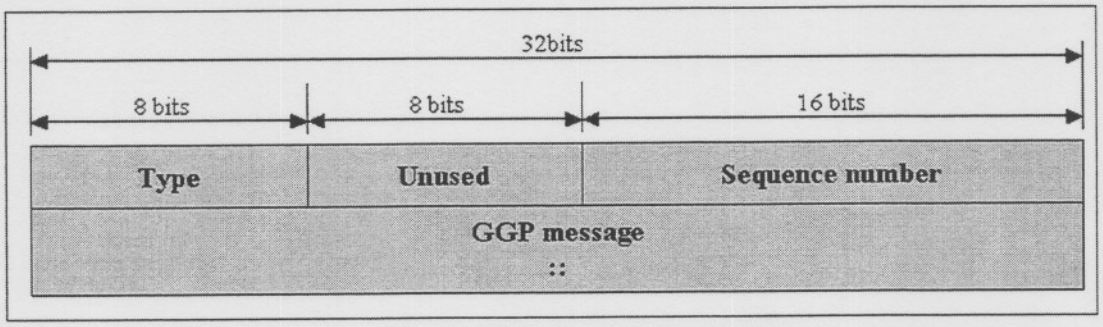

Figure B.6.1 GGP packet construction

The type field specifies the format of the GGP message and is 8-bits in total length. The unused field is 8-bits in total length and must be set to zero. The GGP message can vary in length. The contents of the type field can be seen in table B.6.1.

Type	Description
0	Echo reply
2	Acknowledgement
8	Echo request
9	Network interface status
10	Negative acknowledgement
12	Routing update

Table B.6.1 GGP type field content

B.7 IP in IP encapsulation

This type of configuration is normally used for tunneling purposes, with the outer IP header specifying the end and start points of the tunnel and the inner IP header specifying the destination and source IP addresses. The inner IP header is not changed, except to decrement the time to live and remains unchanged during its delivery to the tunnel exit point. The inner headers security options may also affect the outer headers security options. The outer header is constructed in the same format as a normal IP header [RFC 1853, RFC 2003].

B.8 Internet stream protocol (ST)

The Internet stream protocol is an experimental connection orientated internetworking protocol that operates at the same layer as connectionless IP. This protocol is used to deliver data streams to single or multiple destinations for applications that require a guaranteed quality of service. ST can therefore be used to reserve bandwidth for real time applications ensuring that real time packets are delivered in time. ST consists of two protocols, ST (providing data transport) and SCMP (providing control functions).

ST contains only a single PDU format and can therefore achieve low communication delays. ST applies the same addressing schemes as IP with the difference in the first four bits in the packet, which contains the protocol version number.

Data are transferred in streams, with a stream established between the transmitting node and the receiving node/nodes in the form of a routing tree. Nodes in the tree are ST agents and links are called hops with the ST agents executing the ST protocol. The streams are established and maintained by SCMP. The ST header format can be seen in figure B.8.1.

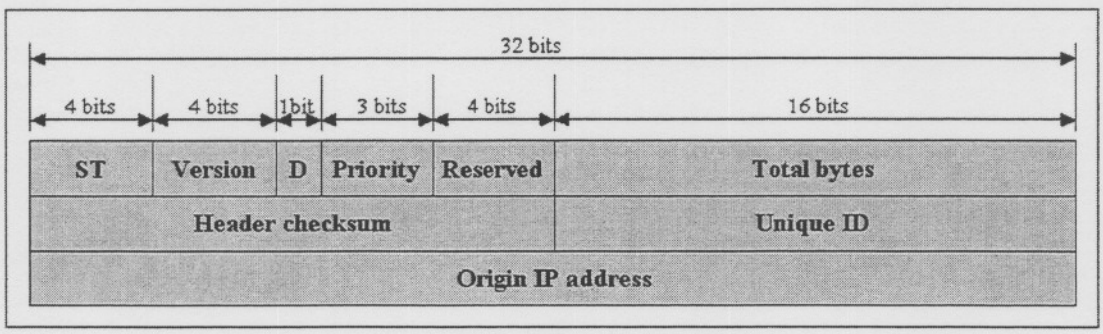

Figure B.8.1 ST header format

B.9 CBT (Core based trees)

The CBT protocol is a multicast routing protocol that builds and maintains a shared delivery tree for a multicast group. The transmitting and the receiving of multicast data by hosts on a sub-network conforms to the traditional IP multicast service model. CBT is mainly suited for inter and intra domain multicast routing and builds a shared multicast distribution tree per group.

