ANALYSIS OF RED PACKET LOSS PERFORMANCE IN A SIMULATED IP WAN

by

Nico Engelbrecht

Submitted in partial fulfilment of the requirements for the degree

Master of Engineering (Computer Engineering)

in the

Faculty of Engineering, the Built Environment and Information Technology

UNIVERSITY OF PRETORIA

April 2013

SUMMARY

ANALYSIS OF RED PACKET LOSS PERFORMANCE IN A SIMULATED IP WAN

by

Nico Engelbrecht

Supervisor: Prof A. P. Engelbrecht

Department: Department of Computer Science

University: University of Pretoria

Degree: Master of Engineering (Computer Engineering)

Keywords: RED, packet loss, packet discarding, network delay, differentiated services, quality of service, network simulation

The Internet supports a diverse number of applications, which have different requirements for a number of services. *Next generation networks* provide high speed connectivity between hosts, which leaves the service provider to configure network devices appropriately, in order to maximize network performance. Service provider settings are based on best recommendation parameters, which give an opportunity to optimize these settings even further.

This dissertation focuses on a packet discarding algorithm, known as *random early detection* (RED), to determine parameters which will maximize utilization of a networking resource. The two dominant traffic protocols used across an IP backbone are *user datagram protocol* (UDP) and *transmission control protocol* (TCP). *UDP traffic* flows transmit packets regardless of network conditions, dropping packets without changing its transmission rates. However, *TCP traffic* flows concern itself with the network condition, reducing the packet transmission rate based on packet loss. *Packet loss* indicates that a network is *congested*. The *sliding window* concept, also known as the TCP *congestion window*, adjusts to the number of acknowledgements the source node receives from the destination node. This paradigm provides a means to transmit data across the available bandwidth across a network.

A well known and widely implemented simulation environment, the *network simulator* 2 (NS2), was used to analyse the *RED* mechanism. The NS2 software gained its popularity as being a complex networking simulation tool. Network protocol traffic (UDP and TCP) characteristics comply with theory, which verifies that the traffic generated by this simulator is valid. It is shown that the *autocorrelation function* differs between these two traffic types, verifying that the generated traffic does conform to theoretical and practical results. UDP traffic has a *short-range dependency* while TCP traffic has a *long-range dependency*.

Simulation results show the effects of the RED algorithm on network traffic and equipment performance. It is shown that *random packet discarding* improves source transmission rate stabilization, as well as node utilization. If the packet dropping probability is set high, the TCP source transmission rates are low, but a low packet drop probability provides high transmission rates to a few sources and low transmission rates to the majority of other sources. Therefore, an ideal *packet drop probability* was obtained to complement TCP source transmission rates and node utilization. Statistical distributions were modelled according to sampled data from the simulations, which also show improvements to the network with random packet discarding.

The results obtained contribute to congestion control across wide area networks. Even though a number of queuing management implementations exists, RED is the most widely used implementation used by service providers.

OPSOMMING

ANALISE VAN RED PAKKIE VERLOOR PRESTASIE IN ‘N GESIMULEERDE IP WAN

deur

Nico Engelbrecht

Studieleier: Prof A. P. Engelbrecht

Departement: Departement van Rekenaar Wetenskappe

Universiteit: Universiteit van Pretoria

Graad: Magister in Ingenieurswese (Rekenaar Ingenieurswese)

Sleutelwoorde: RED, pakkie verloor, pakkie weggooi, netwerk vertraging, gedifferensieerde dienste, kwaliteit van die diens, 'n netwerk simulasie

Die Internet akkommodeer 'n diverse aantal applikasies, wat verskillende vereistes vir 'n aantal dienste benodig. Volgende generasie netwerke bied hoë spoed konneksies tussen verskeie terminale, wat die diensverskaffer verantwoordelik hou om toestelle toepaslik te konfigureer, om die netwerk prestasie te maksimeer**.** Diensverskaffer instellings is gebaseer op die beste aanbevole parameters, wat 'n geleentheid aanbied om hierdie instellings nog verder te optimeer.

Hierdie verhandeling fokus op 'n pakkie verlies algoritme, bekend as “*random early detection*” (RED), om parameters te bepaal wat die gebruik van ‘n netwerk hulpbron te maksimeer. Die twee dominante verkeer protokolle wat gebruik word oor 'n IP rugsteen “*user datagram protocol*” (UDP) en “*transmission control protocol*” (TCP). UDP tipe verkeer stuur pakkies ongeag van die netwerk kondisies en gooi pakkies weg sonder om die transmissie spoed te wysig. TCP tipe verkeer neem network konsies in ag, deur die vermindering van die pakkie transmissie gebaseer op die pakkie verlies. Pakkie verlies dui aan dat 'n netwerk oorbelas is. Die verskuif venster konsep, ook bekend as die “TCP congestion window”, pas aan tot die aantal erkennings wat vanaf die bron node ontvang word. Hierdie paradigma bied 'n manier aan om data oor die beskikbare bandwydte oor 'n netwerk te stuur.

'n Bekende en wyd geïmplementeer simulasie omgewing, die *netwerk simulator 2* (NS2), is gebruik om die RED-meganisme te analiseer. Die NS2 sagteware is gewild as 'n komplekse netwerk simulasie hulpmiddel. Netwerk protokol verkeer (UDP en TCP) voldoen aan teoreteise eienskappe, wat bevestig dat gegenereerde verkeer van hierdie simulator geldig is. Daar word aangetoon dat die outokorrelasie funksie verskil tussen hierdie twee verkeer tipes, wat verifieer dat die gegenereerde verkeer voldoen aan die teoretiese en praktiese resultate. UDP tipe verkeer het 'n kort-afstand afhanklikheid terwyl TCP verkeer het 'n lang-afstand afhanklikheid eienskappe het.

Simulasie resultate toon die uitwerking van die RED algoritme op netwerk verkeer en toerusting prestasie. Dit word getoon dat 'n willekeurige pakkie verlies verbeter en stabiliseer bron transmissie speod, sowel as node benutting gebruik. Indien die pakkie verlies waarskynlikheid is hoog, sal die TCP bron oordrag transmissie lag wees, maar 'n lae pakkie verlies waarskynlikheid bied hoë transmissie spoed vir 'n paar bronne en 'n lae transmissie spoed aan die meerderheid van die ander bronne. 'n Ideale pakkie verlies waarskynlikheid was verkry om TCP bron transmissie spoed en node gebruik te komplimenteer. Statistiese verspreidings is gemodelleer vanaf data monsters van die simulasies, wat ook verbeterings aan die netwerk wat willekeurige pakkie verlies gebruik toon.

Die resultate wat verkry is dra by tot kongestie beheer oor wye area netwerke. Selfs bestaan daar 'n aantal van toustaan beheer implementasies bestaan, is RED die mees gebruikte implementering wat deur diensverskaffers genruik work.

LIST OF ABBREVIATIONS

ACL	Access control lists
ACF	Autocorrelation function
AF	Assured forward
AToM	Any transport over MPLS
ATM	Asynchronous transfer mode
BB	Bandwidth broker
BE	Best effort
BRI	Basic rate interface
BGP	Border gateway protocol
CBWFQ	Class based weighted fair queue
CBTS	Class based traffic shaping
CBR	Constant bit rate
CBS	Committed burst size
CLI	Command line interface
CIR	Committed information rate
CPE	Customer premise equipment
CRC	Cyclical redundancy check
CRTP	Compressed real-time protocol
CIR	Committed information rate
CTR	Committed target rate
DCE	Data circuit-terminating equipment
DTE	Data terminating equipment
DiffServ	Differential services
EBS	Excess burst size
ECN	Explicit congestion notification
EF	Expedite forward
FCS	Frame check sequence
FR	Frame relay
FTP	File transfer protocol
GFI	General format identifier
GRE	Generic routing encapsulation
IntServ	Integrated services
IP	Internet protocol
ISDN	Integrated service digital network
ISP	Internet service providers
UDP	User datagram protocol
VoIP	Voice over IP
VPN	Virtual private network
WRED	Weighted random early detection
LAPD	Link access procedure, D channel
LCI	Logical channel identifier
LFI	Link fragmentation and interleaving
LLQ	Low latency queuing
LRD	Long-range dependence
MTU	Maximum transmission unit

MPLS	Multi packet label switching
MQC	Modular QoS CLI
NIC	Network interface card
OSI	Open system interconnection
PBS	Peak burst size
PBX	Private branch exchange
PHB	Per hop behaviour
PIR	Peak information rate
PLP	Packet layer protocol
PTR	Peak target rate
PTI	Packet type identifier
PVC	Permanent virtual circuit
PoS	Packet over SONET
PRI	Primary rate interface
RPSL	Router policy specification language
RSVP	Resource ReSerVation Protocol
RTP	Real-time transport protocol
SLA	Service level agreement
SPP	Service provisioning policy
SRD	Short-range dependence
SVC	Switched virtual circuit
SONET/SDH	Synchronous optical network/synchronous digital hierarchy
srTCM	Single-rate three-color marker
srRAS	Single-rate rate adaptive shaper
SMB	Server message block
TA	Terminal adapter
TCP	Transmission control protocol
TDM	Time division multiplexing
ToS	Type of service
TSWTCM	Time sliding window three color marker
trTCM	Two-rates three-color marker
trRAS	Two-rates rate adaptive shaper
QoS	Quality of service

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1. BACKGROUND

The *internet* serves as a means to transport data across a large number of loosely interlinked networks. *Information* sent between computers is divided into packets, which is then forwarded to more network devices, until the information reaches the desired destination. Because the traffic flows between different nodes, the traffic is not controlled by a single computer. Each packet that traverses through the network contains addressing information, which is then used by routers to make forwarding decisions.

Internet protocol (IP) networks are the most popular global communication infrastructures. An increasing number of different applications are continually conveyed by IP networks, causing a *fragmentation of performance and service requirements*. These applications experience packet loss and delay, which resulted in research being done into optimizing performance measures for effective performance. Continuous efforts have been made to develop a number of new technologies for enhancing quality of service capabilities [1].

The *quality of service* (QoS) delivered by the network mechanisms needs to ensure that data is transferred efficiently over a wide network within a certain tolerance level of service provided. Parameters of QoS are measured by jitter, delay and packet loss through a larger network carrying the communicating data.

Network performance measures differ for each customer, because each individual customer needs different service requirements for a diverse number of applications. To provide QoS on an individual basis is a difficult task. Two possible solutions are leased lines, namely integrated service digital network (ISDN), and permanent virtual circuit (PVC), namely frame relay (FR) [2].

Leased lines [2], such as ISDN, is a costly solution and does not offer a high degree of availability, since connections are prone to a dedicated link failure. Businesses experience

major losses when the single dedicated link goes down from time to time. The link thus needs to be constantly maintained by the service provider. This service limits functionality by not providing external access to the network, since the network is physically separated from the outside world.

Frame relay [2] uses *permanent virtual circuits* (PVC) to provide a connection-orientated service through a public network, but appears to be a dedicated physical connected link. These links provide sufficient quality guarantees, however, the drawback to PVCs is that these connections become costly and do not provide scalability of its services.

Virtual private networks (VPN) [2] can be seen as private networks connected over a public network, which appears as a single connected network. This type of network provides connectivity between remotely located business branches through the publicly used Internet. In fact, all data paths are secret to the outside world except to the users of the network, such as employees, managers, and network administrators.

Today's network parameters are based on *CISCO best practice values* for the various QoS mechanisms [3]. The research conducted in this dissertation was motivated by the fact that the *transmission control protocol* (TCP) throughput depends on delay and packet loss. Hence, this dissertation analyses, evaluates, and provides an analytical analysis of the delay and loss within a network environment, which purposely causes packet loss.

1.2. PROBLEM STATEMENT

The *integrated service* (IntServ) and *differentiated service* (DiffServ) are two techniques for end-to-end QoS as defined by the *internet engineering task force* (IETF) [4] [5]. IntServ provides end-to-end signalling, state maintenance and admission control at each network element. DiffServ provides that network traffic is separated into different classes, called *class of service* (CoS), which apply the QoS parameters to different flows of classes. Usually IntServ provides resource management inside local area networks, whereas DiffServ provides traffic regulation over wide area networks.

Network traffic transmission rates are limited by the use of traffic *policers* and *shapers*. *Policers* [6], also known as *packet droppers*, are used to limit the amount of bandwidth for delay-sensitive traffic, which makes this policy useful for real-time multimedia applications. *Shapers* [6] are responsible for regulating the rest of the data types that can tolerate some delay. Both these mechanisms limit the number of bits that may enter a network. Even though these mechanisms cause packet loss, their effects do not form part of the conducted research.

Random early detection (RED) [7] is an active queue management system, which is deployed on routers to randomly drop arriving packets, even though the queue for an outbound interface is not full. The different traffic flows are marked to be either green traffic, which has a low drop probability on the primary queue, or yellow marked traffic which has a high drop probability on the virtual queue. This research builds on the fact that the RED algorithm drops packets at the edge routers for the purpose of congestion avoidance. The core network needs to be routed and provisioned for bandwidth; therefore, packet loss guarantees are not a concern for this research. The difficulty to measure packet loss across a network backbone is because of the various paths that network traffic flows across.

Research describes analytically how traffic policy parameters influence packet loss and delay characteristics for a given traffic model, but a limited amount of work exists for the RED algorithm [1]. The research proposed is intended to provide RED parameters for network configuration, with the objective to achieve a target probability of packet loss in order to ensure that the probability of loss is not exceeded, or that it is maintained. Current implementations of service allocation provide that the loss in an end-to-end link be given by experienced best practice guesses from the system designers to the service providers. This research also gives insight into network protocol behaviours, showing how protocols react to the RED algorithm.

1.3. OBJECTIVES

The objective of this study is to determine the effects of the RED parameter values to network dynamic behaviour within a real-world implemented simulation environment by implementing mechanisms and protocols used across a network architecture with simulation. In order to understand the simulation environment and networking behaviour, the study within this research includes the following:

- Understanding the QoS concept and which factors it influence.
- Studying the mechanisms implemented to deliver QoS.
- Studying network dynamics, which includes mechanisms and protocols, and the development thereof to support QoS.
- Comparing the traffic generated by the simulator to theoretical and practical values.
- Determining how the RED algorithm affects the different traffic flows. The parameters of interest are:
    - packet loss probability,
    - TCP congestion window size probabilities,
    - TCP round trip time probability,
    - queuing size probability, and
    - node utilization probability.
    - This research also involves the effects that buffer management has on packet loss, delay, and queuing behaviour.
- Finding statistical distributions for the above mentioned parameters which aid with data analysis.
- Evaluating the results obtained.

1.4. RESEARCH CONTRIBUTION

The contribution of this research is to observe network behaviour when guaranteeing packet loss though RED, which provides congestion control in an effort to establish best empirical values within a network environment. This research requires a proper understanding of shapers, policers, packet scheduling, and the RED algorithm, which is primarily implemented for queue management.

Packets are discarded randomly to improve network traffic, thus it is important to determine the packet loss probability, by varying the RED parameter values. As a result, the effects that RED imposes on the transmitting sources are also observed. Since service providers aim to utilize resources effectively, it is important to determine how well network resources are utilized. Even though packet loss seems to be the measure that is going to be analysed, it is important to keep these network dynamics in mind.

1.5. RESEARCH OUTLINE

This dissertation consists of several chapters in which the various aspects of today's IP network are discussed, as well as the analytical aspects for the various QoS mechanisms which are used for simulation purposes. This chapter provides the problem statement, research objectives and contributions. Included below is the outline of the dissertation.

Chapter 2 discusses existing literature in a broad view as it relates to high level concepts across an Internet environment. This chapter aids as motivation to conduct this research, stating the agreements that the Internet needs to comply with.

Chapter 3 discusses the concept of "quality of service" that is provided by today's IP networks, as well as the various aspects which influence network services. This chapter provides information on the various factors that influence service quality because it is important to understand which factors influence network performance. The aim is to logically separate the influential factors that service providers need to optimize.

Chapter 4 provides information about the various aspects of an IP networks' architecture with regards to technology implementation and separation. This chapter discusses network protocols, network architectures, and models that are used within the existing technologies. To understand packet loss within a network, it is necessary to thoroughly understand how network traffic is handled once it propagates across network nodes.

Chapter 5 describes the methods used to generate traffic and configure networking parameters. It contains well defined parameters in an effort to observe how a certain mechanism, namely RED, changes network performance.

Chapter 6 provides the results obtained from statistical analysis and simulation, which consequently gives the best configuration parameters. The best parameters are used to observe the effects of network traffic and node utilization which can be implemented on any network that uses the RED algorithm.

CHAPTER 2 LITERATURE STUDY

2.1. INTRODUCTION

This chapter contains information concerning the importance of a service level agreement which is used to measure a service providers' performance to a customer. Section 2.2 discusses the service agreements between customer and service providers. Section 2.3 discusses the service provisioning policy that service providers use for service provisioning. Section 2.4 discusses the traffic models for the understanding of traffic behaviours between different traffic protocols. Section 2.5 discusses the traffic models that are supported by the discussion of the queuing theory philosophy. Customer network traffic within a backbone should conform to the agreed service levels stated within the service provider policy which leads to the discussion of the service architecture that is used to obtain the required service for required service violations. Section 2.6 discusses schedulers for the understanding on how a service class are treated before traffic is sent across an access link.

2.2. SERVICE LEVEL AGREEMENT

Internet service providers (ISPs) are responsible for monitoring, data regulation and to provide service quality reports to customers. The customer service from a service provider is agreed upon with the use of a *service level agreement* (SLA) [8]. The need to ensure service agreements arises from the fact that customers have a need for guaranteeing their applications that have a certain amount of bandwidth or delay bound.

The SLA consists of the following information:

- A description of the nature of the service provided. The types of service that must be provided and network maintenance measures are specified.
- Specifying the performance level guarantee that is expected from reliability and responsiveness.
- Monitoring and reporting service levels.
- The penalties for not adhering to the service specifications.
- Constraints and escape clauses.

In principle, the SLA seems simple. Yet, to obtain a general consensus to such an agreement is not easily reached, since the Internet is limited by the following concerns:

- A SLA is not dynamic, but need manual attention which makes it time consuming and prone to errors. The SLA is prone to regular changes, depending on service duration times and the provided service type.
- Traffic flowing between the providers and clients is not characterized by any parameters other than bandwidth. All traffic is contracted as being best-effort.
- ISP contracts are based upon two traffic types, namely transit and local traffic exchanges. Thus the contract only states services between neighbouring and world-wide networks.
- To ensure reliability or exploiting load balancing, the *border gateway protocol* (BGP) or *router policy specification language* (RPSL) cannot be used by itself, since traffic has many different paths throughout the Internet.
- Pricing can only be given by statically routed paths. An ISP cannot react quickly to changes in the market, which makes it difficult to establish peering agreements without any financial compensation.

Bandwidth is allocated by the following two types of *bandwidth brokers* (BB):

1. *Centralized agents* allocate bandwidth between any defined end-points. These agents provide bandwidth between companies for set connections. For example, Internet Café's provide a service for various clients by providing a set connection with a bigger service provider between customer sites.
2. *Decentralized agents* allocate bandwidth between neighbouring networks and exchange routing information to interconnected agents associated with these networks. For example, ISPs exchange routes that interconnect international data exchanges for world wide data transactions between countries regarding bandwidth and routing.

The services guaranteed in the SLA for the DiffServ environment are more flexible for allocation according to the type of service, since these services are divided into classes. In the case of DiffServ, provisioning is done according to bandwidth, delay, and loss of service specific applications such as *Voice over IP* (VoIP) [3].

This agreement is a challenging task, since customers have different service needs over a network consisting of multiple technologies having a large number of architectural variations. It is the service providers' aim to regulate network traffic across its network backbone. This dissertation concerns itself with packet loss, since TCP throughput is calculated as a function of delay and packet loss, as follows:

$$B(p) = \frac{MSS}{RTT\sqrt{p}} \tag{2.1}$$

where *B(p)* is TCP throughput, the *round trip time* (RTT) value is the round trip time between source and destination, the MSS value is a constant of 1460, and *p* is the packet loss rate parameter. It is clear that TCP throughput is characterized by average packet RTT and the *packet loss rate* parameter *p*. A more detailed mathematical model for TCP throughput is given in [9].

Delay guarantees have been studied extensively. However, packet loss is difficult to measure because of various mechanisms [3]. This dissertation entails loss probability and queuing performance, caused by the RED algorithm, to improve network performance.

2.3. SERVICE PROVISIONING POLICY

The s*ervice provisioning policy* (SPP) is a policy for traffic conditioners on DiffServ boundary nodes. This policy thus defines how traffic is mapped to DiffServ behaviour aggregates between networks or different traffic flow services. RFC 2475 [6] defines a service provisioning policy as "a policy which defines how traffic conditioners are configured on DiffServ boundary nodes and how traffic streams are mapped to DiffServ behaviour aggregates to achieve a range of services".

2.4. AN OVERVIEW OF EXISTING TELECOMMUNICATION NETWORKS

It is important to observe that traffic behaviour for a service provider can be distinguished by the protocol used for communication with various applications. Since traffic across a provider network is mixed in terms of applications and protocols it is not possible to have one model to characterize traffic behaviours. This section provides a discussion of work that was conducted in an effort to understand and model traffic behaviour. Section 2.4.1 discusses the different traffic models such as stochastic, deterministic, short- and long range dependant, and self-similar traffic models.

2.4.1. Traffic models

Traffic models are used to mathematically model network traffic behaviour with respect to the number of bits transmitted as a function of time over broadband networks [10]. Traffic behaviour directly affects network performance metrics such as throughput, delay, and packet loss probability. Hence, traffic modelling and management is essential for network planning, reporting on performance, and controlling network protocol design.

Research in network traffic modelling is continuously conducted, since there is no single model to fully characterize network traffic. No general consensus therefore exists on a common model characterization since more models arise with the growth of various network applications and implementations [11]. This work implements real simulated network traffic protocols which are verified by theoretical analysis that were conducted in practice by various academics [11].

The two classes these models fall into are *stochastic* and *deterministic models* [12]. *Stochastic models* are difficult to analyse, as it is difficult to establish a mathematical model to describe the network traffic due to source burstiness. *Deterministic models* are less complex to analyse due to the fact that the stochastic behaviour of traffic is bounded.

Stochastic models

Stochastic models are an attempt to characterize network traffic behaviour which tends to be extremely random in nature. Many authors, such as [13], [14], [15], [16] and [17], have modelled stochastic network traffic. Because of the heavy-tailed probability distribution behaviour, none of the proposed models fully characterize network traffic [11].

Traffic analysis becomes a daunting task, since mathematical equations become extremely complex, even for simple network configurations.

The most popular stochastic models used in theory include:

- Markov Modulated Poisson Process (MMPP) [14]
- Multi-fractal Wavelet Model (MWM) [15]
- Self-Similar Traffic Model [16]
- Hurst-parameter Model [17]
- S-BIND (Statistical Bounded Interval Dependant) Model

Deterministic models

Deterministic modelling provides a means to characterize network traffic with hard bounds, such that network traffic show limited stochastic behaviour [18]. The hard bounds also eliminate the occurrence of bursty traffic.

The most popular deterministic models used in theory include:

- Deterministic bounded interval-length dependant (D-BIND) model [19]
- Leaky bucket model [20]
- Token bucket model [20]
- Adversial model [21]
- (X_{min}, X_{ave} ,I, S_{max}) model
- Exponentially bounded burst (EBB) model

The goal of these traffic models is to control and manage network traffic with the aid of mathematical equations, as seen in [19], [20] and [21].