An extension to CBT routing is abstract CBT, which is a multicast routing architecture that builds a single delivery tree per group and shares it between all the transmitters and receivers. The primary advantage of a shared tree approach within multicasting is that it would typically offer more favorable calling characteristics than all other multicast algorithms [RFC 2189, 2201]. Figure B.9.1 shows a typical version 2 CBT header followed by an explanation of the different header fields.

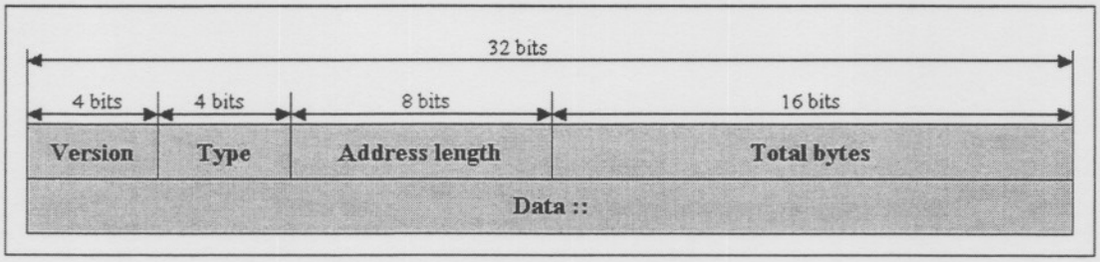

Figure B.9.1 CBT header format

The first field in the CBT header is the CBT version field and contains information concerning the version of CBT used in this frame, the second field is the type field and contains information about the type of the CBT message. The different types that the message could be can be seen in table B.9.1.

The next field is the address length field and is 8-bits in length and holds information which depicts the size of the unicast or multicast addresses carried in the control packet in bytes. The checksum field follows this field, which holds the 16-bit one's complement of the one's compliment sum of the entire CBT packet.

Type	Description
0	Hello
1	Join request
2	Join acknowledgement
3	Quit notification
4	Echo request
5	Echo reply
6	Flush tree
7	Bootstrap message
8	Candidate core advertisement

Table B.9.1. CBT type field definitions

B.10 EGP (Exterior gateway protocol)

The main aim of EGP is to convey net-reachability information between neighboring gateways. These gateways may even be in different autonomous systems. The

protocol therefore includes mechanisms to acquire neighbors, monitor neighbor reachability and exchange net-reachability information. The method used to determine the net-reachability is a periodic polling method. Figure B.10.1 is a representation of a typical EGP header and its relevant fields [RFC 904].

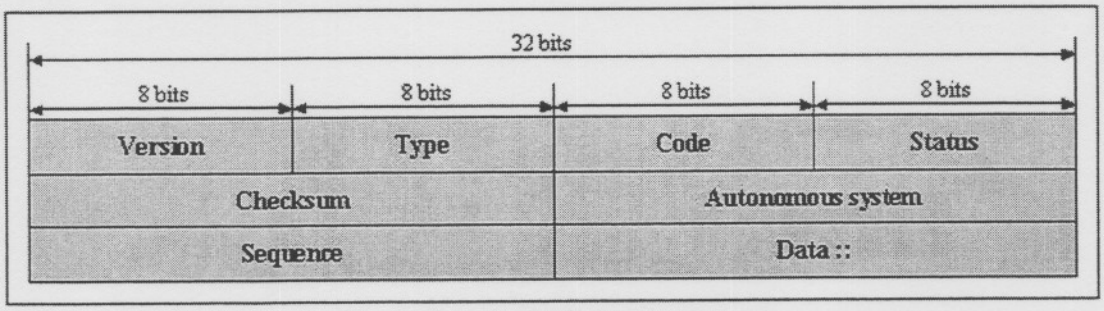

Figure B.10.1 EGP header format

The version field holds the version of the EGP currently used and the type field contains information concerning the type of EGP message this is. The status field however contains message specific status information and the checksum field holds the one's complement of the one's compliment sum of all the EGP fields.

The autonomous field contains the assigned number used to identify the particular autonomous system. The sequence number is used for the send state variables (commands) or receive state variables (responses and indications) [RFC 827,888,890,911,9751004,1092,1156,1158].

B.11 UDP (User datagram protocol)

UDP is normally used for services, which doesn't need the level of service of TCP or to services that wish to use communication services not available from TCP. UDP therefore only offers a minimal transport service, a non-guaranteed datagram delivery as well as direct access to the datagram service of the IP layer for applications.

UDP is almost a null protocol; the only services it provides over IP are checksumming of data and multiplexing by port number. Therefore, an application program running over UDP must deal directly with end-to-end communication problems that a connection-oriented protocol would have handled for example, retransmission for reliable delivery, packetization and reassembly, flow control, congestion avoidance, etc., when these are required. The fairly complex coupling

between IP and TCP will be mirrored in the coupling between UDP and many applications using UDP. A representation of the UDP header and the fields it's constructed with can be seen in figure B.11.1.

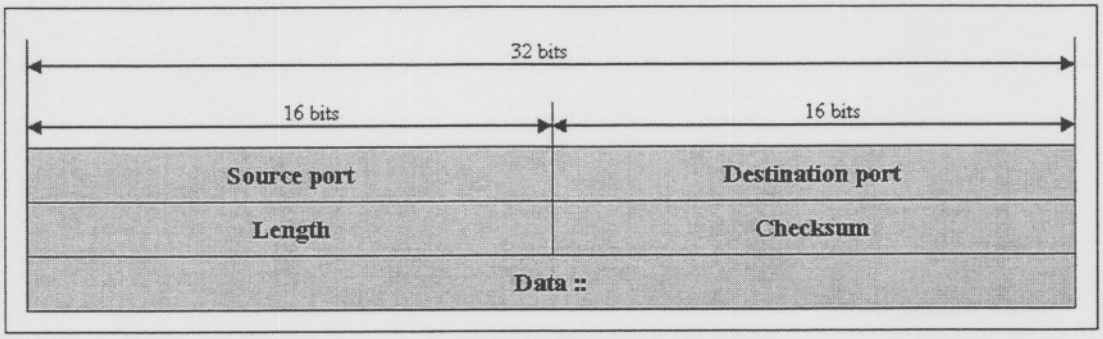

Figure B.11.1 UDP header format

The source port field is 16-bits in total length and is an optional field. If it is not used, it is set to zero. In the event it is used, it specifies the port of the sender. The destination port is also 16-bits in total length and hold the address of the port the packet is addressed to. The length field is 16-bits in length and contains the length in bytes of the UDP header and the encapsulated data.

The minimum value for this field is 8. The checksum field is 16-bits in length and contains the header checksum value which is computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes.

If the checksum is set to zero, then checksuming is disabled. If the computed checksum is zero, then this field must be set to 0xFFFF. When transported by IPv4, the pseudo header contains the fields depicted in figure B.11.2. If the pseudo header is transported by IPv6 the fields contained within the header can be seen in figure B.11.3.