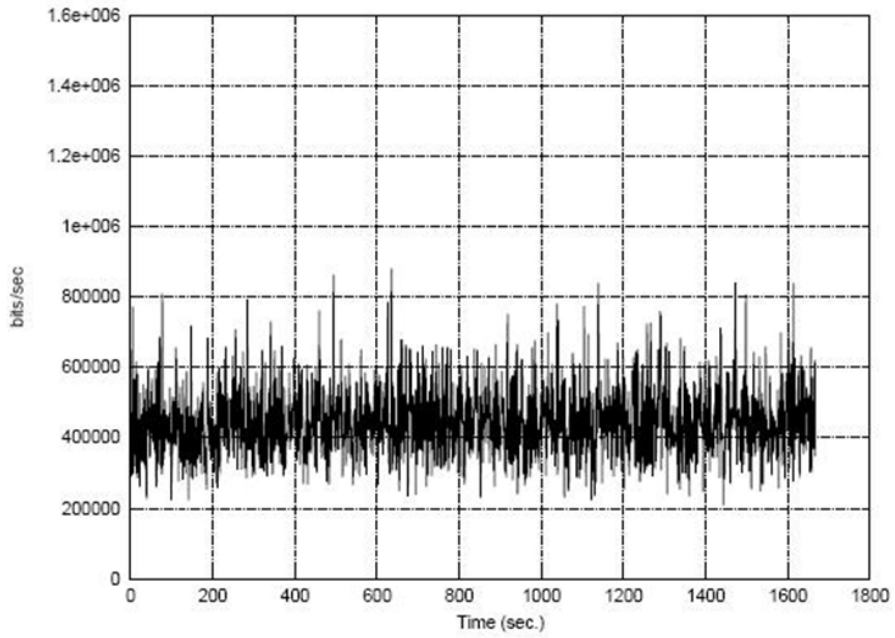
Short-range dependence, long-range dependence and self-similarity

Long-range dependence (LRD) models are used to characterize mixed voice and data traffic, which are heavily-tailed distributions. *Short-range dependence* (SRD) models are used to characterize voice traffic, which are exponentially distributed. Figure 1 illustrates practical results, from [22], of the actual traffic traces. Packet-switched networks exhibit both LRD and SRD, which are correlated over a wide range of time scales. LRD correlations decay slowly, while SRD correlations decay fast, as shown in Figure 2. Because network traffic is correlated, the inter-arrival time distribution becomes heavy-tailed and no longer exponential. Consequently, because of LRD, it becomes very difficult to design for congestion control and active queue management.

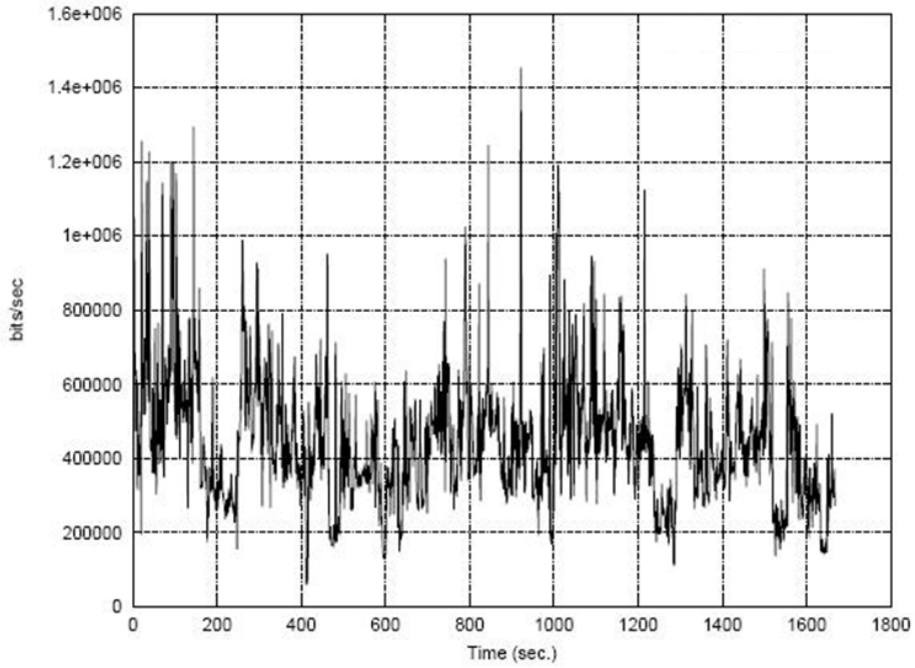
Self-similar traffic models are based on the characteristic that traffic "looks-alike", statistically modelled in [16]. For instance, when a number of multimedia application traffic is multiplexed on a network node, the traffic looks alike, resulting into self-similar behaviour. This phenomenon can be described by a *Hurst parameter*, studied in [17], [23], [24] and [25]. Simultaneous traffic bursts are the reason for the similar behaviour even over many time scales, exhibiting a special type of LRD. Since s*elf-similar traffic* looks the same over many time-scales, the same traffic pattern repeats over time; the amount of traffic over time exhibits a fractal property. A fractal is an object whose appearance is unchanged, regardless of the scale it was viewed. An example of this phenomenon is Ethernet network traffic [26].

The most popular *self-similar traffic* models used in theory include:

- Markov Modulated Poisson Process, which describes UDP traffic [14]
- Multifractal Wavelet Model (MWM) , which describes TCP traffic [15], and
- Fractional Brownian Motion (fBm), which describes TCP traffic [27].



(a) UDP traffic



(b) TCP traffic

Figure 1 UDP and TCP traffic arrival rates [22]

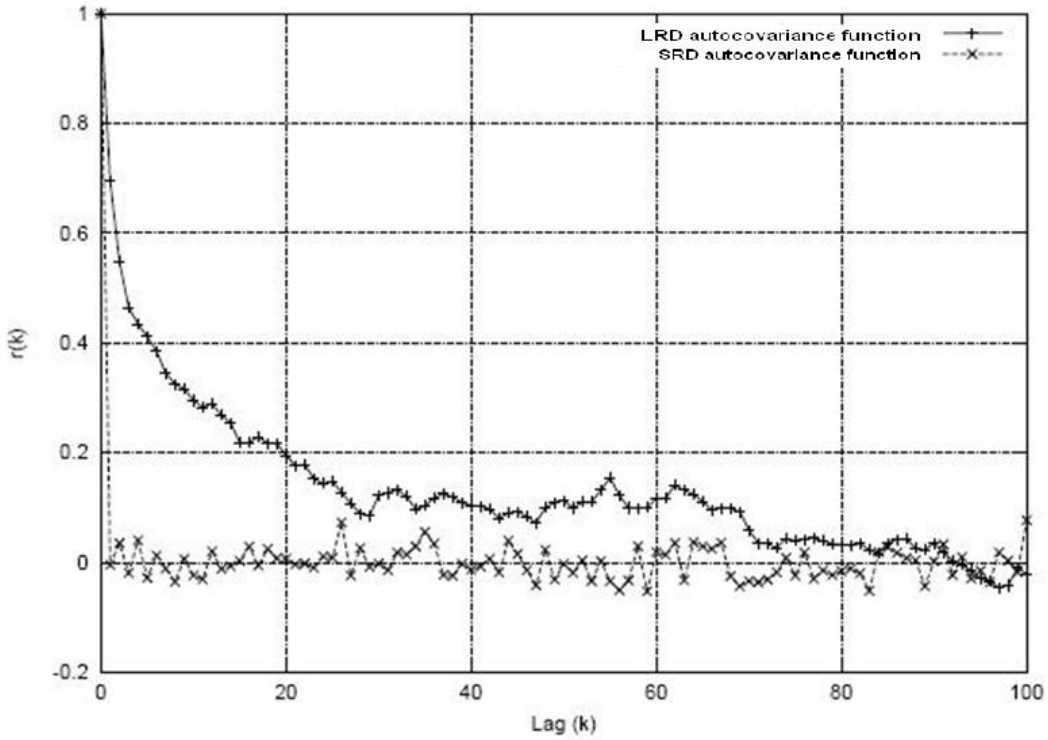


Figure 2 Short-range dependence vs. long-range dependence autocorrelation [22]

2.5. QUEUING THEORY

A large amount of theory exists regarding queuing implementations for various queuing systems, for example [28] and [29], ranging from queues at supermarkets to computer network applications. Multiplexed packets from a large number of independent bursty sources can accurately be modelled by means of a poisson process [11]. This leaves only the inter-arrival times as a network performance measure to be managed. The limitation of queuing theory lies in the fact that the statistical nature of these queues is based on Poisson distributions, which implies that no retry will be attempted by customers. For this reason, classical queuing does not accurately estimate overall network traffic performance, because classical queuing models fail to capture the heavy-tailed distributions of the queues.

The most important equation in queuing theory is known as "Little's theorem",

$$E[D] = \frac{E[Q]}{Link\ Capacity} \tag{2.2}$$

which states that the average delay (E[D]) is a function of the average queue size (E[Q]) and link capacity. This is a general and effective way in which delay is calculated, providing statistical delay bounds for queuing systems.

This section discusses the queuing methodologies that a network employ to control network traffic bandwidth usage. The differentiated service model in section 2.5.1 discusses that network traffic is classified into queues and processed differently for each queue. Section 2.5.2 discusses the basis for classification and the various queue types. Section 2.5.3 discusses traffic limiting, shaping and congestion avoidance to ensure effective network utilization.

2.5.1. Differentiated service model

The *differentiated service model* [6] separates traffic flows into a more manageable structure, called classes, for the purpose of guaranteeing a certain system performance measure per class. Figure 3 illustrates how traffic is separated and managed by this model. Traffic entering the DiffServ enabled node is *classified*, *marked*, and *shaped* at the edge routers.

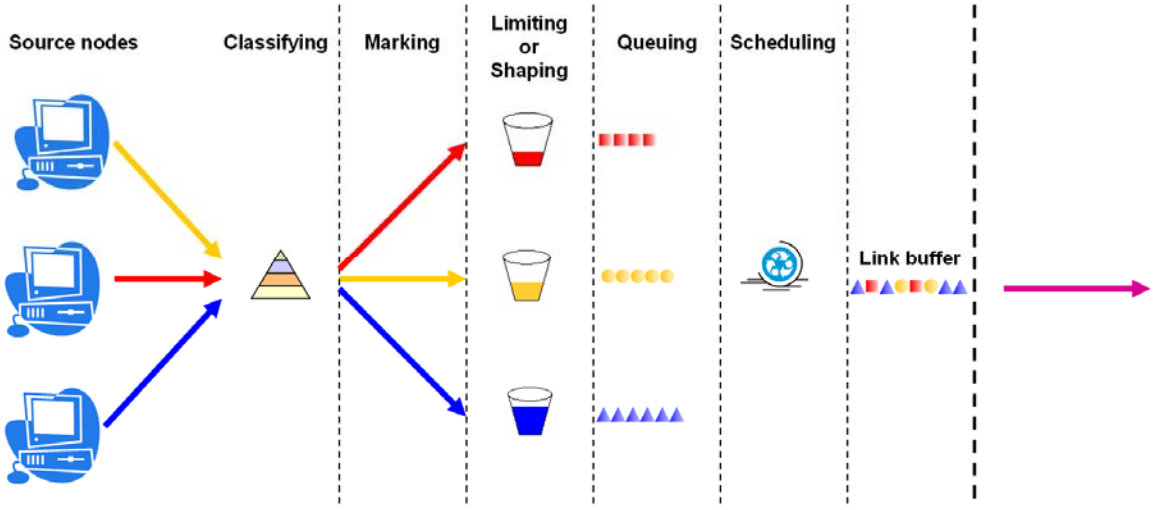


Figure 3 Differentiated service model

Classification occurs by the use of access lists which identify traffic types according to port numbers or host source addresses. Marking occurs after traffic is classified to a certain value, as specified by the service provider. Shapers prevent traffic bursts from filling the outbound queues. Schedulers determine the amount of service time a queue requires above the other queues. Once the traffic is admitted to the core network, traffic classification occurs according to its marked value.

2.5.2. Traffic classification in differentiated service

Network traffic consists of three traffic service classes:

- The *expedited forward* (EF) service class that receive priority over all traffic such as for voice or multimedia applications. This class must have minimal delay and packet loss measures,
- The *assured forward* (AF) service class that needs to be transmitted successfully to the end host. Packets from a *terminal access controller access control system*[1] (TACACS) may experience some delay, but with little to no packet loss, and the packets must reach the end systems' destination.
- The *best effort* (BE) service class that does not have any strict bounds regarding delay and throughput performance measures, and therefore does not have service guarantees.

DiffServ separates traffic with the aid of classification and marking mechanisms such as policers and shapers. *Classification* and *marking* maps the QoS parameters of the incoming packet into the *quality of service* (QoS) *per-hop-behaviour* (PHB) that is the policy and priority applied to a packet passing through a router. QoS PHB is supported by the corresponding DiffServ domain, while providing the same grade of service.

[1] A *Terminal Access Controller Access Control System* (TACACS) is a remote authentication protocol that is used to communicate with an authentication server commonly used in UNIX networks. TACACS allows a remote access server to communicate with an authentication server in order to determine if the user has access to the network.

2.5.3. Traffic conditioning in differentiated service

DiffServ routers condition traffic by *metering*, *dropping*, *queuing* and *scheduling* of network flows within the DiffServ domain [6]. *Metering* measures the rate of a traffic aggregate which uses the aggregation data as a statistic to decide on the streams' level of conformance according to certain rules. *Packet dropping* occurs when the router queues cannot support the amount of arriving traffic before service. *Schedulers* decide which queue or class needs to be serviced for transmission by the node.

Traffic shaping or limiting

Shapers or policers (limiters) manage the way that bandwidth should be allocated to an interface for a specific traffic flow [6]. Policers are typically configured based on traffic characteristics, traffic types, the packet's origins, the packet's destinations, and the packet's IP precedence settings (colouring of traffic). *Shapers* attempt to smooth the traffic to meet certain rate requirements whereas *policers* only prevent traffic from exceeding an average transmission rate. Leaky buckets (shapers) allow burstiness, while token buckets (policers) limit burstiness for network traffic streams.

Congestion avoidance

Active queue management algorithms are deployed on routers to randomly drop arriving packets to prevent buffer sizes from becoming excessively large. A difficulty in queue management is that, even though the queue for an outbound interface is not completely full, the packet is discarded. These algorithms are used to reduce the rate at which *transmission control protocol* (TCP) connections transmit data, and to discard other traffic types, which share the same queues, to ensure that queue overflow does not occur. Empirical data shows that the "early" dropping of packets reduces packet loss in a TCP traffic environment found in the internet [30].

2.6. SCHEDULERS

Schedulers provide a means for the selection of packets from various service queues. The scheduler then places different class packets on a link buffer by priority or service time, then transmit the packets over a physical medium. Figure 4 is a visual interpretation of the operation of a scheduler.

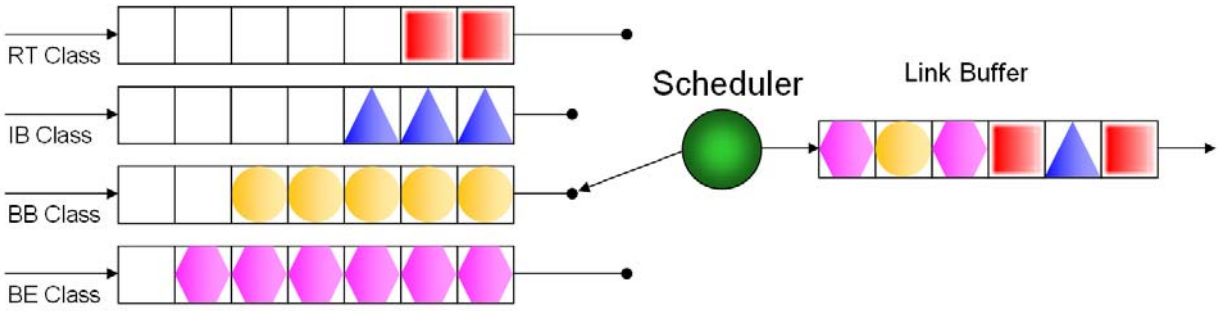


Figure 4 Packet scheduling

Essentially, scheduling creates a policy of isolation and sharing [31] [32]. Scheduling is employed as resources, for instance limited link capacity are scarce. Scheduling satisfy some constraints and optimal constraints, for instance queue delay or loss. Schedulers are designed to adhere to the constraint,

$$\text{Link Capacity} = \sum_{n=1}^{N} \rho_n q_n \tag{2.3}$$

where N is the number of queues that needs to be serviced, q_n is the scheduling waiting time in terms of bits per second, and ρ_n is the mean link utilization of the queue, defined by $\mu_n\lambda_n$, where μ_n is the waiting time and λ_n is the service time.

2.7. SUMMARY

This chapter discussed the need for quality across a service provider network and the methods that are used to analyse a network. It can clearly be seen that there are multiple factors that influence the service levels required by customers across a large service provider backbone.

The following chapter details the mechanisms that are utilized to achieve service quality across an IP network. This includes a study of the protocols that were used previous to IP protocols and methods to reduce data size across the backbone.

CHAPTER 3 QUALITY OF SERVICE

This chapter's purpose is to discuss the various aspects within a network that will have an impact on offered customer services. An introduction discussing the customer service needs from a service provider is given in section 3.1. The requirement on how to address service delivery priority to network packets for applications is then given in section 3.2. The internet protocol as architected to support network traffic priority for application services above best effort is given in section 3.3. Once certain packets receive priority over best effort, the next step is to transmit data more effectively to increase throughput as discussed in section 3.4. Some applications, such as management applications, require network delay to be at a minimum. Thus, a discussion on delay is given in section 3.5. Other applications, such as voice implementations, are sensitive to the time between packet arrivals at the remote endpoint. This type of delay is discussed in section 3.6. Data file transfer applications rely on the network protocol to determine its transfer rate to the remote system through packet loss. The ways in which packet loss occur is discussed in section 3.7. Finally, the mechanisms to support the service quality requirements are discussed in section 3.8.

3.1. INTRODUCTION

Telecommunication networks in the past existed as a public circuit-switched network to support voice over dedicated temporary links. These networks were implemented to connect customers directly, which by its nature guarantees full service to connected customers. Circuit- switched services are limited, in the fact that only a limited number of customers can be supported. This has only recently, over the past few years, become a concern to telecommunication service providers.

With the evolution of the telecommunications' network, the requirement to support a large number of customers, as well as the support of different data types across the network became a concern. To implement such a packet-switched delivery system on a classical communications network, the service providers had to provide a separate network for carrying data packets between customers. The limitation of the number of customers as well as bandwidth scalability was overcome by these networks. However, for a long time packet switched networks were used purely for file transfers as well as sending e-mails that is not bandwidth intensive.

It became costly to support two different networks, thus the need to converge these networks became a priority. As public networks extended to support more customers as well as a diverse number of applications, currently a great emphasis is placed on guaranteeing services to customers. The convergence of different types of services gives rise to *new generation networks* (NGN) [8]. New generation networks aim to provide service anywhere, at any time, for an unlimited number of customers.

Since customers' needs for service differ, a need to separate the services became evident. Convergence of different service types lead to service separation, which differs from customer to customer. It is necessary to separate the various types of services in order to have different guarantees per service. Various techniques to separate data needs and services are discussed in the following chapter.

3.2. THE QoS CONCEPT

An organization's success is based on its communication network infrastructure. These networks must provide *secure*, *measurable*, *predictable* and *guaranteed services* [33]. A customers' demand for a certain guaranteed service level lies in the fact that a variety of applications and data, such as high-quality video and real-time voice, has a low tolerance for packet delay and loss. QoS is guaranteed by the measurement and efficient management of *bandwidth, delay,* and *packet loss* parameters, for end-to-end connections. Consequently, QoS is delivered by a number of different techniques of managing network data flows.

QoS is provided for various traffic flows across a wide network. Therefore, these flows are managed differently based on the policy for the various flows. Thus, the three main features of QoS are traffic *classification*, *queuing* and *provisioning*, each of which is detailed below.

Classification

Packets entering a network node, usually at the edge routers, are partitioned into multiple priority levels or classes of service. Classification and marking is only done at the edge routers in order to minimize computation at the core routers. The *internet engineering task force* (IETF) defined the *differentiated service model* (RFC 2474 and 2475) [6] [34] to

separate application traffic into a maximum of 64 different classes. Classifications are made by certain policies that the network administrator specifies. These policies are based on port numbers, IP addresses, *media access control* (MAC) addresses of a network card, network protocol types by using *access control lists* (ACL) and *modular QoS command line interface* (MQC) [35]. Once packet classification occurred, handling policies, such as bandwidth allocation, delay bounds and congestion management, are assigned for each class at the core of the network, which leaves packet forwarding as the major function performed in the core.

Queuing

Queues are used to *separate traffic flows* which are treated differently in a manageable manner. A single queue results in unmanageable packet loss and delay variations. Thus, by using multiple queues these measures can be controlled more appropriately. Voice packets are placed into a strict priority queue, namely *low latency queue* (LLQ) [36], on the egress interface to separate these packets from the data packets. For instance, if a large amount of data downloads or large e-mails fill the queues, the LLQ scheduler puts a higher priority to the voice and multimedia applications. The other data queues are handled by a *class based weighted fair queue* (CBWFQ) scheduler [35].

To avoid that queue back-logging becomes too large, or congested, congestion avoidance algorithms are used to manage buffer overflow. The most widely used queue management algorithm is the *weighted random early detection* (WRED) [7] algorithm, which is detailed within chapter 4 section 4.9.2.

Provisioning

Provisioning provides a means to *calculate* and *allocate bandwidth* for the various queues on a DiffServ enabled router. For wide area networks, not more than 75% of the provisioned bandwidth can be used by the total combined application bandwidth, while the remaining 25% of the provisioned bandwidth is used for layer 2 link protocols, layer 3 routing protocols, and application overhead [37]. Bandwidth is provisioned across a network backbone to minimize delay and loss bounds for the various traffic classes.

3.3. RESOURCE ALLOCATION TECHNIQUES

The Internet was originally designed to be a best-effort service network, with no traffic regulation or reliable data transfers. Therefore, no admission control or service guarantees were set for any type of packets delivery, since the amount of data was tolerable for packet loss and delay. In the current Internet age, there is a diversity of applications to support, which gives rise to the concern of packet loss and delay bounds for the various application requirements. Multimedia applications, which support voice and video, require that packet loss and delay be kept small, since the data becomes inadequate if there is too long packet delays and excessive packet loss.

This section discusses various efforts to guarantee service across a wide area network. It describes the way in which the Internet developed over the past few years.

RFC 791

The *IP* protocol, *RFC 791* [41], was standardized in an effort to guarantee some measure of service to network packets. Based on the definition of a *type of service* (ToS) field, which is part of the IPv4 header that is discussed in Chapter 4 packet are prioritized and serviced according to importance. Figure 5 shows how the ToS header is defined.

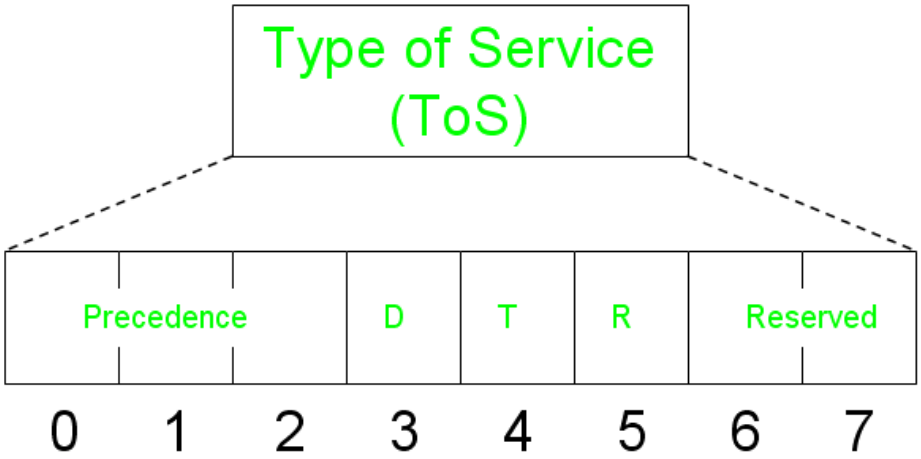


Figure 5 IP's ToS field

The IP packet type of service bits represent the following:

- *Precedence bits* (Bits 0-2): Packet priority or precedence is set by these bits. High priority bits ensure the importance of the packets traversing through a network.
- *Delay bits* (D, Bit 3): This field specifies whether a packet has normal or low delay.
- *Throughput bits* (T, Bit 4): This field specifies whether a packet has normal or low throughput.
- *Reliability bits* (R, Bit 5): This field specifies whether a packet has normal or low reliability.
- *Reserved bits* (Bits 6-7): In practice, there were some implementations that used these bits. Generally, these bits never served any purpose.

IntServ

The *integrated service model* (IntServ) remaining unchanged for 20 years was the first model to be used above traffic with no priority over other traffic types [42]. The IntServ architecture, however, only extended to the delivery service of IP with the use of different components and mechanisms. Therefore, IntServ provided a means for bandwidth allocation while restricting usage of resources to only a single class. The assumption that was made is that resources needed to be allocated to a user. Therefore, a requirement for resource reservation and admission control was needed. Resource reservation was made possible by the use of the RSVP signalling protocol described in RFC2205 [38]. Today, only *local area networks* (LAN) use the IntServ model for the allocation of resources to nodes connected to the network.

The following factors limited IntServ to large IP networks [39]:

- Because of resource reservation, this model is not scalable since each flows' state needs to be maintained separately for a single class, which is a challenging task for a large number of flows. High-speed networks support over a million connections, which makes state maintenance nearly impossible.
- Applications at all nodes need to support IntServ signalling protocols. This implies that the RSVP protocol needs to be supported on all applications and devices.
- All nodes between end-to-end users need to be IntServ enabled as shown in Figure 6.

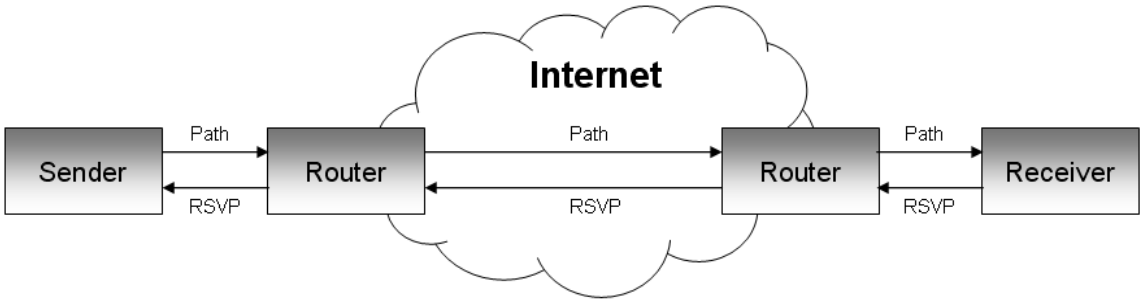


Figure 6 Integrated services model

DiffServ

The *differentiated service* m*odel* (DiffServ) [6] classifies and marks network traffic into smaller, more manageable groups at the edge of the DiffServ enabled network by lifting the computational burden from the backbone routers. The DiffServ mechanism employs a *differentiated services code point* (DSCP) [34] value to profile traffic into manageable classes which receives different priorities. At the network backbone, appropriate bandwidth and delay guarantees can be assured for the different classes of service, since packets are only forwarded based on QoS requirements. Unlike IntServ, DiffServ does not reserve resources, which enables the DiffServ model to be highly scalable to large IP networks.

The three traffic profile classes are defined to have different performance measures:

- *Premium service* (EF): This service class handles delay and loss sensitive traffic, while maintaining a bandwidth for applications such as voice traffic. The premium service queue is handled with the low latency queuing (LLQ) priority queue scheduler [36].
- *Gold, silver and bronze services* (AFxy): These service classes handle applications such as e-commerce, webpage and various other applications that do not require stringent delay or loss guarantees. Network traffic marked gold receives 50%, silver receives 30% and bronze receives 20% of the available link bandwidth. These service queues are handled with the class based weighted fair queue (CBWFQ) priority queue scheduler [35].
- *Best effort*: This traffic class receives the least amount of priority since data does not need any delay, loss, or bandwidth guarantees.

Enhancements on IntServ

RSVP allows only one single reservation per class to be allocated. Therefore, the *RFC 3175* [40] provided a means to aggregate RSVP reservations for a single, shared class across a large IP network. The mechanisms dynamically allocate reservations, place specific traffic reservations according to the applicable reservation, calculate bandwidth requirements per reservation, and free bandwidth for terminated reservations.

RFC 3175 address the scalability to large IP networks in the following way:

- By limiting the number of large reservations, such that the number of reservation states that needed to be maintained is reduced, as well as reducing the number of signalling exchanges between network devices.
- By extending traditional RSVP flow classifications to the *differentiated services code point* (DSCP) in order to identify aggregated flows in the backbone network routers.
- Combining the aggregated streams on the output port, so that packet queuing and scheduling are simplified.

MPLS Traffic Engineering

Multiprotocol label switching (MPLS) provides convergence between various technologies by combining routing and switching performance for layer 3 and layer 2 technologies [43]. Service providers obtain a means to provide frame relay, asynchronous transfer mode (ATM) [33], ethernet, point-to-point protocol (PPP) and high-level data link control (HDLC) [20] with a variety of speeds and granularity levels for a technology rich network. Thus, this technology benefits pure IP, IP over ATM, or mixed layer 2 technologies [33]. Therefore, MPLS traffic engineering provides scalability to VPNs, end-to-end QoS, as well as efficiently utilizing network nodes with network growth. MPLS is used with the support of DiffServ to improve scalability, end-to-end services with simpler configuration, management, and provisioning for network traffic.

3.4. THROUGHPUT

Throughput is the amount of data that is transmitted per second to a destination host within the protocol overhead information, such as start and stop bits, TCP/IP overhead, and hypertext transfer protocol (HTTP) headers [44]. Factors that affect throughput are packet size, payload size, latency, hop count and link bandwidth. To calculate the throughput for a link, the number of bits that can be transmitted across the network link within a certain amount of time is observed. Thus, by transferring a large file, typically in the range of Megabits, the file size is divided by the amount of time it took to transfer the file. Another means of calculating throughput is by utilizing the simplified equation (2.1), mentioned in Chapter 2.

Factors that influence throughput include [45]:

- *Compression*: This feature can either be implemented by hardware or software, such as some analogue modems and popular operating systems. Compression improves throughput due to the reduction in the number of bits transmitted from the host. Usually, compression is done without the users' awareness which affect throughput measurements of a compressed or an uncompressed file. Theoretically, it would take 8.192 seconds to transfer a 64 KB file over a 64 Kb/s link, but 6.125 seconds for the same file compressed over the same link.
- *Data formats and overheads*: More overhead is placed on the data when data bits are similar to the flag bits, thus, making *data checking* necessary to validate the data authenticity. These frames use flag bits to indicate the *beginning* and the *end* of each frame, where the beginning and end bits are a unique bit sequence. *Bit- or byte-stuffing* is a technique to ensure that data is not read as signalling flag or control bits. To ensure that data contains no errors, the *frame check sequence* (FCS), such as CRC-16 and CRC-CCITT, is used to calculate a bit sequence, from the address, control, and data fields. As a result, lost bits, swapped bits and extraneous bits can be detected after frame transmission. HDLC frames consist of a *maximum address size* of 32 bits, a *maximum control size* of 16 bits, a *frame check sequence* of 16 bits, and *overhead* of 64 bits.

- *Higher level protocols*: These protocols place data into the data fields of HDLC or PPP frames. The protocol layer is responsible for formatting the data inside these fields. The IP protocol is the most commonly used medium of data transfer, adding a header to data, referred to as the payload.

3.5. DELAY

As the global network infrastructure grows and connects more clients to a wide variety of services, congestion occurs. Congestion, which occurs when the number of packets in the buffers grows larger than the buffer sizes, is the primary cause of delay between two communicating nodes. The factors that influence delay are *data transmission* discussed in section 3.5.1, *serialization transmission* discussed in section 3.5.2, *propagation transmission* discussed in section 3.5.3, and *queuing transmission* discussed in section 3.5.4, and *data compression transmission* discussed in section 3.5.5 [46]. These factors are discussed briefly in the following subsections in order to illustrate where the various types of delay measures are applicable in the overall end-to-end delay as discussed in section 3.5.6.

3.5.1. Transmission delay

Transmission delay, also known as *store and forward delay*, is experienced once all packets are received at a router and forwarded to the next router. This delay thus is the amount of time that the router needs to transmit all the packet bits to the next node. Therefore, the amount of transmission delay introduced to the data is calculated by

$$\text{Transmission delay} = \frac{L}{R} \text{ ms} \tag{3.1}$$

where L is the packet length in bits, and R is the link bandwidth between two routers in bits/second. The amount of transmission delay introduced is for practical reasons zero, since values are typically in microseconds, thus not adding a significant amount of delay to the system.

3.5.2. Serialization delay

Serialization delay, also known as *handling delay*, occurs due to the placing of packets into their appropriate queues. Thus, serialization delay is calculated as

$$\text{Serialization delay} = \frac{F}{C}\,\text{ms} \tag{3.2}$$

where F denotes the frame size and C denotes the interface clock speed. For example, 1500 byte frames, thus 21000 bit frames, with a clock speed of 56-kbps has a delay of 214 ms per frame to serialize the bits into the queues. Table 1 gives a detailed breakdown of serialization delay for different line speeds of various packet sizes.

Table 1 Link speed vs. packet sizes

Packet size \ Physical link speed	DS-1 1.5 Mb/s	DS-3 44 Mb/s	OC-3 155 Mb/s	OC-12 620 Mb/s	OC-48 2.5 Gb/s	OC-192 1 Tb/s
40 bytes	0.2074 ms	0.0072 ms	0.0021 ms	0.0005 ms	0.0001 ms	<0.0001 ms
256 bytes	1.3264 ms	0.0458 ms	0.0132 ms	0.0033 ms	0.0008 ms	0.0002 ms
320 bytes	1.6528 ms	0.0572 ms	0.0165 ms	0.0041 ms	0.0010 ms	0.0003 ms
512 bytes	2.6528 ms	0.0916 ms	0.0264 ms	0.0066 ms	0,0016 ms	0.0004 ms
1500 bytes	7.7720 ms	0.2682 ms	0.0774 ms	0.0193 ms	0.0048 ms	0.0012 ms
4470 bytes	23.1606 ms	0.7994 ms	0.2307 ms	0.0575 ms	0.0144 ms	0.0036 ms
9180 bytes	47.5648 ms	1.6416 ms	0.4738 ms	0.1181 ms	0.0295 ms	0.0074 ms

3.5.3. Propagation delay

Propagation delay is the amount of time that a single bit needs to propagate between router interfaces, in other words, on a network link. This parameter depends on the physical medium that is used to connect network routers to each other. Equation (3.3) shows how to calculate propagation delay,

$$\text{Propagation delay} = \frac{D}{S}\,\text{ms} \tag{3.3}$$

where D denotes the distance between the connected routers and S denotes the propagation speed of the link.

The *propagation speed* of copper links allow electrons to be transmitted at 2×10^8 meters per second while fiber links transmit light at 3×10^8 meters per second [47]. For large IP networks, propagation delay is in the order of milliseconds which can be neglected from analysis.

3.5.4. Queuing delay

This dissertation deals with how queues affect network traffic for highly utilized links. *Queuing delay* is the amount of time that packets wait in their respective queues before being placed on the output link buffer. This parameter depends on how much the router is utilized, depicted in Figure 7. Queuing theory and applications are described in detail within [28], [29] and [48].

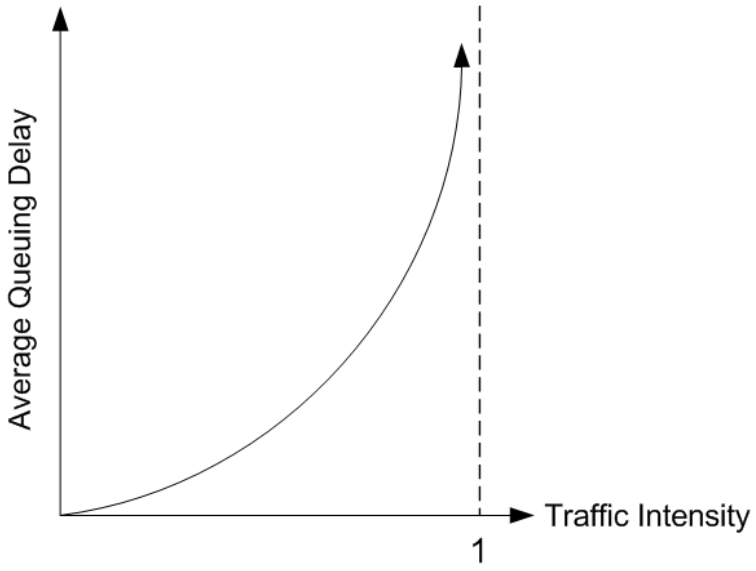


Figure 7 Queuing delay vs. traffic intensity

It is clear that the queuing delay depends on the node utilization, also known as the *traffic intensity*, *traffic inter-arrival times* and the *link transmission rate*. *Queuing delay* differs from packet to packet, depending on the packet size. For example, if a large number of packets arrive at a queue at the same time, the first packet to arrive at the queue will suffer no queuing delay, whereas the last packet will suffer a large delay. Therefore, queuing delay is characterized by statistical means such as average queuing delay, queuing delay variance, and queuing delay exceeding some value.

3.5.5. Compression delay

Compression provides a means to reduce the number of data bits that are transmitted over a network. Hence, compression may reduce the overall delay measures mentioned above. The most popular compression formats used for web applications are GIF, MPEG and JPEG, while text files use word-based compression algorithms [49]. Generally, files are compressed at the workstation, which does not burden hardware with compression techniques. However, hardware devices that handle compression usually offer encryption for security purposes.

3.5.6. End-to-end delay

The previously mentioned delay measures are discussed for a single node. *End-to-end delay* constitutes the total amount of delay experienced between remotely placed devices. Figure 8 illustrates the end-to-end delay from the ingress router to the egress router. As can be expected, the delay for each node will vary. Therefore, the end-to-end delay will be different for each packet.

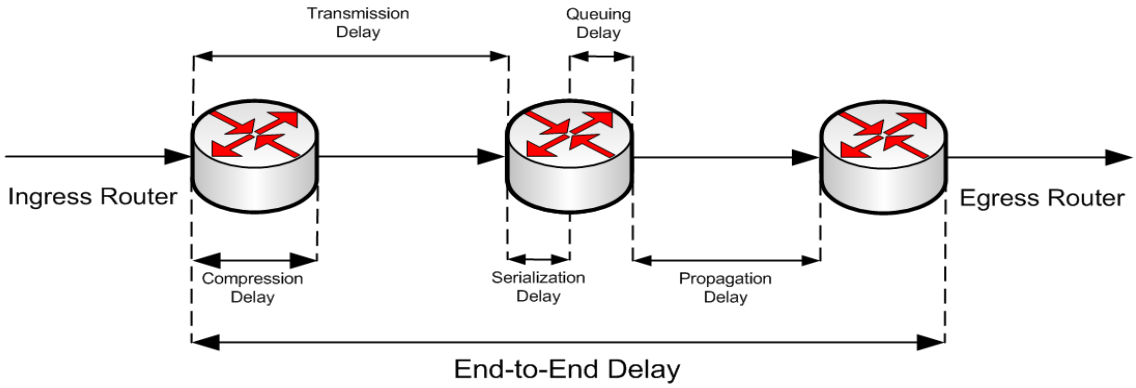


Figure 8 End-to-end delay

3.6. JITTER

Jitter is the time difference between packets arriving at a receiving node [3]. This delay affects voice streams, causing discontinuities between expected arrival times. To compensate for this phenomenon of delay, a playout buffer is used to smooth out the voice stream. The *real-time transport protocol* (RTP) [50] encapsulation method for voice is used for playout

control to smooth the jitter effects. The delay variation for IP packets is known as *IP delay variation* (IPDV) [3], which serves an important measure for real-time multimedia applications. The effect of jitter is shown with Figure 9.

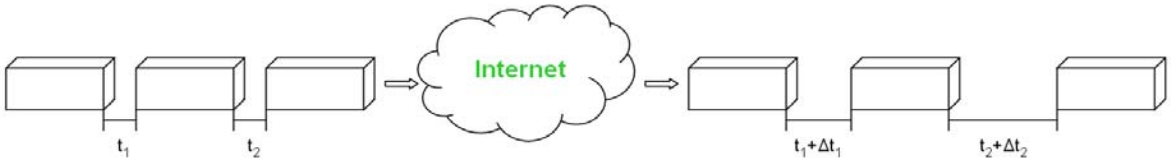


Figure 9 Jitter (IPDV)

3.7. PACKET LOSS

Packet loss occurs when the queuing buffers are too small to handle incoming data rates, consequently resulting in congestion. By the use of buffers, packet delay reaches infinity slower as traffic intensity approaches one. When the queues reach capacity, packets are lost due to the fact that the router has no storage space for arriving packets and, consequently the packets that overflow are discarded. As traffic intensity increases, packet loss increases, which is an important performance measure in conjunction with delay. Lost packets are retransmitted, either by the application or the transport protocol.

The four situations that cause packet loss are [51]:

- *Drop tail*: When queues are filled, no more packets can be stored, which results in packets being dropped at the input interfaces illustrated by Figure 10, where the various colours present different sources.

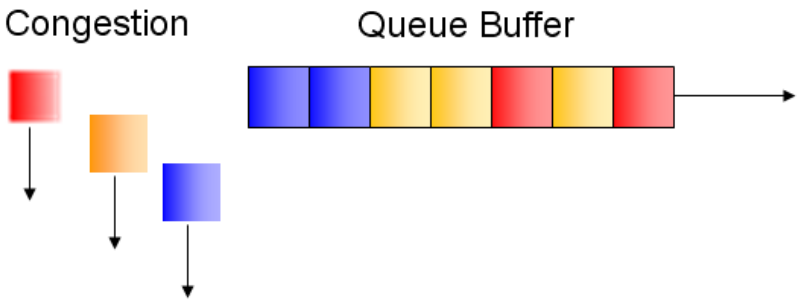


Figure 10 Queue tail drop effect

- *Discarding packets*: When the queue size reaches a certain threshold, packets are randomly dropped based on precedence on the incoming link. Figure 11 is a graphical illustration to illustrate the idea behind the concept.

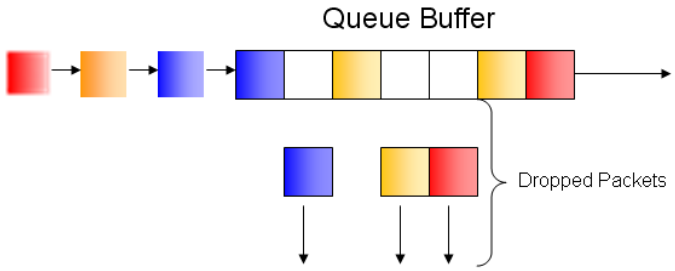


Figure 11 Packet dropping

- *Packet corruption*: A packet becomes corrupted once the transmission medium creates errors in the bits being transmitted to the following node, as illustrated in Figure 12. *Cyclical redundancy check* (CRC) checks will detect if a packet is not healthy, which will result in the packet being discarded.

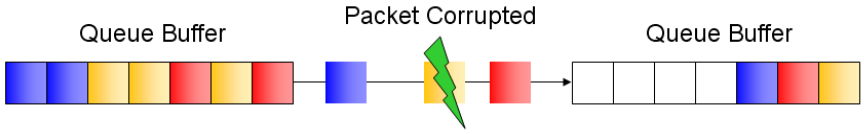


Figure 12 Packet corruption

- *Link failures*: Once the physical medium is not available while packets are being transmitted, the packets carried on the line, as well as queued packets are lost as illustrated in Figure 13. It is up to the routing protocol to signal connected devices to reroute data transmissions

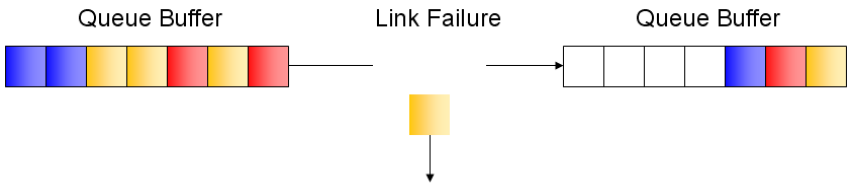


Figure 13 Link failure

Packet loss in an IP network should not be avoided, since it leads to desired effects. The TCP protocol for instance, aims to utilize the maximum available bandwidth for each session flow, which results in node over-utilization and congestion. This protocol has a defined congestion window, initially transmitting slowly and gradually increasing packet transmission rate until packet loss occurs. In the event of packet loss, the TCP sessions decreases its transmission rates, since packet loss indicates congestion. After some time, all sessions would have settled into a maximum transmission window that it can utilize across the network. All packets that did not receive acknowledgements are retransmitted until they are acknowledged. TCP protocol dynamics are discussed in detail within chapter 4.

3.8. QoS MECHANISMS

This section details network traffic rate limiting, scheduling, as well as congestion control that provide service quality across a networking infrastructure. Traffic rate limiting is achieved by the use of a *single-rate three-color marker* or a *two-rate three-color marker* which marks traffic when congestion occurs. Network traffic is dropped if marked. However, packets can still be serviced by the use of a shaper to buffer packets received in excess. The means to mark network packets is discussed in section 3.8.1. The time that a queue receives service is decided by a scheduler. If a scheduler can not provide a queue sufficient service time, then congestion needs to be controlled. Scheduling is discussed in section 3.8.2. When a queue is filled with packets the congestion avoidance algorithm will notify the sources of the congestion. The way that a source is notified of congestion in discussed in section 3.8.3.

3.8.1. Marking, policing and shaping

Network traffic tends to be bursty, which results in congestion, and as a consequence packet loss. The first proposal to limit traffic burstiness was made by the ATM forum with the implementation of a leaky bucket. Turner [20] proposed and studied the affects of shaping network traffic. *Traffic policers* limit traffic rates, which eliminates network traffic from being bursty. *Leaky buckets*, also known as *traffic shapers*, controls access to available bandwidth by ensuring conformance to policies and regulate traffic flow rates for congestion avoidance. Improvements to policers allow some traffic burstiness were made possible by the implementation of leaky buckets. Elwalid et al. [21] studied the effects policers have on multiple data steams for a single node, which gave rise to the *adversarial model*.

Figure 14 is an illustration of arriving traffic, which shows signs of burstiness at a network node. Figure 15 illustrates that traffic bursts are limited by discarding traffic that arrives faster than a set arrival rate. Figure 16 illustrates how traffic is shaped by adding delay to bursty network traffic.