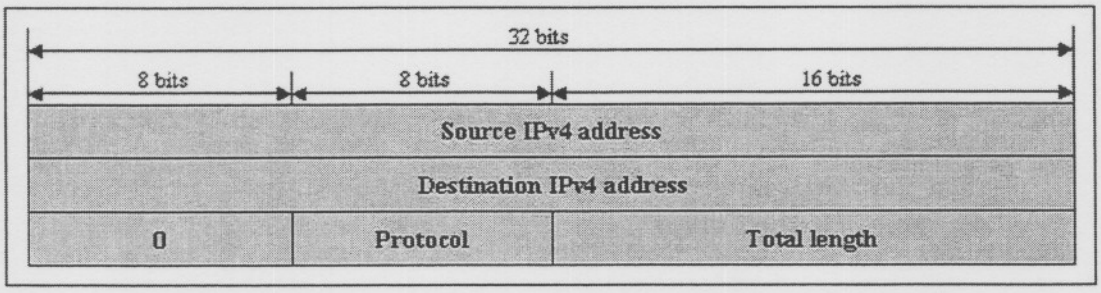

Figure B.11.2 Pseudo header contents if the field is carried via IPv4

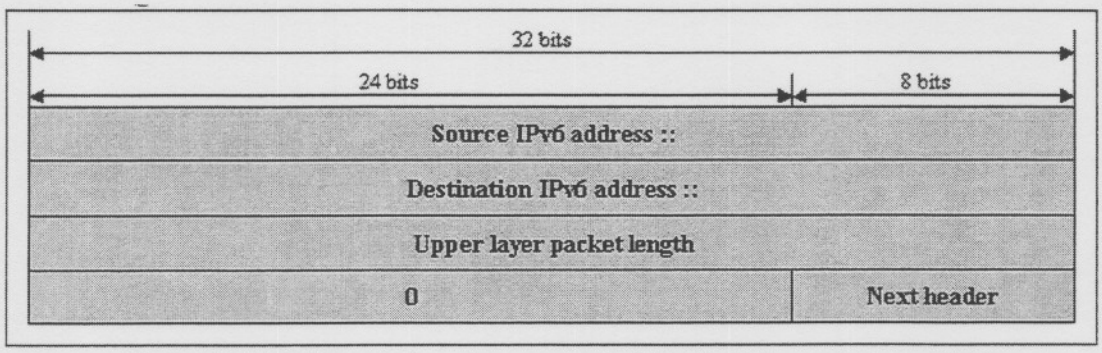

Figure B.11.3 Pseudo header contents if the field is carried via IPv6

B.12 IRTP (Internet reliable transaction protocol)

IRTP is a full duplex, transaction oriented, host-to-host protocol, which provides reliable sequenced delivery of packets of data, called transaction packets. A typical IRTP header can be seen in figure B.12.1 followed by a description of the different header fields.

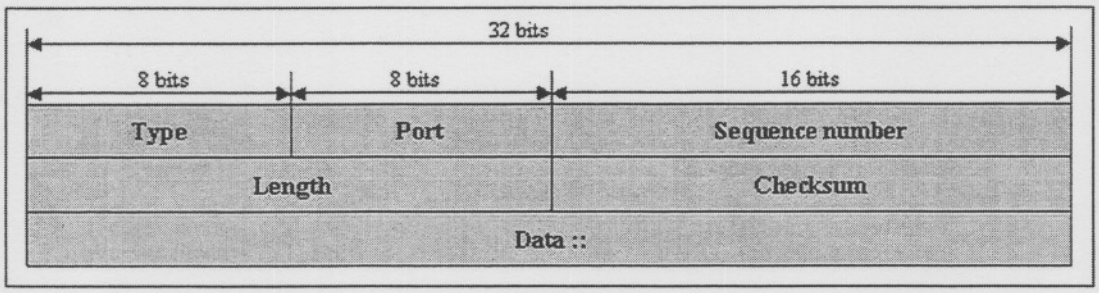

Figure B.12.1 IRTP header format

The type field is 8-bits in length and holds information about the type of IRTP message. The port field is 8-bits in length and is used for the multiplexing and demultiplexing of packets from multiple user processes across a single IRTP connection. Processes that desire to use IRTP must claim port numbers.

A port number represents a higher level protocol, and data to/from this port may be exchanged only with a process which has claimed the same port number at a remote host. A process can claim multiple port numbers, however, only one process may claim an individual port number with all port numbers are well known [RFC 938]. The sequence number field is 16-bits in length and contains for each communicating pair of hosts, two sequence numbers, which are the send sequence numbers for the two ends. Sequence numbers are treated as unsigned 16 bit integers. Each time a new transaction packet is sent, the sender increases the sequence number by one. Initial

sequence numbers are established when the connection is resynchronized. The length field holds the amount of bytes in this transaction packet including the header and data. The checksum field holds the 16-bit one's compliment of the 16-bit ones\'s compliment sum of the IRTP header and the transaction packet data. The data field contains the data and can vary in length [RFC 938].

B.13 SDRP (Source demand routing protocol)

The main purpose of SDRP is to support source-initialized selection of routes to compliment the route selection provided by existing routing protocols for both inter domain and intra domain routes. SDRP makes minimal assumptions about the distribution and acquisition of routing information needed to construct the SDRP routes. The SDRP header configuration can be seen in figure B.13.1.