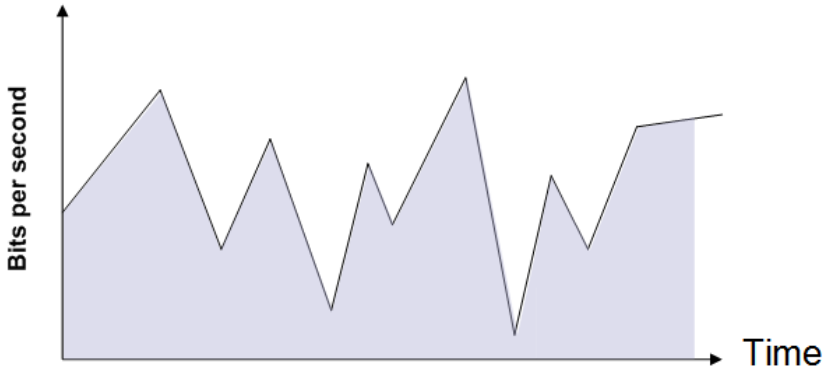


Figure 14 Original traffic arrival rate

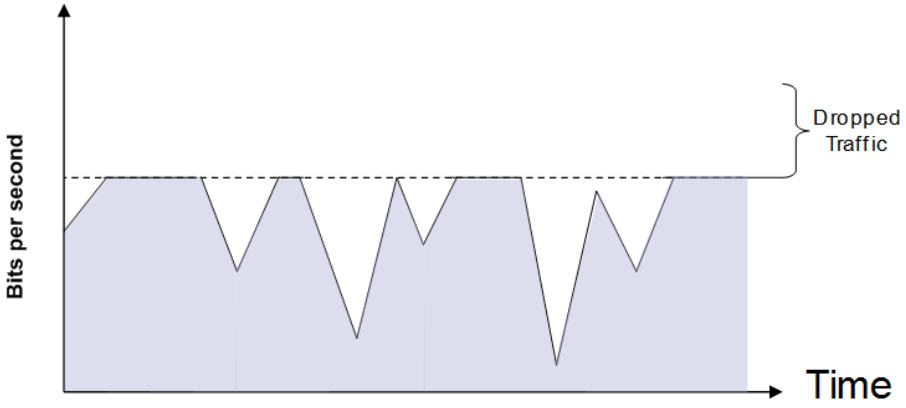


Figure 15 Policing effect on traffic departure rate

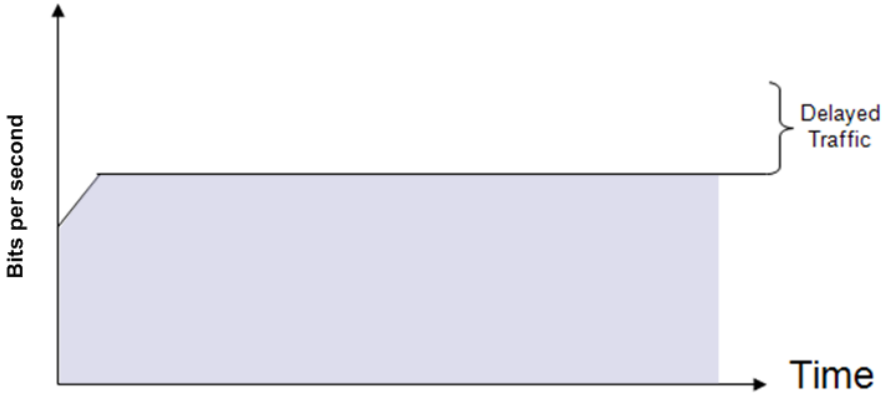


Figure 16 Shaping effect on traffic departure rate

The following section describes policing and shaping methods which prevent queues from being flooded by bursty traffic. These mechanisms ensure that maximum delay guarantees are not violated across a network by greedy traffic sources.

Single-rate three-color marker

A *single-rate three-color marker* (srTCM) [52] marks IP packets as the packet enters a DiffServ enabled domain. Arriving packet streams are marked *green* if the arrival rate is less than a *committed information rate* (CIR), *yellow* if the traffic burst rate exceeds a *committed burst size* (CBS) but not an *excess burst size* (EBS), or *red* if the traffic exceeds the EBS. This policing scheme regulates traffic based on packet length, not average peak rates. The colours of the packets are indicated by the DiffServ field in a PHB manner. Packets that are marked red are not omitted to a traffic class queue and thus dropped, whereas the yellow packets are dropped if the following queue is full.

Two-rate three-color marker

A *two-rate three-color marker* (trTCM), described by [53], operates in the same way as the srTCM policer, with the exception that the token buckets operate at different rates. The token bucket marked *P* provides a *peak information rate* (PIR) and the bucket, marked *C*, provides a *committed information rate* (CIR). The size of bucket *P* is a *peak burst size* (PBS) while bucket *C* is a *committed burst size* (CBS). Initially, the buckets are full, thus, $T_p(0)=PBS$ and $T_c(0)=CBS$ after which token count T_p increments every *PIR* and T_c increments every *CIR*. When a packet arrives, the *P* token bucket is first checked to verify that there are enough tokens in the bucket to send the packet. The violating condition occurs if there are not enough tokens in the *P* token bucket to transmit a packet. If there are enough tokens in the *P* token bucket to send the packet, then the *C* token bucket is checked. The exceeding condition occurs if there are enough tokens in the *P* token bucket to transmit the packet but not enough tokens in the *C* token bucket to transmit the packet. The conforming condition occurs if there are adequate tokens in the *C* bucket to transmit the packet.

Time sliding window three color marker

Similar to the previously mentioned mechanisms, the *time sliding window three color marker* (TSWTCM) marks IP packet streams *red*, *yellow* or *green* [54]. Packet marking is based on

traffic throughput which is compared against a *committed target rate* (CTR) and a *peak target rate* (PTR). Packets below CTR are marked *green*, between CTR and PTR are marked *yellow*, and exceeding PTR are marked *red*. Packets marked red are dropped immediately, where yellow packets are dropped once the following queue is full.

Rate adaptive shaper

Rate adaptive shapers (RAS) are generally used with the srTCM and trTCM mechanisms [55]. The aim of this shaper is to limit burstiness on the output interface in order to avoid discarding packets. This shaper consists of a tail-drop first-in-first-out (FIFO) queue that empties at a variable rate. The queue is emptied as a function of the number of packets within the queue. For instance, if the queue size becomes too large, the shaping rate increases in order to prevent large delays and packet loss. The shaping rate is also a function of the average rate of the incoming traffic. The two defined RAS are the *single-rate rate-adaptive shaper* (srRAS), which is typically used upstream of a srTCM, and the *two-rates rate-adaptive shaper* (trRAS), which is typically used upstream of a trTCM.

Token bucket traffic shaper

Token buckets shape traffic generated by a source [20]. Thus, this mechanism provides rigid output patterns for leaving traffic flows at a set rate with no concern of traffic burstiness. Thus, unregulated bursty traffic on the input interface is shaped to produce smooth, regulated traffic on the output interface. This mechanism is used to police network traffic, but with the addition of a buffer, implemented as a shaper. Thus, a buffer converts bursty, unregulated traffic flows into predictable, smooth and regulated flows.

Leaky bucket traffic shaper

Leaky buckets address the limitation of token buckets, by allowing the output interface to transmit data at higher speeds, thereby, allowing bursty traffic to be transmitted [20]. Instead of regulating the output buffer, tokens fill up the bucket at a constant rate. Each packet that enters the bucket obtains a token which allows the packet to pass through. However, if there are no available tokens, packets wait until a free token is generated. Bursts are allowed when enough tokens are available for the amount of data that needs to pass through.

3.8.2. Queuing scheduling

Queuing theory is the most important aspect of IP networks as it provides proper insight into network traffic analysis [29]. In order to understand the more complex queuing techniques, a thorough discussion of simple queuing techniques is provided below.

FIFO

The *first-in-first-out (FIFO)* queue, also known as the *first-come-first-served (FCFS)* queue, is the most basic queuing technique to understand. Figure 17 illustrates the model for a FIFO link scheduling discipline. Arriving packets wait in a queue while the server is busy transmitting another packet. But the arriving packets are discarded if the queue is full. If an active queuing management system is implemented, packets that fill the queue might be discarded in order to make space for other arriving packets.

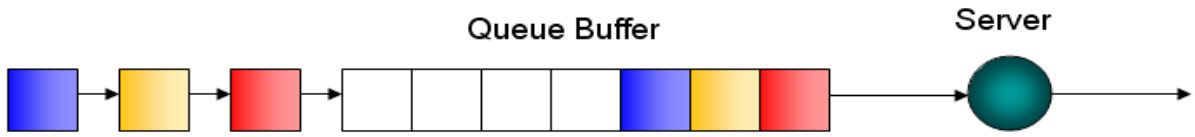


Figure 17 FIFO queue

Figure 18 is an example of how the packets are treated within a FIFO queue. Because the queue is based on the FIFO queuing discipline, packets leave in the same order as they arrived. The queue remains idle if not in use, as can be seen with packet 4.

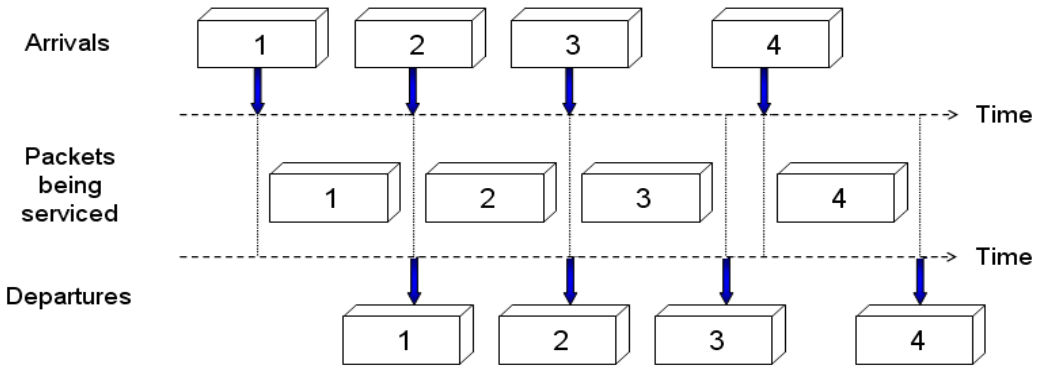


Figure 18 Logical FIFO queue operation

PQ

The *priority queue* [36] policy places packets which are classified to different queues on the output link buffer according to queue priority. Packet priority depends on the colour the packet is marked with, as will be discussed with the DiffServ model in chapter 4. Generally, each priority class has its own queue, where each class is assigned a priority with regards to other queues. Figure 19 illustrates that packets are classified into different queues, which is defined by the user.

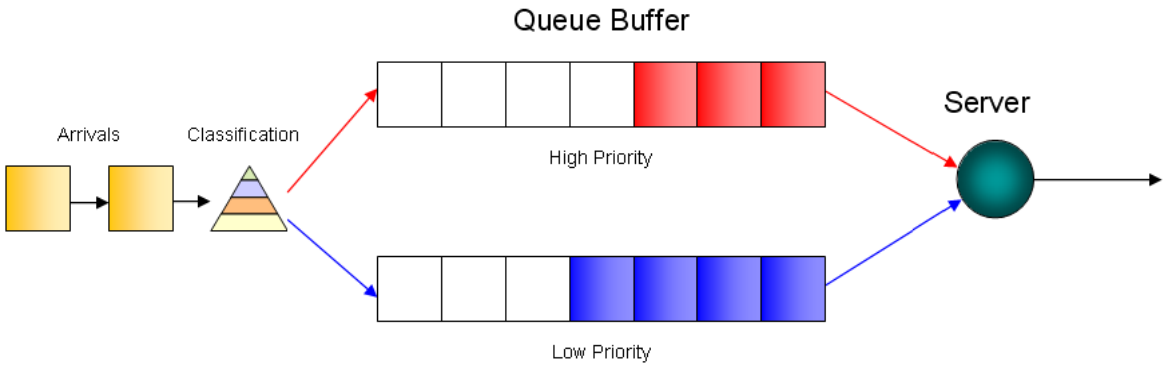


Figure 19 PQ queue

Higher priority classes enjoy service above the lower priority classes, as shown in Figure 20. Service time for lower priority packets is not interrupted with the arrival of a higher priority packet.

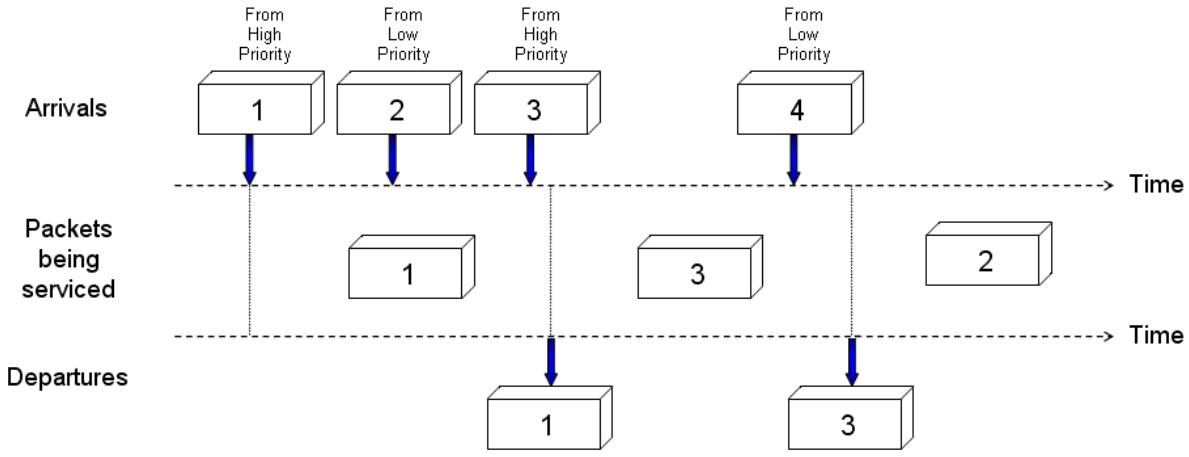


Figure 20 Logical PQ queue operation

RR and WFQ

The *round robin queuing* (RR) discipline alternates between different service classes, transmitting packets regardless of queue priority. Thus, this queuing technique can also be represented by Figure 21, with $w_1 = w_2 = \ldots = w_i$. The *weighted fair queuing* (WFQ) discipline is a special round robin discipline, adding weights to the different queues in order of precedence. Figure 21 illustrates that each class is assigned a certain weight, denoted by w_i, which determines packet precedence to the server.

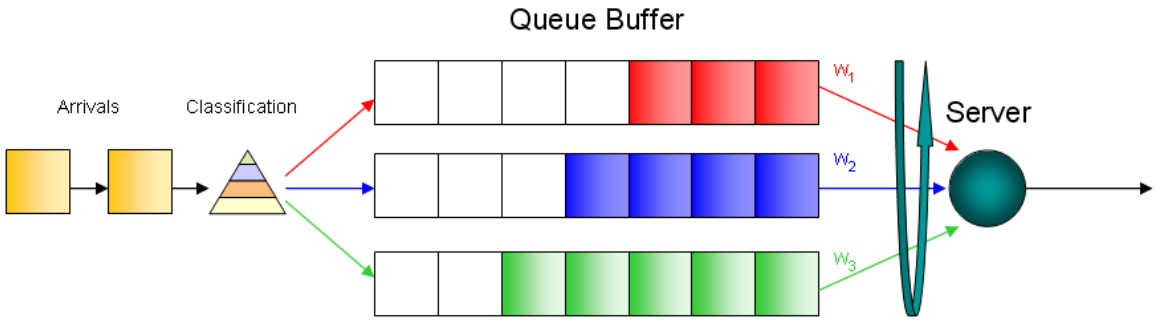


Figure 21 RR/WFQ queue

WFQ differs from RR in the sense that each class receives a differential amount of service for any given time interval. Figure 22 shows the order in which RR places packets onto the output buffer.

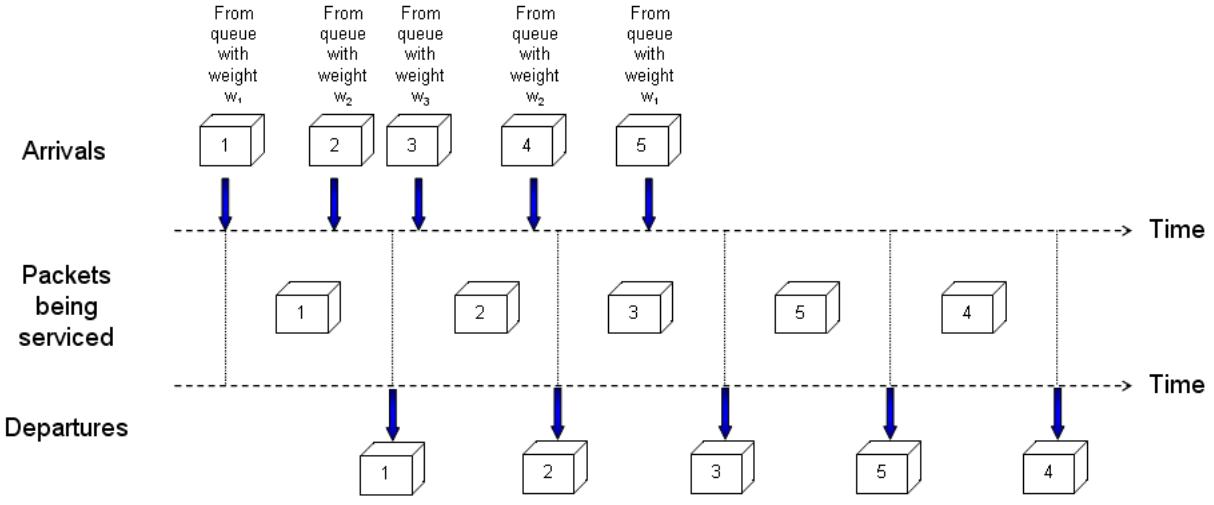


Figure 22 Logical RR/WFQ queue operation

In the case of WFQ, when no more packets remain within a queue that can be placed on the output buffer, the scheduler checks the next queue to be serviced. Therefore, no idling time is allowed between packet transmissions on a certain queue. Such a discipline is known as a work-conserving discipline, which processes data for as long as there is data to be processed. Throughput for a given class can be determined using

$$B(p) = R * \frac{w_i}{\sum w_j} \tag{3.4}$$

where R is the departure rate, w_i is the various queue weights, and $\sum w_j$ the collective queue weights.

3.8.3. Congestion avoidance

Congestion occurs when the amount of arriving traffic at a network node becomes too large for the buffers to store. In order to keep the buffer length to a minimum, packets are discarded using packet dropping algorithms, for example [7]. These congestion avoidance algorithms use the *TCP explicit congestion notification* (ECN) mechanism to inform sources to transmit at a lower data rate. Hence, the transmission rate is adjusted according to the TCP transmission protocol, which decreases its congestion window size. The RED mechanism takes advantage of TCP mechanisms to reduce transmission rates. UDP is not designed to be aware of congestion; therefore, delaying or dropping packets does not influence the transmitting sources.

3.9. CONCLUSION

This chapter detailed the mechanisms which are used to ensure QoS within a network. Queuing theory is used to provide statistical analysis to predict network behaviours. This dissertation is concerned with the effects of congestion avoidance with the understanding what other measures there are to provide QoS.

The following chapter provides a study of protocols used within a service provider backbone network.

CHAPTER 4 IP BACKBONE ARCHITECTURE

This chapter provides a study of IP network protocols and the evolution from only providing a means to distribute information to the guarantees across multiple traffic types. Through this study it can be seen how traffic is controlled by the relevant protocol that is used to transport data between hosts. The previously discussed QoS mechanisms can in some cases utilize the inherent properties within a protocol to control traffic flow. This chapter begins with the network services required within a service provider network in section 4.1. Layer 2 protocols used on *local area networks* (LAN) switch packets between each connected node are discussed in section 4.2. Various layer 3 protocols used to direct traffic across a *wide area network* (WAN) are discussed in section 4.3. Network packet presentation and construction are then discussed in section 4.4. The way in which network routing nodes are connected by transport technologies is discussed in section 4.5. The two most frequently observed types of packets across a service provider network are discussed in section 4.6. Various upper-layer application relevant protocols are then discussed in section 4.7. *Multi packet label switching* (MPLS) is effectively used across a service provider network and discussed in section 4.8. Finally, the congestion control mechanism for this dissertation is discussed in section 4.9.

4.1. INTRODUCTION

Telecommunication networks provide services to a large number of hosts, which, along with the sites they support, grew exponentially fast, over the past ten years. Customers were given the opportunity to publish and expand their business opportunities to a whole new level. The *world wide web* (WWW) consists of a number of innovations, such as HTTP, TCP and the IP protocol stack, which made connectivity between various hosts possible [20]. The “plug-and-play” and “easy-to-use” approaches are a driving force for numerous opportunities and business ventures.

Hardware depends on protocols to ensure that data is transmitted correctly over the network medium. A protocol is a standard procedure for regulating data transmission between nodes, such as the TCP stack. The most important protocol that is supported over *IP backbones* is the IP protocol [41], which is deployed in all internet protocol stacks. Continuous research is being conducted to provide a certain level of service, namely QoS parameters, to guarantee throughput, loss and delay measures [8]. TCP is the most significant protocol that is used to

transport data over IP, where the combination of the TCP protocol and IP protocol is called the TCP/IP stack. Both of these protocols are described in the sections that follow. Other underlying protocols that IP supports are ethernet and ATM [56].

Internet services and mechanisms enable the Internet model to be heterogeneous towards various technologies. Link bandwidths are also heterogeneous since bandwidth varies between kilobits per second and gigabits per second between connected hosts. Various heterogeneous protocols, ranging from connectionless protocols, such as UDP, to connection-oriented protocols, such as TCP, and multicast protocols provide a basis for communication between hosts. Applications vary between non real-time applications, for example FTP, and real-time applications with QoS constraints, for example *voice over IP* (VoIP) [20].

The global *IP backbone infrastructure*, which supports multiple data handling infrastructures, is divided into layer 2 and layer 3 technologies. Layer 2 technologies provide packet-based infrastructures, namely *any transport over MPLS* (AToM) for MPLS-based core networks, and *layer 2 tunnelling protocol version 3* (L2TPv3) for "*Native IP*"-based core networks. Layer 3 technologies, which are router protocols, support IP based communication such as IPSec, *generic routing encapsulation* (GRE) and *multi-point label switching/border gateway protocol* (MPLS/BGP), which is used to transport IP packets over the IP/MPLS core.

The following sections describe technologies used across a network infrastructure. This knowledge provides an understanding of the complex implementation used within the simulation environment.

4.2. LAYER 2 PROTOCOLS

Layer 2 protocols, or *data link layer protocols*, are used to provide a means to format data to be transmitted over physical connections to other interconnected hosts. The protocols described below are aided by the use of multiplexing, error detection, MAC addressing and upper-layer data.

Multiplexing

Multiplexing [56] provides a means to divide data into smaller more manageable pieces to combine multiple data streams on one link. All upper-layer protocol data is encapsulated into packets at layer 2. Ethernet, which is thought of being a bottom-layer protocol, resides on layer 2.

Error Detection

Error detection [56] ensures that data is received correctly at the receiving host. Ethernet provides a *cyclical redundancy check* (CRC) to ensure that the transmitted data was transmitted correctly over the physical medium. Since the CRC is calculated over the entire contents of the frame, redundancy is greatly ensured. CRC checks are performed by network interface cards (NIC) whenever a frame was fully received by a host.

MAC Addressing

A *media access control* (MAC) [56] code is burned onto NICs, which provides a unique code for every network interface. Each NIC verifies every packet that propagates through the network looking for messages addressed to the specified host. MAC frames are used to carry upper-layer data between Layer 2 interfaces.

Layer 2 VPN

MPLS layer 2 VPNs relies on protocols such as ethernet, frame relay, ATM, *high level data link control* (HDLC), and *point-to-point protocol* (PPP) over an IP/MPLS network [56]. The driving force for these networks lies in the fact that technology can be converged to enhance business services. The services combine *MPLS fast reroute* (FRR) for *any transport over MPLS* (AToM) circuits, *quality of service* (QoS) with the DiffServ model, as well as effective bandwidth utilization with MPLS *traffic engineering* (TE).

AToM is the underlying technology supported by MPLS to encapsulate and transport layer 2 packets over an MPLS backbone. This technology enables service providers to manage private networks without the need to separate network environments by providing dedicated layer 2 connections. Thus, customers are supplied with connectivity between customer sites via a single, integrated, packet-based network infrastructure.

4.3. LAYER 3 PROTOCOLS

Layer 3 protocols, or *network layer protocols*, enable data to be routed within an IP backbone. Network bridges forward packets by examining destination MAC addresses, while routers forward packets based on the network layer addressing. Network layer addresses have their own naming method, for example IP, which is a routed protocol. Thus, each network layer has its own addressing scheme.

Broadcasting domains are separated by routers, which means that layer 2 protocols can no longer be used for communication. Thus, layer 2 protocols are used by packets to reach a network router, and then the router bases its forwarding decisions on the address within the network layer protocol. *Bridges* forward packets based on MAC addresses, thus the MAC addresses remain unchanged with transmission. *Routers* forward packets by manipulating the MAC address, which is the address of the next layer 2 device in its path, whether it is another router or the destination node. The router places its own address in the source address field and modifies the IP (layer 3) field of a packet. Since the packet's content is changed, the CRC needs to be recalculated before forwarding the packet.

Layer 3 VPN

Today, the most widely used technology to provide routing through a network is *MPLS* [43]. A peer-to-peer VPN model leverages from the *border gateway protocol* (BGP) to distribute VPN-based information which allows enterprise subscribers to share information to service providers, resulting in cost reduction and operational simplicity. Since information is distributed, value-added services such as QoS and traffic engineering enable network convergence of voice, video and data. "*Tight SLAs*", as strictly defined guarantees between customer and provider, are offered by the deployment of MPLS TE and *fast reroute* (FRR). Addressing management, which is provided by MPLS Shared Services, is outsourced by DHCP and network address translation mechanisms.

For the purpose of this dissertation, the layer 3 technology that will be focused on is the differentiated service model [34] to improve on device performance. Before discussing the differentiated services model, a thorough discussion of the network protocols is provided below.

4.4. OSI PROTOCOL LAYERS

The *international organization for standardization* (ISO) defined a model, called the *open system interconnection* (OSI) [57] model, to enable vendors with an interoperable architecture to be compatible with any other vendors. The interoperability architecture model was designed by the d*epartment of defence* (DoD) [57], but was limited in functionality to provide extensibility. The OSI model's functions provide successful communication over a network, which support distributed processing for communication.

All the layers are dependent on the services provided by the layers below it. The different layers have no knowledge of each other's functions but needs to know how the layers operate. For example, IP, at the network layer does not need to know which medium the data will be transmitted over, but needs to know how to utilize the layer 3 networking functions. Figure 23 shows how the DoD internet model maps to the more improved ISO model, as well as some applicable protocols per layer.

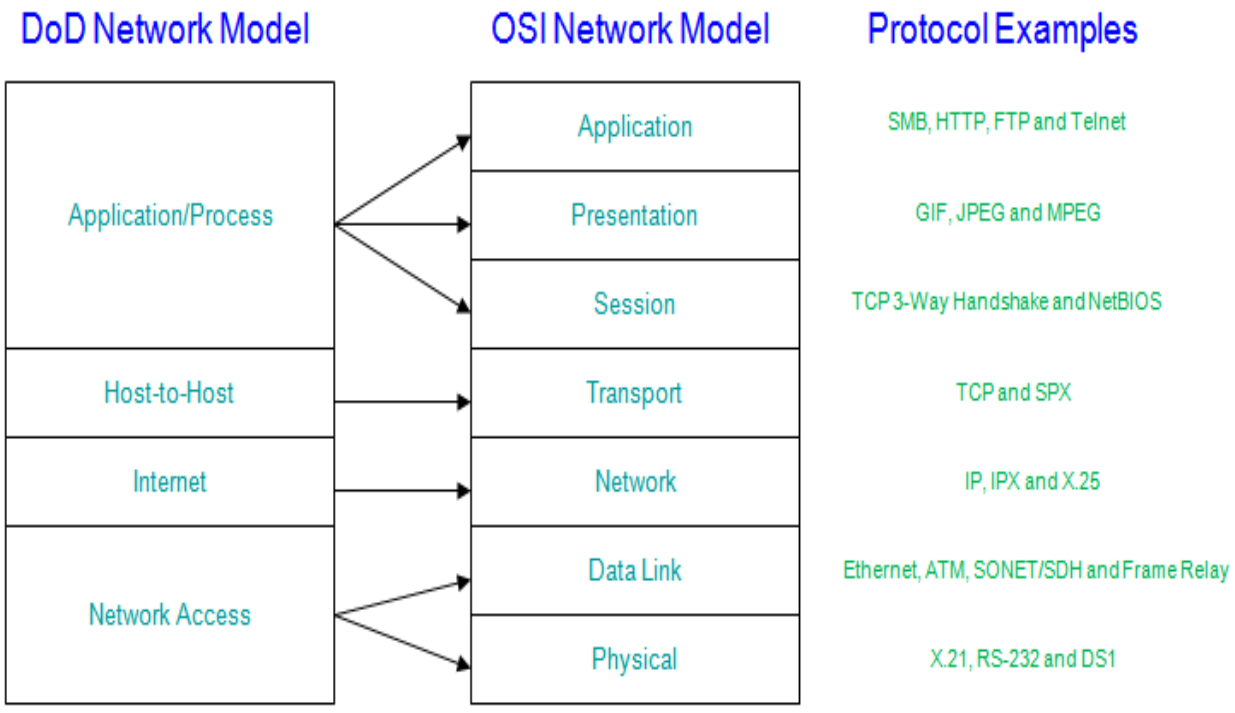


Figure 23 Comparison between the DoD model and the OSI model

The following paragraphs give a brief overview of the OSI model layers which provides information of each layer. This will enable understanding to how the separation of functionality and technology enables a network to communicate.

Layer 1: Physical layer

The *physical layer* is responsible for sending and receiving data over a physical link, thus, moving the bits over the network wires. This layer is also responsible for providing sufficient throughput to network data. Examples of physical layer communication procedures include the RS-232 handshake [58] and zero substitution functions on B8ZS T1 circuits [58].

Layer 2: Data link layer

Data to be transmitted is formatted at the *data link layer* to the next host. Packets are encapsulated into frames with the aid of protocols for addressing and error detection of data. The two sub-layers of the data link layer are the *logic link control* (LLC) and the *media access control* (MAC). The LLC provides an interface between network layer protocols and the media access service, for example, ethernet or token ring [56]. The MAC handles physical media connections, such as coaxial cable or twisted-pair.

The functions and procedures to coordinate frames between devices are handled by layer 2 functionality, which provides a method of logically grouping bits into frames with defined begin and end bits. Unlike the physical layer, the data link layer has some measure of intelligence, such as *carrier sense multiple access with collision detection* (CSMA/CD) [56] which contains detection algorithms for controlling collision detection, corrupted frames, as well as address recognition. Layer 2 supports upper layers by providing an error-free path between nodes and detecting transmission errors since corrupted data is not passed to upper layers.

Layer 3: Network layer

The *network layer* provides packet addressing and routing based on the packet's logical address from source to destination. This layer is responsible for fragmenting and reassembling data, thus, no delivery guarantee is provided by this layer. Layer 3 is used to communicate between hosts, which utilizes multiple layer 2 paths, thus providing end-to-end

communication. As mentioned before, the details of the different layers are transparent to each other, thus, multiple layer 2 technologies may be used in conjunction with layer 3 procedures. Some popular layer 3 protocols are IP, IPX and AppleTalk data delivery protocol (DDP) [43].

Layer 4: Transport layer

The *transport layer* provides a means to guarantee packet reception. This layer can also establish connections, as well as send acknowledgements as packets are received. Protocols residing on this layer establish, maintain and release connections for the hosts involved in communication. TCP ensures that data is transmitted reliably over a network that is not reliable. Control functions such as data retransmission, flow control, and error protection to application data are applied. These control functions are there to avoid collisions which result in data loss, congestion, when packets are dropped at switches and high traffic intensity when links are overloaded.

Layer 5: Session layer

The *session layer* controls temporary communication between remote hosts, which are responsible to set up sessions and connections. The functions provided by the protocols are connection establishment, connection use, and connection termination for completed sessions. Session layer protocols check for transmission errors after connection is established. Control headers are added to data packets during data exchanges.

Communication control is enhanced by providing some measure of communication control between the transport protocol and the application, i.e. between end systems. The application layer sometimes provides communication control, which means that the session layer does not need to employ this function. As a result, the session layer does not reveal itself as a protocol, but as a procedure to allow a protocol to continue its functions.

Layer 6: Presentation layer

The *presentation layer* serves as an interpreter, transmitting data in a language or syntax that the receiving host can understand. These protocols are used to translate data, then compress, and, if necessary, encrypt the data before passing it down to the session layer.

Layer 7: Application layer

The final layer, the *application layer*, manages communication between applications such as FTP and SMTP. These protocols are confused with user applications, but provide applications to operate over a network such as HTTP, which is embedded within applications such as Internet Explorer. Figure 24 illustrates the way in which network packets are constructed. The simulator defines a network packet in this way, creating a packet and adding the appropriate headers to the data.

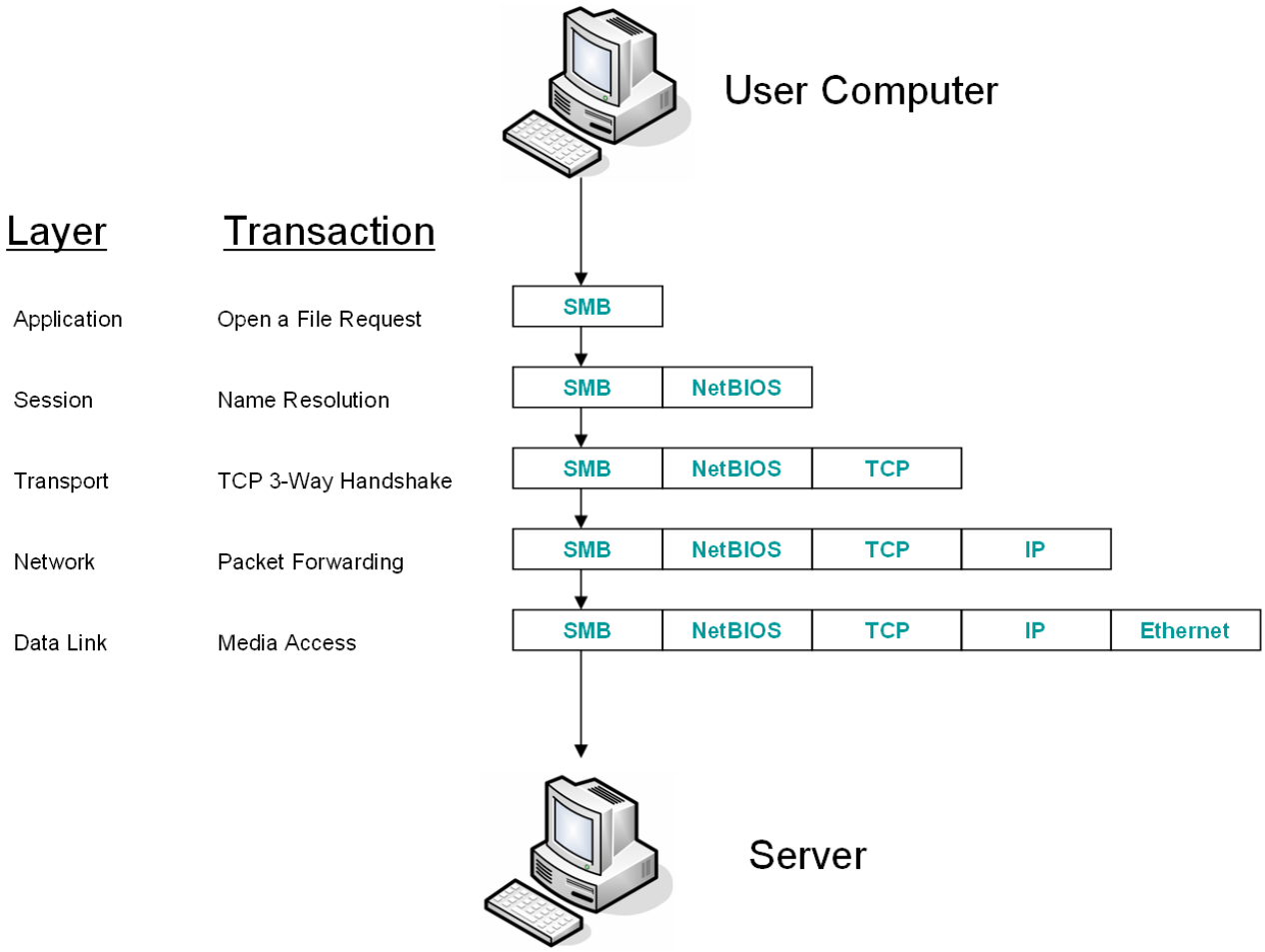


Figure 24 Packet construction

4.5. TRANSPORT TECHNOLOGIES

Transport technologies enable businesses to connect their WAN interfaces to a service provider for inter-office communication. The purpose of this section is to illustrate how WAN technologies developed to show the intelligence needed by these technologies in order to transport data between remote sites.

Layer 3 technologies use routing capability between networks and are discussed in section 4.5.1. LAN switching transport capability for multi-point communication is then discussed in section 4.5.2. Physical layer transport for point-to-point communication is then discussed in section 4.5.3.

4.5.1. Layer 3 technologies

The following section details how transport technologies evolved from a point-to-point data service to a combined voice and data service. It is illustrated that more intelligence is required to ensure sufficient service requirements for the different applications, namely voice and data.

The most widely employed transport technology offering for dedicated bandwidth with serial communication is X.25, discussed in section 4.5.1.1. For connection sensitive applications a more effective technology, namely ISDN, is discussed in 4.5.1.2.

4.5.1.1. X.25

X.25 [2] is a WAN standard communication protocol that defines how connections between user and network devices are established and maintained. This protocol was designed to operate across any type of system connected to the network. The three categories that X.25 is divided into are *data terminal equipment* (DTE), *data circuit-terminating equipment* (DCE) and *packet-switching exchange* (PSE).

A *packet assembler/disassembler* (PAD) device is used between a DTE and a DCE device to encapsulate data. The PAD device assembles outgoing data that include the X.25 header into packets. These packets are then forwarded to the DCE devices. Incoming packets to the DCE device are disassembled, which includes removing the X.25 header, and then sent to the DTE device. Figure 25 shows the packet construction of a X.25 packet.

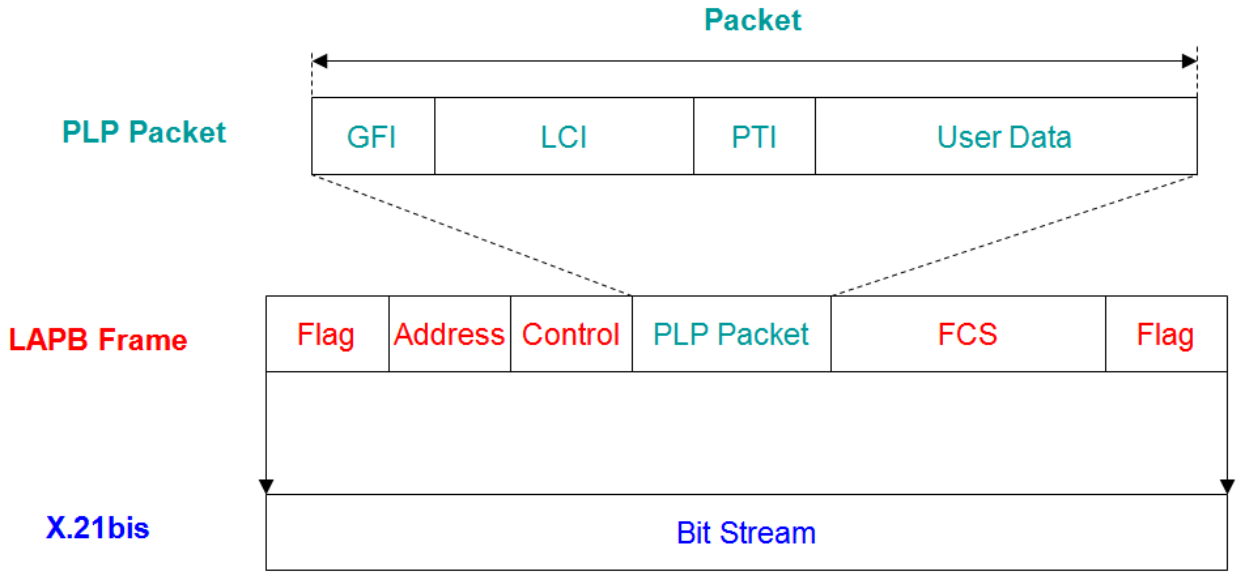


Figure 25 X.25 frame

X.25 uses three protocols, namely the *packet-layer protocol* (PLP), *link access procedure balance* (LAPB), and the *X.21bis protocol* to control data transmission. PLP manages exchanges between DTE devices across a virtual circuit. LAPB is responsible for frame packets and managing communication between DTE and DCE devices, including ordering and error checking. The *X.21bis protocol* defines electrical and mechanical procedures to activate or deactivate physical media connected to DTE and DCE devices.

4.5.1.2. ISDN

Integrated services digital network (ISDN) [2] combines analogue telephone technology with digital data technology. The aim of ISDN is to support digital data such as voice and data across existing telephone wires. The two services supported by ISDN are the *basic rate interface* (BRI) service and the *primary rate interface* (PRI) service. The ISDN BRI service offers two BRI B-channels operating at 64kbps per channel, explicitly used for user data, and one BRI D-channel operating at 16kbps for control and signalling information. BRI has a bit rate of 192kbps for providing framing control and overhead. The ISDN PRI service offers 23 B-channels, providing 1.544 Mbps, and one D-channel, providing 64 kbps, in North America and Japan. Other parts of the world achieve 2.048 Mbps with 30 B-channels and a single 64 kbps D-channel.

It is important to be familiar with ISDN devices in order to understand how framing operates. ISDN users connect with a *terminal equipment type 1* (TE1) interface, which is a specialized four-wire ISDN terminal, to an ISDN network. *Terminal equipment type 2* (TE2) interface, such as DTE, could be a standalone device or a board inside the TE2, to connect users to an ISDN network.

A *network termination type 1* (NT1) or a *network termination type 2* (NT2) device is used to connect the four-wire subscriber wiring to the conventional two-wire local loop. The NT1, which generally forms part of the carrier network, is a customer *premises equipment* (CPE) device. The NT2 device consists of layer 2 and layer 3 protocol functionality and provides concentration services, such as *private branch exchanges* (PBX). The interconnections between the various networking devices are shown in Figure 26.

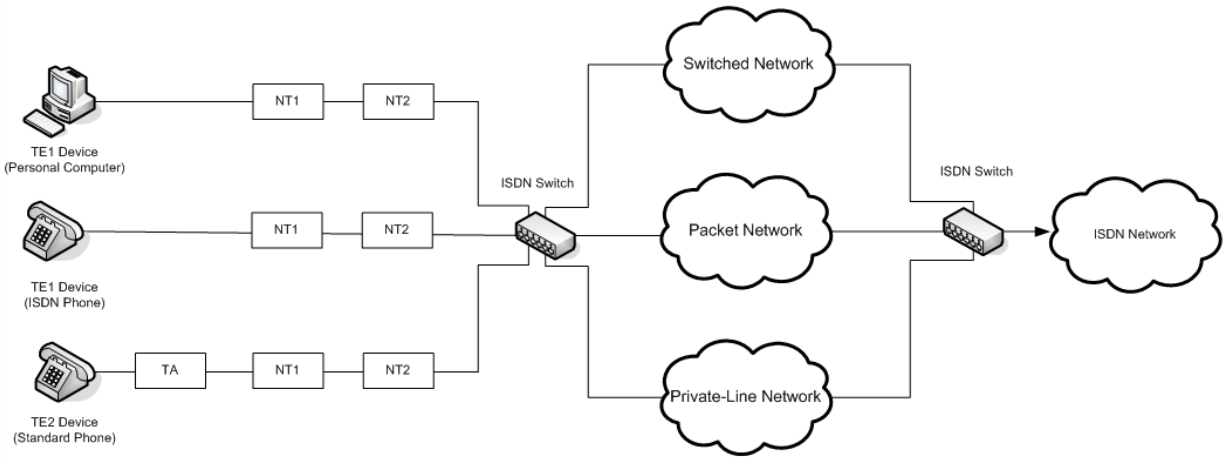


Figure 26 ISDN network construction

The next subsections discuss the three framing layers of ISDN. Layer 1 framing ensures that the transmitted bit data is received correctly by a connected receiver. Layer 2 framing discusses the format of an ISDN packet to show how signalling data is inserted into a packet. Layer 3 signalling management discusses the messages that are used to establish a connection.

Layer 1 framing

ISDN Layer 1 frames are 48 bits in length, consisting of 36 bits of user data. The frame formats differ for inbound and outbound frames, as illustrated in Figure 27. These bits are defined as follows:

- F: Synchronization information (framing bit)
- L: Average bit adjustment value (load balancing bit)
- E: Contention resolution for terminals contending for a channel on a passive bus (echo of previous D bit)
- A: Indicates active devices (activation bit)
- S: Unassigned bit
- B1, B2 and D: Contains user data

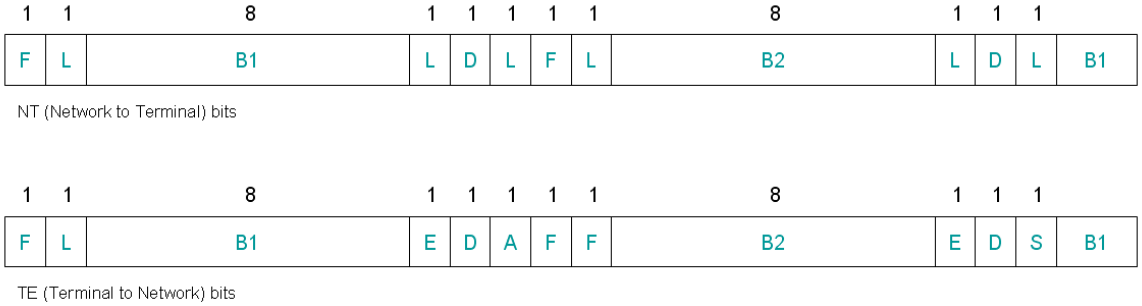


Figure 27 ISDN layer 1 framing

The layer 1 frames are defined to support multiple ISDN user devices on one circuit. ISDN supports a feature to avoid collisions by providing link contention whenever two terminals transmit simultaneously. Once the NT receives a D bit from a TE, the bit is echoed back in the next E-bit position. Therefore, to avoid collisions the next transmitted E-bit needs to be the same as the previously transmitted D bit.

Data cannot be placed into the D channel before a specific number of 1-bits, which indicates "no signal". Priority is established by the number of 1-bits that are signalled. Therefore, lower priority terminals need to wait for more continuous 1-bits than higher priority terminals. Transmission is terminated immediately once the TE detects different D- and E-bits. As a consequence, this technique ensures that only one terminal has access to the D channel at a time.

Layer 2 frame format

The ISDN Layer 2 signalling protocol is a *link access procedure*, D channel (LAPD), which is similar to the *high-level data link control* (HDLC). Control and signalling information moves across the D channel to ensure that data was received properly. Figure 28 illustrates a LAPD frame format. The LAPD flag and control fields are similar to the HDLC. The address field contains the *service access point identifier* (SAPI) used to identify the portal at which LAPD services are provided to layer 3. The C/R bit indicates a command or response to a frame. The *terminal endpoint identifier* (TEI) indicates the number of terminals that must receive the frame. However, if the TEI field are all ones, a broadcast is indicated.

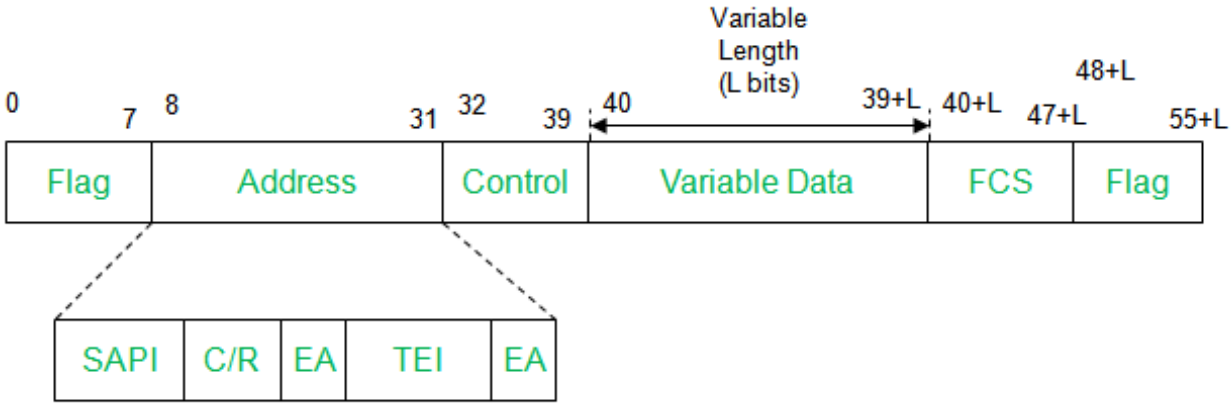


Figure 28 ISDN layer 2 framing

Layer 3 signalling management

Layer 3 provides network signalling management for data across the network. These protocols support user-to-user, circuit-switched and packet-switched connections. Signalling information includes *SETUP* to establish a connection to a remote host, *CONNECT* to connect to the remote host, *USER INFORMATION* to transmit user data to the remote user, *CANCEL* to stop a data transfer, *STATUS* to obtain the link status information, *RELEASE* to free a resource, and *DISCONNECT* to terminate a call session. These functions are similar to the functions used in the X.25 protocol.