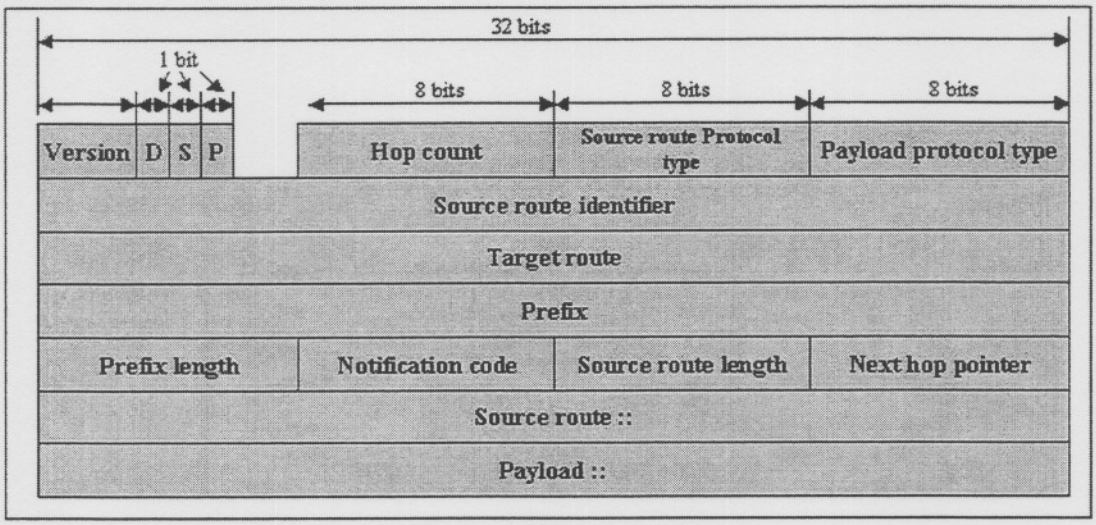

Figure B.13.1 SDRP header format

The version field contains the version of the SDRP used and the flag field holds information about the type of packet. The flag field values can be seen in table B.13.1.

Value	Description		
	D (Data control flag)	S (Loose/strict source route)	P (Probe indicator)
0	Packet is carrying control information	The next hop is a loose source route	The packet contains control information
1	Packet is carrying data	The next hop is a strict source hop	The origin of the packet is probing the network

Table B.13.1 SDRP flag field content

The hop count field is 8-bits in length and contains the maximum amount of hops the SDRP packet can undergo before the data packet may transverse. The value in the field is decremented with 1 until it reaches zero. If a router receives a packet with a zero hop count, the packet should be discarded and a control packet should be created. The source route protocol type is 8-bits in length and indicates the type of information that appears in the source route.

If this field contains one, the source route is explicit but if the field contains a zero, the source route type is setup. The payload protocol type indicates the protocol type of the payload while the source route identifier field contains 32-bit value generated by the transmitting host, which serves as a route identifier. The target router field is only utilized in control packets, and contains one of the IP addresses of the router that that originated the SDRP packet that triggered the control packet to be returned.

The prefix field contains an IP address prefix with only the number of bits specified by the prefix length significant. The field is also used to prevent routing loops when BGP or IDRP is used to route to the next hop in a loose source route. The notification field is 8-bits in length and is used for control packets, and is therefore zero for data packets. The values of the field for a control packet can be seen in table B.13.2.

Code	Description
1	No route is available
2	Strict source route has failed
3	Transit policy violation
4	Hop count has been exceeded
5	Probe has been completed
6	Unimplemented SDRP version
7	Unimplemented source route protocol type
8	Setup request rejected

Table B.13.2 SDRP notification field contents

The source route length field is 8-bits in length and holds the amount of 32-bit words of the domain level source route carried in the SDRP header. The next hop pointer field indicates the offset of the high order byte of the next hop along the route that the packet has to be forwarded. This offset is relative to the start of the source route field. Implying that if the value of the next hop pointer field equals the value of the source route length field, then the entire source route has been completely traversed and all other source routes are then incompletely traversed.

The payload field carries the datagram originated by the end system within the domain that constructed the SDRP packet. The Payload field forms the data portion of the SDRP packet. In a control packet this field may be empty or may carry the payload header of the packet that triggered the control message. It must be remembered that there is no padding between the Source Route and the Payload, and that the Payload may start at any arbitrary byte boundary [RFC 1940].

B.14 The normal distribution

A normal distribution in a variate X with a mean ρ and variance σ^2 has a probability function B.1 on the domain $x \in (-\infty, \infty)$.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\rho)^2/(2\sigma^2)} \qquad B.1$$

Physicists also call this distribution the Gaussian distribution. A representation of a normal distribution can be seen in figure B.15.1.

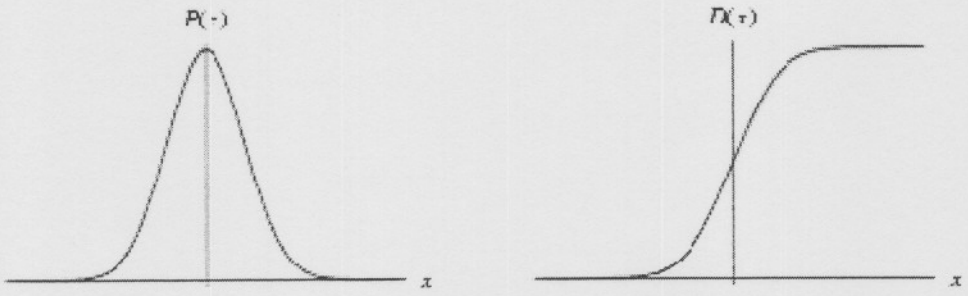

Figure B.15.1 Normal distribution

destination might be unreachable. Most commonly, the source host has specified a nonexistent address and secondly the router may not have a route to the destination. These kinds of messages usually include four basic types.

- Network unreachable: These messages usually mean that a failure has occurred in the routing or addressing of the packet.
- Host unreachable: These messages usually indicate delivery failures for example a wrong subnet mask.
- Protocol unreachable: In the case of such a message the destination does not support the upper layer protocol specified within the packet.
- Port unreachable: A message of this kind implies that the TCP socket or port is unavailable.

To test node reach-ability an ICMP echo-request message is used, which is generated through a ping command and is transmitted by the host to test the reach-ability over the network. If an ICMP echo reply message is received the node has been successfully reached. An ICMP Redirect message is sent by the router to the source host to stimulate more efficient routing. The router still forwards the original packet to the destination. ICMP redirects allow host routing tables to remain small because it is necessary to know the address of only one router, even if that router does not provide the best path. Even after receiving an ICMP Redirect message, some devices might continue using the less-efficient route [5,6,11].

An ICMP Time-exceeded message is transmitted by the router if an IP packet's Time-to-Live field (expressed in hops or seconds) reaches zero. The Time-to-Live field prevents packets from continuously circulating the inter-network if the inter-network contains a routing loop. The router then discards the original packet.

B.2.1 ICMP router discovery protocol (IDRP)

IDRP uses Router-Advertisement and Router-Solicitation messages to discover the addresses of routers on directly attached subnets. Each router periodically multicasts Router-Advertisement messages from each of its interfaces. Hosts then discover addresses of routers on directly attached subnets by listening for these messages. Hosts can use Router-Solicitation messages to request immediate advertisements rather than waiting for unsolicited messages. IRDP offers several advantages over other methods of discovering addresses of neighboring routers.

Primarily, it does not require hosts to recognize routing protocols, nor does it require manual configuration by an administrator. Router-Advertisement messages enable hosts to discover the existence of neighboring routers, but not which router is best to reach a particular destination. If a host uses a poor first-hop router to reach a particular destination, it receives a Redirect message identifying a better choice [11].

B.3 IGMP (Internet group management protocol) version 0

IP multicasting can be defined as the transmission of an IP datagram to a "host group" which could be defined as a set of zero or more hosts identified by a single IP destination address. It must however be remembered that if a multicast datagram is transmitted it will be delivered to all the members of its destination host group with the same best effort as regular unicast IP datagrams. The membership of a host to a host group can be classified as dynamic meaning that hosts may join and leave the group at any time.

There is therefore no restriction on the location or number of members in a host group, but membership in a group may be restricted to only those hosts possessing a private access key. A host may be a member of more than one group at a time and need not be a member of a group to transmit datagrams to it.

A host group may be permanent or transient with a permanent group containing well known administratively assigned IP addresses. It is the address, not the membership of the group, which is permanent. A permanent group may have any number of members, even zero at any time. A transient group, on the other hand, is assigned an address dynamically when the group is created, at the request of a host. A transient group ceases to exist, and its address becomes eligible for reassignment, when its membership drops to zero.

The creation of transient groups and the maintenance of group membership information is the responsibility of "multicast agents". Multicast agents are entities that reside in Internet gateways or other special-purpose hosts. There is at least one multicast agent directly attached to every IP network or sub-network that supports IP multicasting. A host requests the creation of new groups, and joins or leaves existing groups, by exchanging messages with a neighboring agent. Multicast agents are also responsible for inter-network delivery of multicast IP datagrams.