4.5.2. Layer 2 technologies

This section discusses the use of layer 2 technologies to support point-to-point connections only over distances that directly connected interfaces cannot support. Frame relay, ATM and SONET/SDH are the technologies that are used as transport technologies. It is important to understand that the overhead of these technologies can influence QoS parameters significantly by the number of bits imposed by the specific technology, which is discussed below.

Section 4.5.2.1 discusses frame relay, a widely deployed technology for customer multi-point access. ATM is used for customer point-to-point access and is discussed in section 4.5.2.2. SONET/SDH is a core layer transport technology, discussed in section 4.5.2.3.

4.5.2.1. Frame relay

Frame relay (FR) [59] was proposed in 1984 by the *Consultative Committee on International Telephone and Telegraph* (CCITT). However, FR was originally designed for ISDN, but later formed part of high-performance WAN protocols that operate in the physical and data link layers of the OSI reference model. However, this framing technique is used over various network interfaces in contemporary IP networks. FR provides convergence between various networks, as it is a packet-switched technology, which enables dynamic network resource, bandwidth, and physical medium sharing. FR is a strict *layer 2 protocol suite* and, as a result, high performance and great transmission efficiencies are achieved, which are suitable for *WAN applications* and *LAN interconnections*.

FR consists of two categories namely, *data terminal equipment* (DTE) and *data circuit terminating equipment* (DCE). Networking equipment such as workstations, personal computers and terminals are known as DTE devices. Data carrier devices such as routers and switches that provide clocking and switching services to transport data through a network are known as DCE devices.

FR implements two virtual circuit services, namely *switched virtual circuits* (SVC) and *permanent virtual circuits* (PVC). As these services are connection-oriented data link layer communication services, a defined communication exists between each pair of devices, which is uniquely defined by a *data-link connection identifier* (DLCI). This identifier provides a bi-directional communication path per virtual circuit multiplexed into a single physical circuit between DTE devices across a network.

SVCs are *temporary connections* that provide unpredictable or random data transfers between DTE devices across a FR network. However, ISDN signalling protocols are used to establish, maintain, and terminate each session created. For each virtual circuit connection there are four SVC operating states, namely:

- *Call setup* is used to establish a virtual circuit connection between two DTE devices.
- *Data transfer* occurs when data is transferred between the DTE devices across the virtual circuit connection.
- *Idle* occurs when the virtual circuit connection is maintained without any data transfers. Consequently, the connection is terminated if no data is transferred within a certain amount of time.
- *Call termination* refers to the termination of the virtual circuit connection between the two DTE devices.

PVCs are *permanent connections* that provide dedicated connections for frequent and consistent data exchanges between DTE devices. Contrary to SVC, no call setup or termination exists for communication across a PVC. Data transfer may occur at any time since the connections are permanently established. Each PVC operates in one of two states, namely:

- Data transfer, which occurs when data is transferred between the DTE devices across the virtual circuit connection.
- Idle, which occurs when the virtual circuit connection is maintained without any data transfers. Unlike SVCs, these connections will not be terminated.

Frame format

A basic FR frame is shown in Figure 29. The beginning and ending of each frame are indicated by flags. The primary fields of a FR frame are the address field, the user-data field, and the *frame check sequence* (FCS) field. The address field, also known as the frame header field, represents the DLCI which identifies the path between the DTE device and the switch, and fields related to congestion management consisting of 6 bits.

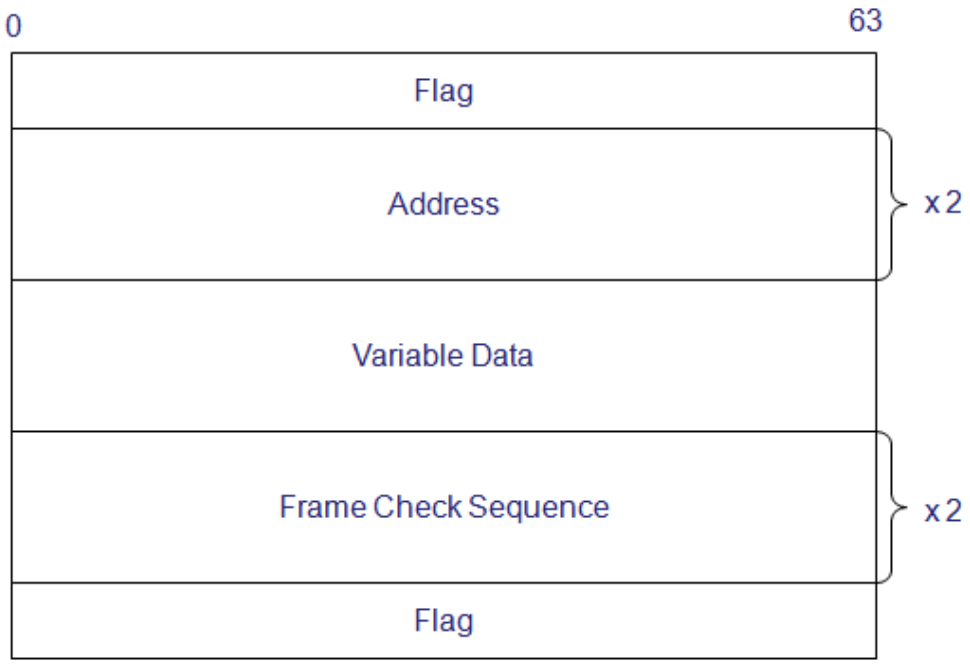


Figure 29 Frame Relay Frame

The following descriptions briefly summarize the FR frame:

- Flags: This field indicates the beginning and the ending of a frame. The default value of this field is 01111110_2 or $7E_{16}$.
- Address: Since this field represents the header, the following information is imbedded in this field:
  - DLCI: This value, which is 10 bits long, represents the value used between the DTE device and the switch. The DLCI value has local significance only. Therefore, different DLCI values can be used at the opposite ends of a connection.

- Extended address (EA): Every eight bit of each byte is used to indicate an extended address. Therefore, if the eight bit is '1', the last addressing octet is indicated otherwise the next byte is included in the address.
- C/R bit: This bit succeeds the most significant DLCI byte in the address field. Currently, this bit is not defined.
- Congestion control: This value, which is 3 bits long, control the FR congestion avoidance mechanisms. The *forward-explicit congestion notification* (FECN), set to '1' by a switch, indicates congestion to an end DTE device, such as a router, in the direction that the frame was transmitted. The *backward-explicit congestion notification* (BECN), also set to '1' by a switch, indicates congestion opposite of the direction that the frame was transmitted. Finally, the *discard eligibility* (DE), set to '1' by the DTE (a router), indicates lower precedence for a frame according to the other frames; thus, these frames will be discarded first.

- Variable data: This field contains encapsulated upper-layer data which varies according to the encapsulation technique used.
- Frame check sequence: This field contains a value for data integrity verification which is calculated at the source and destination nodes.

LMI frame format

The *local management interface* (LMI), developed by Cisco Systems, StrataCom, Northern Telecom, and Digital Equipment Corporation in 1990 [60], consists of a set of enhancements for FR specifications. The extensions manage complex networks in terms of global addressing, virtual circuit status messages and multicasting. Figure 30 shows the extended fields for FR framing.

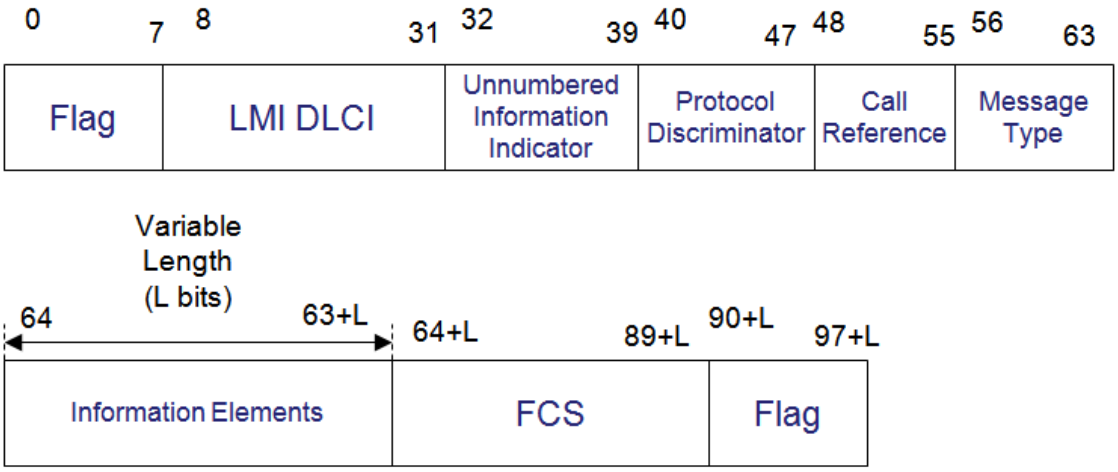


Figure 30 Frame relay LMI frame

The following descriptions briefly summarize the FR LMI frame:

- The *flag* bits indicate the beginning and ending of each frame by 01111110_2 or $7E_{16}$.
- The *LMI DLCI* bits indicate a LMI frame supposed to a basic FR frame, which has a default value of 1023_{10}.
- The *unnumbered information indicator* bits set the poll/final bit to zero.
- The *protocol discriminator* always contains a value that indicates that the frame is a LMI frame.
- The message type indicates one of the following message types:
  - Status-inquiry message: Inquire the status of the network for a user device.
  - Status message: This is the response to a Status-Inquiry Message which includes a SVC session- and PVC status information.
- The *information elements* (IE) contains the following information:
  - IE ID: Unique IE identity.
  - IE Length: Indicates the IE length.
  - Data: Contains variable byte length information, encapsulated at the upper-layers.
- The *frame check sequence* contains a value for data integrity verification which is calculated at the source and destination nodes.

4.5.2.2. ATM

Asynchronous transfer mode (ATM) [61] is a cell switching technology, integrating voice, video and data transmission with connection-oriented connections over high-speed networks. ATM cells, as illustrated in Figure 31, are 53 bytes in length, consisting of a 5 byte header and 48 bytes payload field for very efficient switching. Each ATM frame is consequently added to a DS-1/E1 or a DS-3/E3 frame for transport.

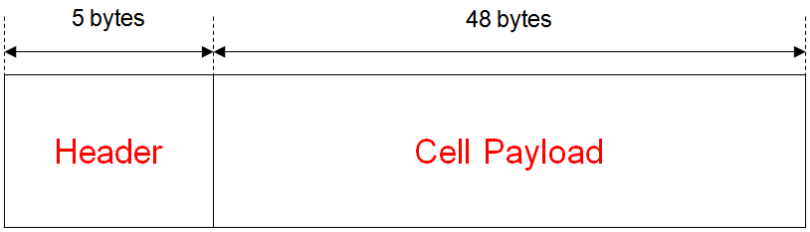


Figure 31 ATM cell format

ATM creates virtual links between end-to-end nodes, such as virtual circuits, which are identified by the VPI/VCI values. ATM users and the network control exchange information, request resources, or negotiate the use of circuit parameters via ATM signalling. As a result, the VPI/VCI pair and requested bandwidth are allocated to users.

4.5.2.3. SONET/SDH

Synchronous optical network/synchronous digital hierarchy (SONET/SDH) [57] is a significant technology that supports large-scale, high-speed time division multiplexed physical-layer transport technology and an IP-based network over fiber optic media. *Time division multiplexing* (TDM) [57] is a physical layer transport technology specially optimized for voice. Cisco supported the development of *Packets over SONET* (PoS), described in RFC 2615 [62], in the development of high-bandwidth capacity for transporting data over the Internet and large enterprise data networks. PoS is therefore the only technology that supports efficient transport of data over SONET/SDH networks.

The following headings discuss the way in which packets are physically transmitted in frames over a transport medium and how PoS utilizes this scheme.

SONET/SDH frame format (physical layer)

A SONET/SDH frame is 810 bytes which are represented as a two-dimensional byte-per-cell grid of 9 rows and 90 columns. Each SONET frame, as shown in Figure 32, consists of transport overhead and data payload. The transport overhead bytes consist of section and line overhead bytes, while the payload bytes consist of the payload capacity and path overhead bytes.

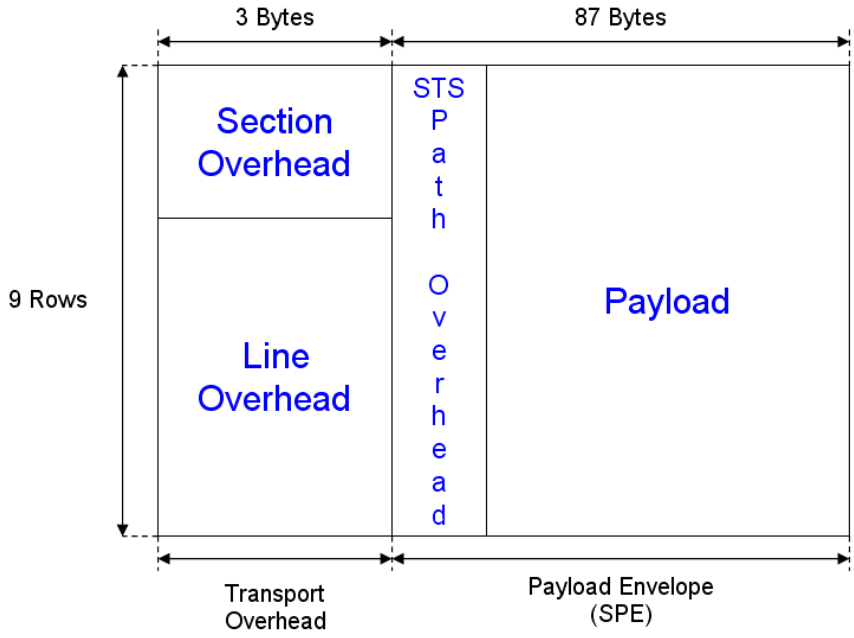


Figure 32 SONET frame

PoS framing (data link layer)

PoS framing consists of PPP framing in a HDLC format for IP packets. Therefore, packets acquire a five byte header and a four byte trailer. In order to extend PoS frame length, byte stuffing is used for frames.

Figure 33 illustrates a PPP frame which is described by RFC1661 [63]. Originally, PPP was used to transport IP packets over point-to-point links. The advantage of PPP is the ability to operate across any DTE or DCE interface, which imposes a restricted transmission rate by the particular DTE or DCE interface used. The PPP link control protocol (LCP) [63] provides a method of establishing, configuring, maintaining, and terminating the point-to-point connection.

PPP also established a standard for assigning and managing IP addresses, asynchronous and bit-oriented synchronous encapsulation, network protocol multiplexing, link configuration, link quality testing, error detection, and option negotiation for added networking capabilities.

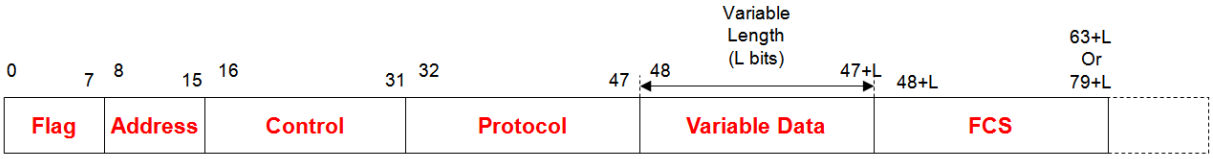


Figure 33 PPP frame

The following is a description of the PPP frame fields:

- The *flag* bits indicate the beginning and end of a frame with 01111110_2 or $7E_{16}$.
- The *address* bits where PPP does not assign individual addresses. As a consequence, all the bits are '1' which indicates a standard broadcasting address.
- The *control* bits which calls for data transmission from another user with a frame not in sequence with the other frames. The standard control value is 00000011_2 or $2E_{16}$.
- The *protocol* bits indicate which encapsulation protocol were used to encapsulate the data field information.
- The *data* bits contain the datagram for the protocol specified in the protocol field. The end of this field is indicated by the closing flag sequence, thus, allowing the FCS field to be 2 bytes long.
- The *frame check sequence* bits are used for error detection.

Figure 34 illustrates a PPP in HDLC framing which provides efficient packet delineation and error control. PoS frames are imbedded in the payload envelope as octet streams aligned to the octet boundaries.

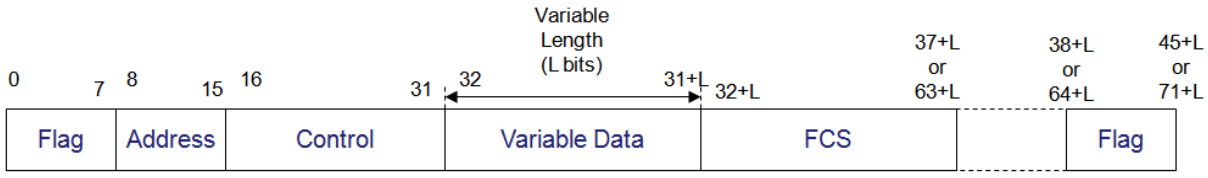


Figure 34 PPP in HDLC formatting

Physical specifications

The ATM forum [64] defined multiple encoding standards for ATM such as DS-1 and E1, to name a few. The physical specifications for SONET/SDH technology are given by Table 2.

Table 2 SONET/SDH specifications [64]

Optical Level	Electrical Level	SDH Equivalent	Line Rate (Mbps)
OC-1	STS-1	-	51.84
OC-3	STS-3	STM-1	155.52
OC-9	STS-9	STM-3	466.56
OC-12	STS-12	STM-4	622.08
OC-18	STS-18	STM-6	933.12
OC-24	STS-24	STM-8	1244.16
OC-36	STS-36	STM-13	1866.24
OC-48	STS-48	STM-16	2488.32
OC-96	STS-96	STM-32	4976.64
OC-192	STS-192	STM-64	9953.28

4.5.3. Layer 1 technologies

Table 3 gives a detailed list of framing techniques, as well as the physical media and line speeds, for the various framing techniques. These technologies are used to provide connectivity between clients, mostly implementing E1 lines, and provider lines which mostly consist of variants of SDH/SONET.

Table 3 Physical access media and line speeds for various technologies [64]

Framing	Data Rate (Mbps)	Multi-Fiber Mode	Single-fiber Mode	Coaxial Cable	UTP-3	UTP-5	STP
ATM DS-1	1.544			✓			
ATM E1	2.048			✓			
ATM DS-3	45			✓			
ATM E3	34			✓			
SONET STS-1	51				✓		
SONET STS-3c **SDH STM1**	155	✓	✓	✓		✓	
SONET STS-12c **SDH STM4**	622	✓	✓				
TAXI 4B/5B (Electrical)	100	✓					
TAXI 8B/10B (Fiber)	155	✓					✓

4.6. UDP AND TCP

Protocols are used to provide a communication standard, so that network equipment has a means to communicate with each other. The most widely used protocols are *user datagram protocol* (UDP) [65], *transmission control protocol* (TCP) [66] and the *internet protocol* (IP) [41]. IP, defined by RFC 791 [41], is a best-effort service that does not guarantee packet delivery; UDP, defined by the RFC 768 [65], is a connectionless, best-effort packet delivery system; TCP, defined by the RFC 793 [66], is a reliable, connection-oriented packet delivery system. The protocol to be used for data transmission depends on the type of application that the users use. The IP protocol is a best-effort service, which is delay- and loss intolerable. UDP is generally used for delay intolerable and loss tolerable applications, such as voice, video and multimedia. TCP is generally used for data that is delay tolerable but loss intolerable, such as FTP and HTTP downloads. Figure 35 graphically represents the percentage of protocols that are generally used across IP backbones, taken from 9 August 2000 to 9 December 2003 [67] from the Sprint service provider backbone.

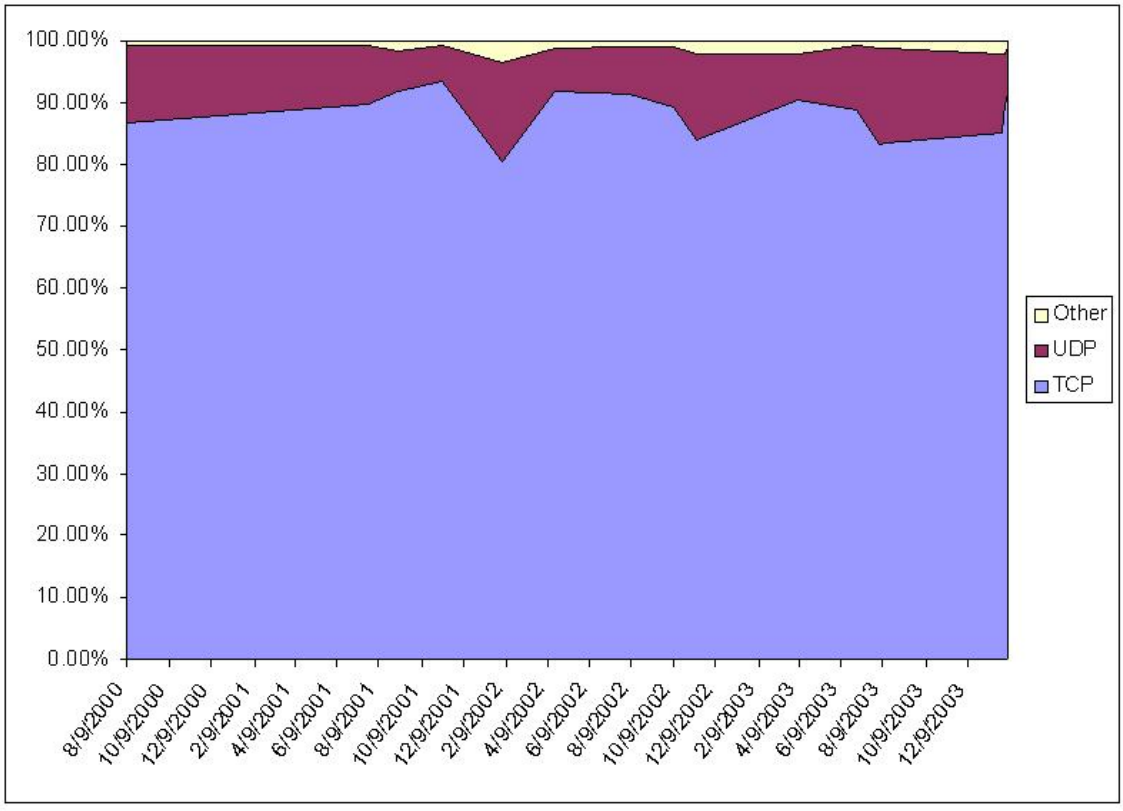


Figure 35 Protocol breakdown structure

IP consists of 32-bit size addresses, which are divided into a network portion and a host portion. The data link layer is uniquely identified by the network portion, while the host portion identifies a specific device connected to the network. The two IP header standards are shown in Figure 36 and Figure 37. Two protocol standards, TCP and IP, combine to form the TCP/IP protocol, which combines the power of the best-effort service of IP addressing and the reliable packet delivery service of TCP. The motivation behind the implementation of IPv6 is that IPv4 can only address up to 2^{32} = 4,294,967,296 hosts while IPv6 can address up to 2^{128} = $3.4*10^{38}$ hosts.

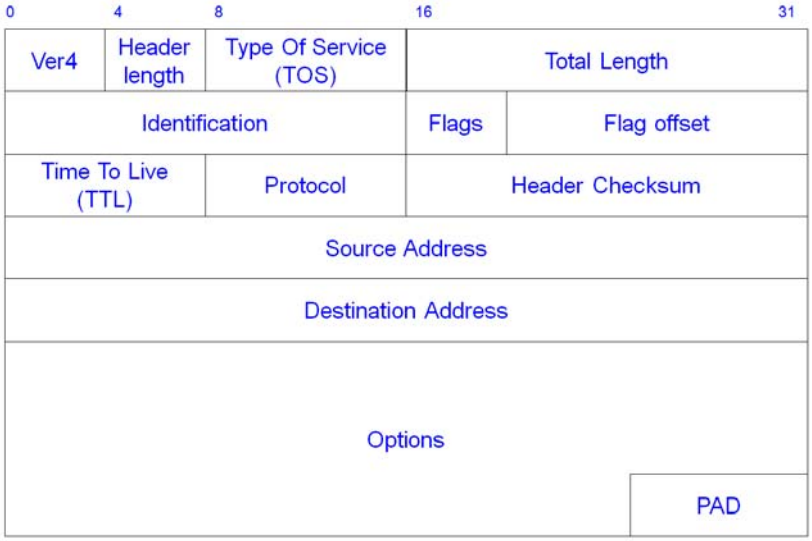


Figure 36 IPv4 addressing

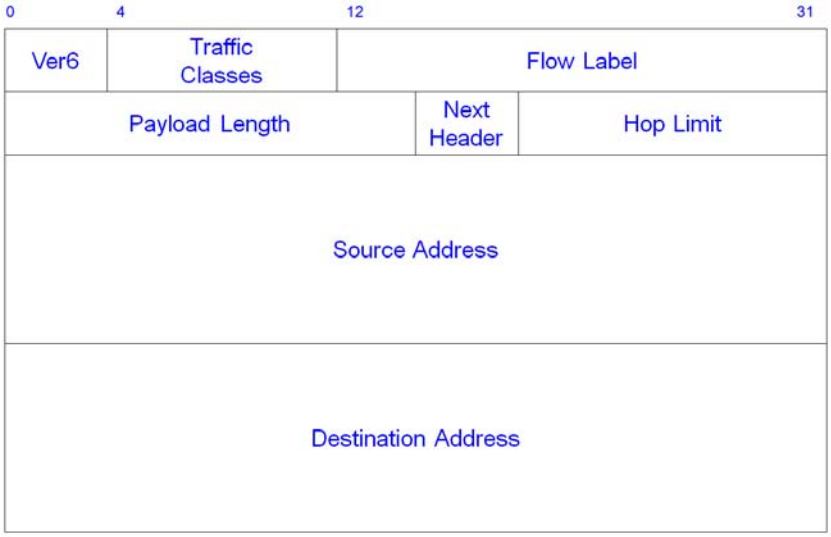


Figure 37 IPv6 addressing

TCP dynamics

The TCP protocol dynamically adjusts its transmission rates based on network congestion [9]. In order to utilize network resources optimally, a windowing scheme enables effective bandwidth utilization. As an example, given that the bandwidth for TCP is *W*, once a TCP source transmits data, the *congestion window* (*cwnd*) increases exponentially until it reaches a specific value (*ssthresh*), which is known as slow-start. For every *acknowledgement* (ACK) packet received, the congestion window is doubled. Once the congestion window value exceeds *ssthresh* the packet transmission rate increases at a slower rate. The acknowledgement packets aid to detect congestion. If duplicate acknowledgements are received, then congestion is present. The TCP protocol consequently halves the congestion window (*W*), and then recalculates the *ssthresh* and retransmits lost packets. This process is known as fast retransmit and fast recovery. Figure 38 illustrates the various TCP parameters.

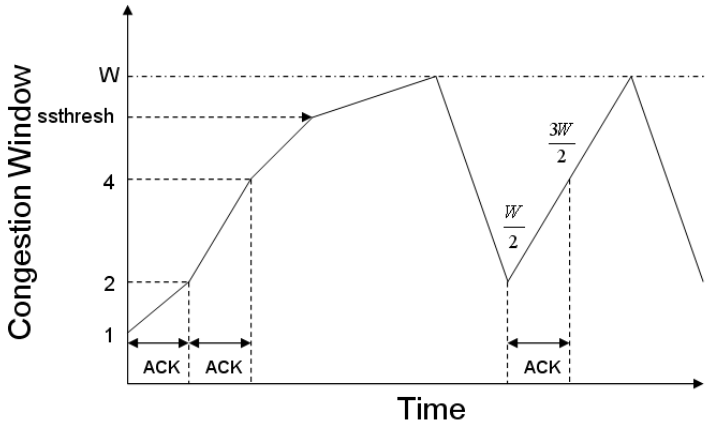


Figure 38 A TCP congestion window illustration

Packet retransmission also occurs if a mechanism, known as the timeout mechanism, is triggered. Delay also indicates congestion since queue length is proportional to link speed. If packets are not acknowledged within a certain time, packet retransmission occurs. TCP uses an exponential back-off algorithm to prevent continuous retransmission. For a segment that does not receive acknowledgement, TCP doubles the timeout interval, sending the same segment. When an acknowledgement is received the timeout interval is reset and normal transmission occurs. As a result, the RED mechanism takes advantage of TCP mechanisms to control congestion. How RED uses the TCP parameters is discussed in section 9.6.

4.7. UPPER-LAYER PROTOCOLS

The *upper-layer protocols*, such as *FTP [68]*, are used by computer applications. The files and data are simply copied into the "data fields" of layer 3 protocol frames, such as *PPP* [63]. The IP protocol is the most commonly used means of transporting data over the Internet. TCP manages the connections between the hosts as the data is forwarded between network nodes. While IP and TCP provide reliable data transfers, the *file transfer protocol* (FTP) defined by RFC 959 [68], is used to manage the actual data transfer process.

This dissertation does not concern itself with upper-layer protocols, since network traffic is generated arbitrarily. The focus is rather on the lower layers of the ISO model.

4.8. MULTI PROTOCOL LABEL SWITCHING

Service providers rely on *layer 2* and *layer 3* technologies to transmit data reliably across IP backbones as illustrated in Figure 39. These two layers are integrated through the use of *multi packet label switching* (MPLS) [43]. MPLS provides "constrained-based routing" for traffic flows, by obtaining the shortest path for resource requirements for each traffic flow admitted into the network. The aim of this traffic engineering technique is to improve data throughput on the lower OSI layers, thereby, lowering ISP expenses.

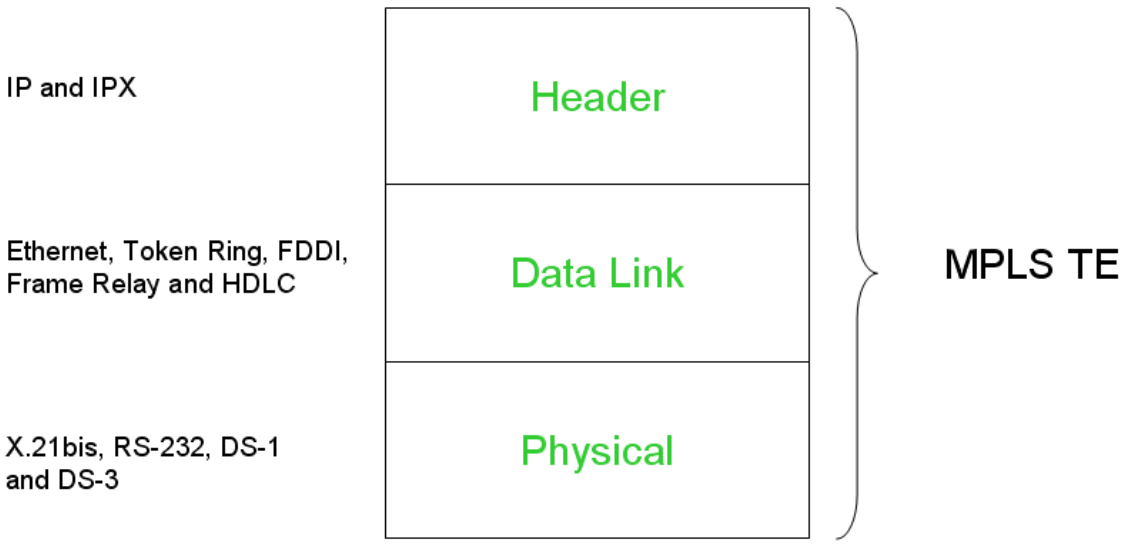


Figure 39 MPLS layer integration

Traffic engineering (TE) techniques are widely used within ATM and FR networks. MPLS TE provides a better utilization of resource capabilities by integrating traffic engineering into layer 3 technologies. Hence, routing of IP traffic is optimized since specific routing might be required by an IP backbone. Layer 2 bandwidth assurance is employed to strictly control the way in which traffic utilizes available bandwidth. *Fixed paths*, also known as *tunnels*, are established and maintained automatically across a network backbone. As a consequence, traffic flow requirements are maintained, despite link failures that lead to topology changes.

MPLS TE is based on the following mechanisms to coordinate traffic admission and control:

- A *link-state interior gateway protocol* (IGP) is used for automatic routing of traffic flows onto LSP tunnels. The most popular IGPs for exchanging router information tables between interconnected routers are RIP [69], OSPF [70] and IS-IS [71].
- *Label-switching path* (LSP) tunnels are signalled through RSVP by utilizing traffic engineering extensions. RSVP provides tunnels or pathways across router interfaces, which are configured for a specific destination and are uni-directional.
- *Label switched forwarding*, using a resource-based routing algorithm, directs traffic across multiple hops, thereby providing layer 2 capabilities.
- The most optimal paths for the LSP tunnels to utilize is calculated by the MPLS TE *path calculation module*.

4.9. DIFFERENTIATED SERVICES

The *differentiated service model* (DiffServ), defined by the RFC 2474 [34], is the primary focus of this dissertation when congestion occurs for a given queue. The power of this service model is due to network traffic flows being classified into various traffic classes, which provide class-based QoS. Packets are marked directly on the packet, within the *type of service* (ToS) field, as the packet enters the ingress router to a DiffServ enabled network. Forwarding decisions for each packet, at the routers, are made based on per-hop behaviour (PHB) policies, specified along all nodes across the DiffServ domain.

The following sections discuss the Diffserv *architecture*, *service classes*, *queuing techniques* and *congestion avoidance*. The focus of this work is on congestion avoidance optimization.

4.9.1. Architecture

Packets that enter the DiffServ domain are *classified*, *marked*, *policed*, or *shaped* at the edge routers to lift the calculation burden from the core routers. The different stages, illustrated by Figure 40, are discussed next.

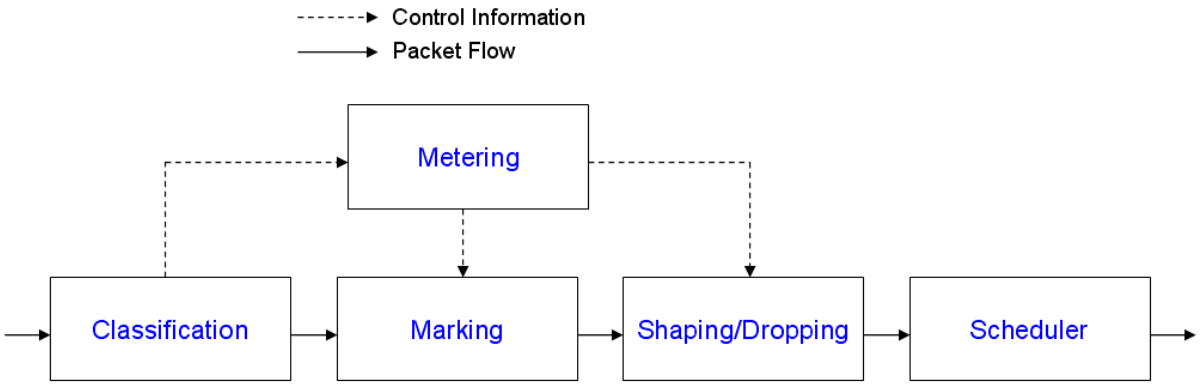


Figure 40 DiffServ architecture

Packet classification

Packets are classified with a *modular QoS CLI* (MQC) or *access control lists* (ACL), which are techniques used to separate packet classes. The packets must conform to different policies for each packet class. Classification can be based on source address, destination addresses, port numbers, protocol type, MAC addresses, as well as IP precedence. The packets are marked after the packets were classified.

Packet marking

Figure 41 illustrates how the *type of service* (ToS) field is converted to contain the *differentiated services code point* (DSCP). Packet marking occurs by changing the IPv4 headers' ToS field. However, IPv6 consists of a traffic class field, which contains the DSCP value. This field value determines how a packet is treated on a per-hob basis. Traffic shaping re-colours packets by changing the DSCP value, thus, resulting in a higher probability that the packet will be dropped at the successive node.

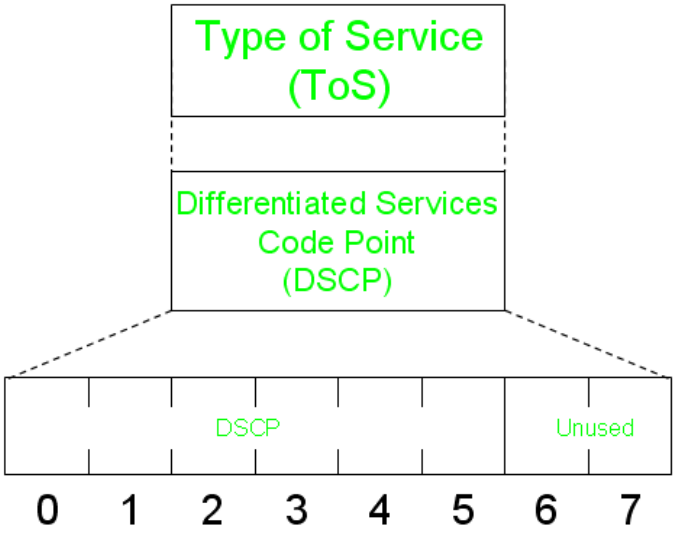


Figure 41 Mapping of the DSCP to the ToS field

Traffic limiting and shaping

Figure 42 illustrates the traffic policing metaphor [6] which allows a packet service in a queue if there are enough tokens in the bucket to the number of bits requesting service. This mechanism can be implemented as a traffic policer, such as *committed access rate policer* (CAR) [52], or a traffic shaper, such as *generic traffic shaper* (GTS) [53], or *frame relay traffic shaper* (FRTS) [55]. A token bucket does not manage traffic flows and therefore has no discarding ability of priority policies. As a result, flows that over-drive the regulator are managed within the transmission queues.

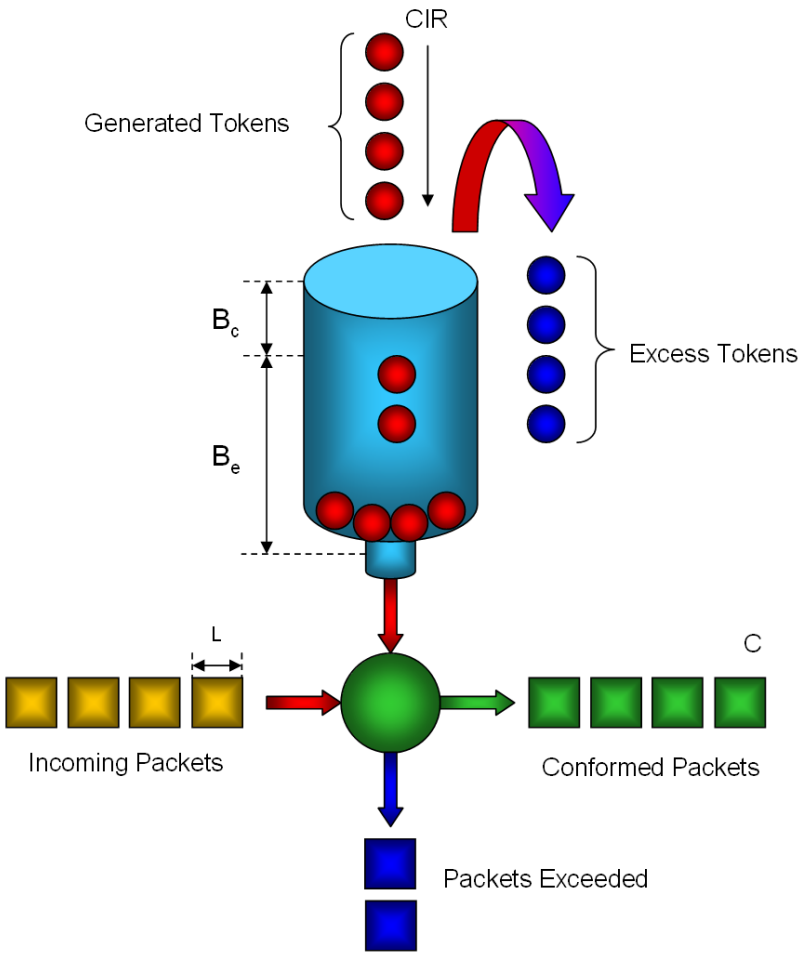


Figure 42 Token bucket model

The concept of a *token bucket* provides a method to manage a device that regulates the rate at which network traffic enters a network core. There are three parameters that determine the operation of a token bucket:

- *Committed information rate* (CIR) which specifies the amount of data that can be transmitted per time unit.
- *Committed burst size* (B_c) which specifies the number of bits, or bytes, that can be sent in each burst, within a given time unit. Bursts are specified with bits per burst for a shaper, but, bytes per burst for a policer.
- *Token accumulation time* (T_c) which specifies the time at which tokens fill the bucket.

The token bucket operates in the following manner:

- Tokens fill the bucket by a CIR, measured in tokens/sec.
- Excess tokens are discarded if the bucket is full.
- In order to transmit a packet, the bucket is checked for available tokens corresponding to packet size *L*.
- If enough tokens are available, the tokens are used to transmit the packet and colour it green.
- In the case of a shaper, packets wait until enough tokens are available before being serviced. However, in the case of a policer the excess packets are discarded.

The token bucket generates B_c tokens per time unit which is specified by T_c, calculated using the B_c and CIR values. By definition, an arbitrary fast interface bit rate will not exceed CIR. Each packet must obtain a sufficient number of tokens for permission to enter the network. In order to transmit a packet, the number of tokens corresponding to the packet size must exist within the bucket and then removed. Hence, a general equation for the token bucket metaphor is given as

$$T_c = \frac{B_c}{CIR} \text{ sec} \tag{4.1}$$

In the case of a shaper, if there are tokens present in the bucket, but an insufficient number of tokens exists for a packet, the packet is marked down and delayed until enough tokens are accumulated within the bucket. In the case of a policer, packets do not experience delay, thus, if there are not enough tokens in the bucket, the packet is discarded. The committed- and extended burst sizes are calculated by using

$$\textit{Committed burst } (B_c) = \frac{1.5 * CIR}{8} \text{ bits} \tag{4.2}$$

$$\textit{Excess burst } (B_e) = 2 * B_c \text{ bits} \tag{4.3}$$

For a policer, bursts are allowed to reduce packet dropping by borrowing excess burst tokens, namely B_e. Accordingly, the following rules determine the number of bursts that may pass through the policer at a given time:

- Determine the *compound dept* (C_D) value as $C_D = \sum_{j=1}^{i} \sum_{k=2}^{L} packet_i$, where i is the number of packets and L the length for a packet.
- Compare B_e with C_D.
- If $B_e > C_D$, all packets will be dropped until enough tokens have been accumulated to compensate for the C_D value. The discarded packets do not remove any tokens from the bucket.
- If $B_e < C_D$, a packet is discarded and C_D is set to zero. Consequently, a new C_D value will be computed for the next packet that needs to borrow tokens.

Per hop behaviour policies

Once packets enter the core network, the PHB policies are enforced. PHB depends on the packet marking provided by the DSCP. Core routers simply store and forward packets, but, provisions bandwidth guarantees according to the traffic type. *Expedited forwarding* (EF) traffic receives *priority queuing* (PQ), while *assured forwarding* (AFxy) traffic receives *class based weighted fair queuing* (CBWFQ) and *weighted random early detection* (WRED), class based policing, or *class based traffic shaping* (CBTS) at the edge nodes. The combination of PQ and CBWFQ is known as *low latency queuing* (LLQ). Detailed discussions on the EF- and the AF services are provided within the following sections.

4.9.2. DiffServ domain

The DiffServ domain is defined at the region where DiffServ functionality is provided by network routers. Functionality is provided by the usage of traffic classification, marking, and shaping or policing at the ingress routers. Traffic policing and shaping at the edge routers lifts the computational burden from the core routers. The core routers' only function is to store and forward packets within the DiffServ Domain. Figure 43 illustrates the logical separation of the various DiffServ nodes where the routing nodes enforce DiffServ policies.

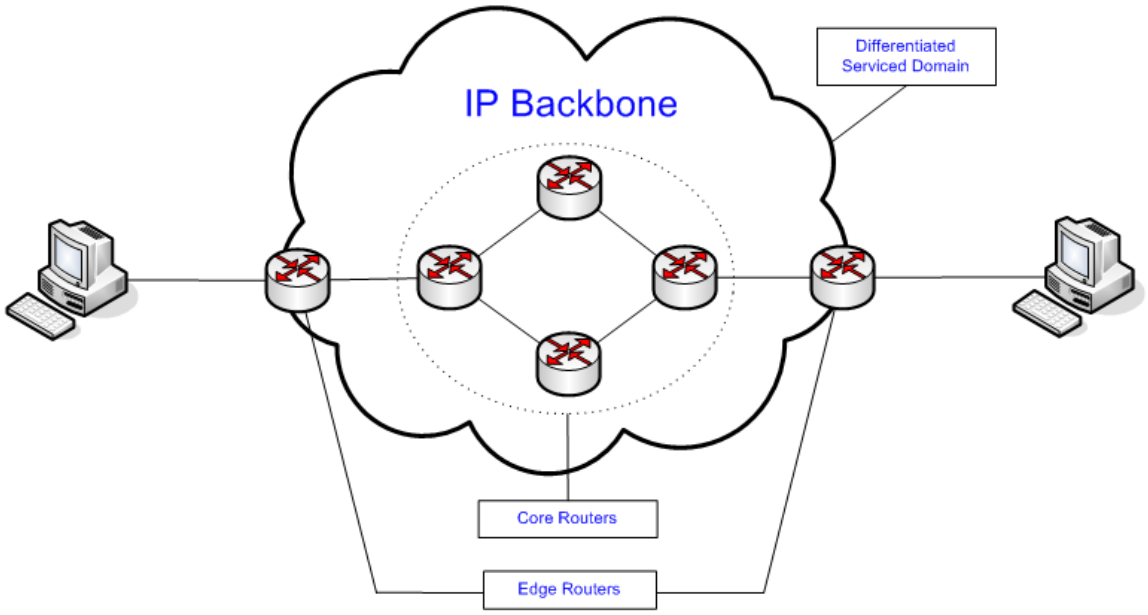


Figure 43 DiffServ domain

4.9.3. EF service

The *expedited forward service* [72], also known as the *premium service*, provides customers with stringent bounds on packet loss, delay and jitter, as well as assured bandwidth in VPNs. The decision of queue size affects packet delay and jitter at each network node. Thus, the queue size increases when the departure rate is less than the arrival rate in the same node. This service provides a minimum departure rate for aggregate traffic where the departure rate is independent from the intensity of the other traffic at the node. However, EF PHB provides no bounds on delay and jitter, but, it is taken to be sufficiently low.

Traffic that is classified to the EF service is *policed* at the ingress routers to control conformance of traffic flow rates according to a SLA agreement. Thus, all traffic that belongs to this service is conditioned, which sets the minimum departure rate at each intermediate node in the DiffServ domain. EF packets are marked with $\mathbf{101110}_2$ within the DSCP. Since delay and jitter are important measures, the queue lengths should be kept small. To ensure that these measures are met, a SLA is applied to the traffic class. Thus, admission control needs to be applied at the network nodes.

EF queues have priority over the other queues, such as the *assured forward* (AF) and *best-effort* (BE) queues. Only a small amount of link capacity is used for this service to avoid low QoS for the other less demanding services. However, bottlenecks may occur in some parts within the DiffServ domain if network traffic is unevenly distributed. Therefore, there is no guarantee that the other services will experience starvation during some time periods. However, *schedulers* provide statistical multiplexing to minimize queue starvation, such as *weighted fair queuing* (WFQ).

4.9.4. AF service

The *assured forward service* [73] provides customers with reliable communication even with network congestion. This service is used by clients that can experience some measure of QoS degradation, for example packet loss or delay. Each DiffServ node allocates a certain number of resources, such as bandwidth and buffer size, to each of the four IP packet classes, numbered from 1 to 4.

This service implements mechanisms to detect and respond to long periods of congestion in an attempt to minimize class congestion. Short traffic bursts may be buffered, since short term congestion does not drastically affect performance, but long periods of congestion may be controlled by discarding packets. However, short term congestion should be handled independently from long term congestion. Thus, an attempt is provided to equal packet drop probability for flows with similar data rates, but with different burstiness over longer time periods. RED is the active queue management system of choice used by service providers to control packet discarding.

Three different priorities, namely low, medium and high drop precedence, are assigned to each of the four AF classes, which are marked in the DSCP field, as shown in Table 4. Each traffic class is provided with its own queue for each node within the DiffServ domain. Packets that do not conform to any policies, or with unrecognized code points, are forwarded with the default DSCP of 0, which is the best-effort service.

Table 4 DSCP corresponding to assured forward traffic

Drop Precedence	Class 1	Class 2	Class 3	Class 4
Low	AF11 001010	AF21 010010	AF31 011010	AF41 100010
Medium	AF12 001100	AF22 010100	AF32 011100	AF42 100100
High	AF13 001110	AF23 010110	AF33 011110	AF43 100110

4.9.5. Queue configuration

The two class based queuing mechanisms that are generally used are PQ and CBWFQ, which forms the LLQ [36]. This section describes the most important configuration aspects for the above mentioned mechanisms. Bandwidth guarantees are only applicable with the occurrence of congestion between router links. Weights for bandwidth allocation and congestion control are therefore specified for traffic over-provisioning purposes to queues. For each queue congestion control measures are different, thus each queue receives different preferences to the link bandwidth. The remainder of this section detail CBWFQ and LLQ as queuing mechanisms for bandwidth allocation and congestion management according to weight.

Class based weighted fair queuing (CBWFQ)

Weighting: A service provider specify weights that are assigned for each class, after packets were classified based on the match criteria filters (ACL or MQC) for each packet that arrives at the output interface. Packet weights are derived from the amount of bandwidth assigned to the specific service classes. The assigned weights ensure that the different service classes are serviced fairly.

Bandwidth allocation: Only 75% of the bandwidth specified per interface can be utilized. The remaining 25% is used for overhead, such as layer 2 information, checksum information and best effort traffic. This rule may be overridden, but great precaution is recommended in ensuring that enough bandwidth remains to support traffic overhead.

Congestion: In the event of congestion, packets are discarded with the aid of active queuing management algorithms. These algorithms are configured to discard packets based on the service class packets reside on.

Low latency queuing (LLQ)

Weighting: Weights are assigned with the use of match criteria filters (ACL or MQC), similarly to CBWFQ. However, the PQ queue receives strict priority over the other service queues. The PQ needs to be limited by the use of a policer; otherwise, this traffic will dominate the lower priority queues.

Bandwidth allocation: Only the maximum allowed bandwidth must be specified for packets that belong in this class. Bandwidth is guaranteed to this priority class which also restrains the flow of packets to the allocated bandwidth. As with CBWFQ, only 75% of allocated bandwidth is allowed per interface and 25% for overhead information. The allocated bandwidth always includes the Layer 2 encapsulation header. Internally, a token bucket measures the offered load to conform traffic streams to the assured bandwidth configuration. Only traffic that conforms to the token bucket is guaranteed low latency.

Congestion: In the event of congestion, which occurs when the specified bandwidth is exceeded, the policing mechanism discards excess packets. As a consequence, priority queue traffic is metered to ensure that the configured bandwidth allocation for the class is not exceeded. Metering is only performed under congestion conditions on a per packet basis. Hence, traffic is allowed to exceed configured bandwidth when congestion is not present. Priority traffic is restrained within its allocated bandwidth to ensure that non-priority traffic has a portion of the link bandwidth available.

Policy configuration

For the real time class, each voice flow is guaranteed a 16kbps pipe session [33]. This queue's committed burst size is the product of the maximum number of concurrent sessions allowed and the average UDP datagram size, which is typically 80 bytes. Therefore, for 8 concurrent voice sessions the committed burst size needs to be set to CIR_{EF} = 8 * 80 = 640 bytes. As a consequence, this setting sets a worst case bound for all customers it supports.

The other queue classes support 8kbps sessions. Therefore, the committed burst size is the product of the *maximum transmission unit* (MTU) value, which is the maximum IP packet size across a medium, and the number of concurrent data sessions that should be supported. The best effort class does not require this setting since data remains unmanaged for this class. If a customer is provided a package to support 16 IB, 8 BB and 16 BE sessions, ignoring Layer 3 queuing overhead, the committed bursts are set to CIR_{IB} = 16*1500 = 24 kbytes and CIR_{BB} = 8*1500 = 12 kbytes.

4.9.6. Congestion avoidance

The *random early detection* (RED) congestion avoidance mechanism [7] is the most commonly used algorithm to avoid congestion across network links. Variants of RED are classified in four different categories, namely:

- the single average, single threshold, for example RED;
- the single average, multiple thresholds for, example *weighted RED* (WRED);
- the multiple average, single threshold for, example Blue; and
- the multiple average, multiple threshold for, example *RED with IN and OUT* (RIO).

The following section explains the various RED implementation philosophies.

RED based congestion avoidance algorithm

The RED [7] algorithm aims to minimize queue length by discarding packets randomly. This scheme has two congestion thresholds, namely Th_{MIN} and Th_{MAX}. If the queue length is below the Th_{MIN} parameter, none of the packets inside the queue are discarded. Once the queue length increases above Th_{MIN}, packets are dropped linearly with probability p, which increases linearly with queue size, until the threshold Th_{MAX} is reached. If the queue length reaches Th_{MAX}, all received packets are discarded, thus, the queue becomes a drop-tail queue. Figure 44 provides a graphical illustration of the queue management algorithm.

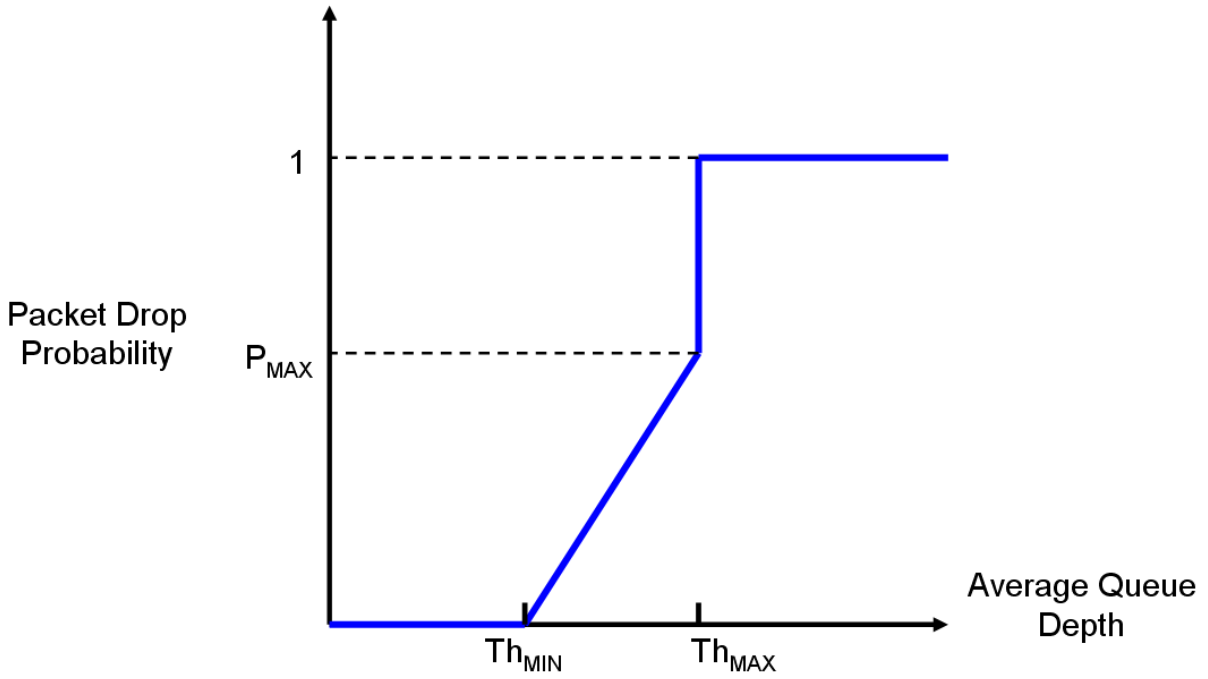


Figure 44 RED packet drop probability parameters

This queue management scheme uses TCP mechanisms to control traffic flows. Thus, all TCP flow control mechanisms are triggered for different end hosts at various times. In this manner queue overflow is avoided, thus, preventing drop-tail behaviour. However, packets are discarded even at times when a queue is not yet full. A disadvantage of drop-tail behaviour is that all connected hosts simultaneously halt data transmission, and then simultaneously start transmitting at higher data rates. Therefore, very high node utilization is followed by very low node utilization, thus causing low network performance. Variants of RED, such as WRED, RIO-C and RIO-D, are discussed next.

Weighted random early detection

Generally WRED is useful to be implemented in the edge routers' output interface where congestion is more likely to occur [7]. An IP precedence value is assigned per packet at the edge routers as these packets enter the network of core routers. Because packets are assigned precedence, WRED determines how individual traffic flows should be treated. A limitation of the WRED algorithm is that the average queue length calculation depends only on the full queue, not taking the various class queues into consideration. This limitation is addressed with RIO-C and RIO-D, which is discussed next.

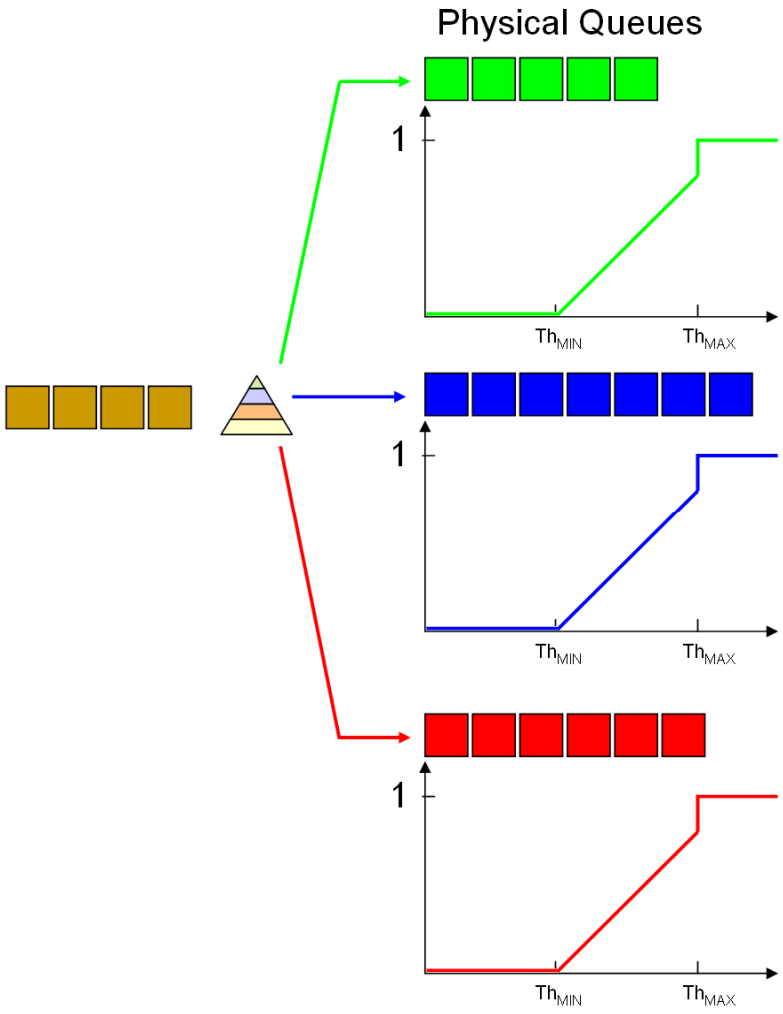


Figure 45 Packet drop probability based on WRED

The average queue length of a priority queue, called a class queue, is determined for the RED algorithm whereby a set Cisco software implementation [7] is as follows,

$$Q_n = ([1 - 2^{-w_q}] * Q_{n-1}) + (2^{-w_q} * Q_{current}) \tag{4.4}$$

where

$$w_q = \frac{-\ln(MTU * \frac{8}{BW})}{\ln 2}, \tag{4.5}$$

and $Q_{current}$ is the current length of the service queue, MTU is the maximum length of a packet in the queue, and BW is the bandwidth available for the service queue.

The w_q value is determined by knowing what *maximum transmission unit* (MTU) and *bandwidth* (BW) are provided by the router. These values are known by router bandwidth and by the rate at which the router transmits data. However, the recommended value for the w_q variable is set to 0.002 [7].

Values of w_q impact the utilization of all resources by having a large w_q result in slowing down the connection, making the averaging process less responsive. For a large w_q value, bursty traffic is forwarded without dropping, making the large value appropriate for business critical data traffic.

Small w_q values are more responsive and provide instantaneous averages but makes bursty traffic less bursty. Small values are suitable for high volume, few bursts, that responds well to short term loss which is appropriate for *bulk business* (BB) and *best effort* (BE) data classes.

The RED algorithm implemented on gateways is given in Figure 46. This algorithm is accordingly implemented within the simulation environment used for the purpose of this thesis. For analysis purposes, it is possible to determine if a packet was discarded as a result of buffer overflow or by the RED congestion avoidance algorithm by employing the ECN bit.

```
ave ← 0
count ← 1
FOR each packet arrival
  Calculate the new ave value (Average Queue Size)
    IF the queue is non-empty
      ave ← (1-w_q) * ave + w_q * q
    ELSE
      m ← f(time –time_idle)
      ave ← (1-w_q)^m * ave
    END
  IF th_min ≤ ave < th_max
    count ← count + 1
    p_b ← p_max *( ave – th_min) / (th_max – th_min)
    p_a ← p_b / (1 - count * p_b)
    Mark arriving packet (ECN)
    count ← 0
  END
  ELSE IF th_max ≤ ave
      Mark arriving packet (ECN)
      count ← 0
  END
  ELSE
      count ← -1
  END
END

IF q == 0
   time_idle ← time
END
```

Legend

Saved Variables

ave : Average Queue Size
time$_{idle}$: Start of queue idle time
count : Number of packets since last marked packet

Fixed Parameters

W_q : Queue Weight
th_{min} : Minimum Queue Threshold
th_{max} : Maximum Queue Threshold
p_{max} : Maximum Drop Probability

Other Parameters:

p_a : Current Packet-Marking Probability
q : Current Queue Size
time : Current Time
f(t) : Linear Function of time t

Figure 46 The RED Gateway Algorithm

RIO-C and RIO-D

RIO-C and *RIO-D* [53] address the limitations of the RED mechanism. These two congestion avoidance algorithms provide congestion avoidance to excessive packets, as illustrated in Figure 47. Hence, packets that violate traffic rate constraints are placed in a virtual queue. Packets placed on the virtual queue are subject to be dropped more stringently. However, the two algorithms calculate the average queue depth based on the primary and virtual queue size.

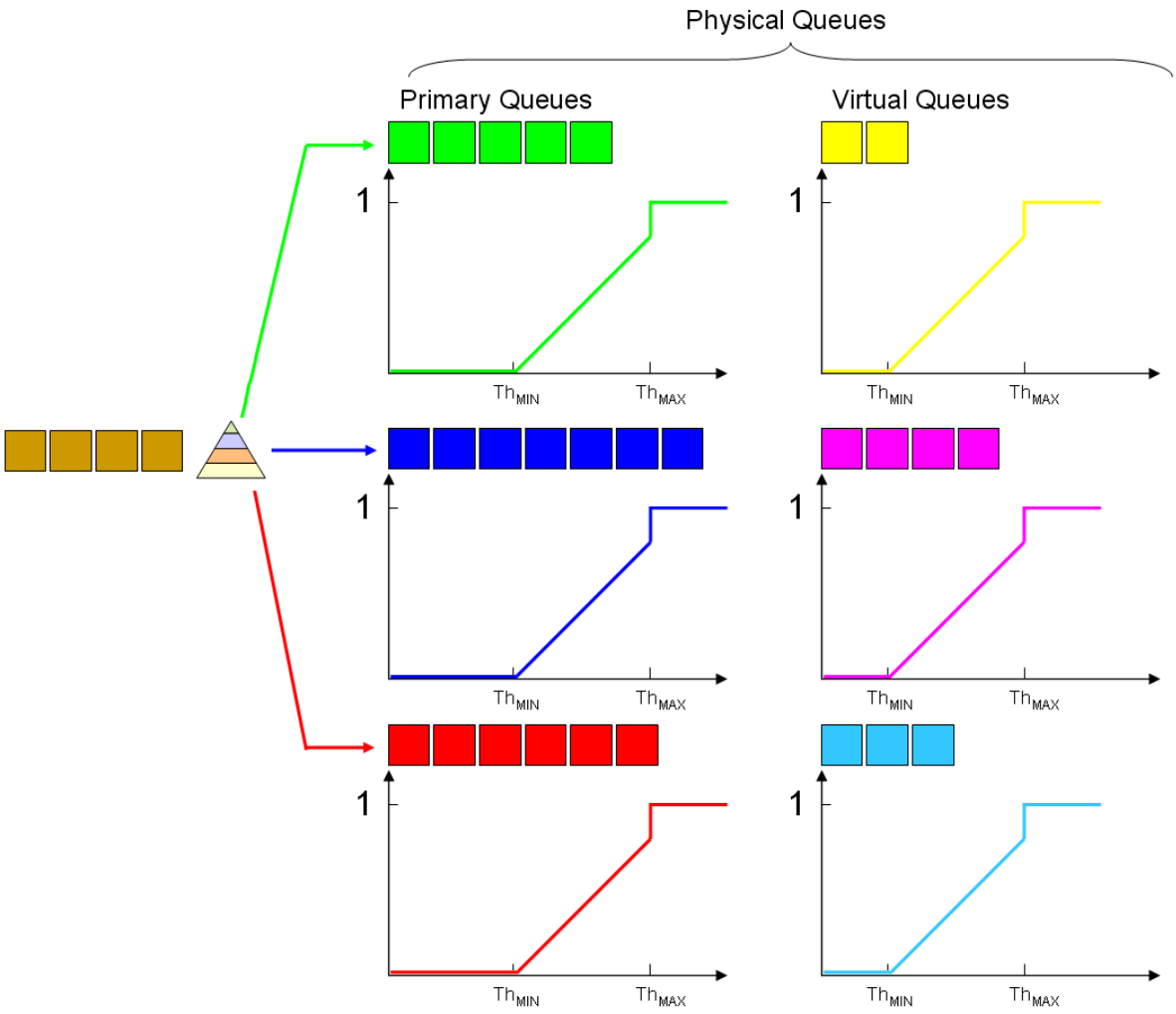


Figure 47 Packet drop probability based on RIO-C and RIO-D

4.10. CONCLUSION

This chapter outlined the different aspects of an IP network in an effort to illustrate the positioning of the research conducted within this dissertation.

Layer 2 protocols provide a means to carry data across network connections. Thus, connected devices need to understand in which format they will receive data packets, including packet overheads. Layer 3 protocols, or routing protocols, provide a means to direct traffic across various nodes. When combining the logic of layer 2 and layer 3 protocols it becomes imperative to ensure that additional overhead of physical links are considered not to degrade network QoS. MPLS is used to combine layer 3 and layer 2 protocol philosophies to effectively switch traffic between service providers' connected links.

The OSI model insures that vendors can develop technologies to be interoperable. Transport technologies are the underlying capability of a network to communicate over remotely located sites. Layer 1 technologies enable the long reach capability of the transport network.

UDP and TCP are two methods for data packet delivery. Both protocols are used by upper-layer protocols described by the ISO model. For this dissertation a protocol breakdown is needed to ensure that a correctly estimated traffic flow amount can be generated by the simulator. The various flows are treated by a differential services philosophy such that QoS can be implemented according to agreed measures contained within SLA measures. This dissertation is concerned with the effective treatment of TCP flows when carried across service provider links. RED is used as a measure to control network flows such that sources can utilize available bandwidth equally. The focus of this dissertation is to analyse the influence of the RED algorithm parameters that can be configured within a network, and to indicate the most optimal values to use for maximal node utilization.

CHAPTER 5 SIMULATION ENVIRONMENT CONFIGURATION

This chapter outlines how the various elements within the simulated network environment are provisioned. A simulation environment which is capable to provide a basis to conduct congestion control features is identified, which closely resembles real network protocol capabilities as discussed in section 5.2. The network simulator is used to model the various class queues of traffic, discussed in section 5.3. Network traffic packets are classified by the use of DSCP values, where the DSCP values classify packets and place the packets into the correct queues, discussed in section 5.4. Section 5.4.1 details the DSCP values, section 5.4.2 detail the link bandwidth provisioning factors, section 5.4.3 details the scheduler weight priorities, and section 5.4.4 details the RED parameters.

5.1. INTRODUCTION

It is important to be aware of all networking conditions and states when conducting research on network QoS implementations. Therefore, the most important QoS measures are given throughout this chapter in an attempt to focus on a specific QoS mechanism, namely RED.

The simulation software used, namely *network simulator 2* (NS2), forms the basis for the simulation environment [74]. A brief background is therefore provided which motivates the use of the simulation software, in order to create real networking implementations. The service models, as well as the hardware configurations, are then described. Specific QoS parameters are consequently outlined for the various mechanisms.

5.2. SIMULATION ENVIRONMENT

A network simulator software, called NS2 [74], developed by the University of California as part of the *Virtual InterNetwork Testbed* (VINT) funded by DARPA, was used to conduct experiments. The two main collaborators to this project are the XEROX *Palo Alto Research Center* (PARC) and *Lawrence Berkley National Laboratory* (LBNL). The *network simulator 2* (NS 2) was developed in an attempt to simulate complex networking conditions. Real network implementations within this simulator are used to generate real network traffic conditions, such as DiffServ, RED, traffic conditioning and scheduling in order to generate accurate real-world results.

The NS2 project, developed to offer substantial support for simulation purposes, is the most extensively used networking tool within the network research community. This simulator implements real networking protocols, such as TCP, routing multicast protocols over wire and wireless networks, as well as call admission control, to great extent for networking environments. The power of NS2 lies in the fact that the implementation is event-driven and executes in a non real-time fashion. This is made possible through the implementation of hardware with C++ code and using Tcl and Object Tcl shells as the interface, allowing the input Tcl file to describe the model that must be simulated.

Researchers can construct complex networks by defining arbitrary network topologies, composed of nodes, routers, links, and shared media. The simulator uses agents, which are defined protocols, to attach protocol objects to network nodes. The *network animator* (NAM) assists to graphically visualize the network environment providing more insight into the simulations by visualizing packet trace data.

The traces generated by NS2 yield data that correlates accurately with practical results. Due to the complex implementation of NS2, measurements that are conducted include protocol and networking layer overheads.

5.3. NETWORK SETUP MODEL

NS2 was chosen to simulate DiffServ functionality, as illustrated in Figure 48. The primary purpose of DiffServ is to obtain performance measures for each queue individually based on QoS requirements between varieties of customers. Packet processing occurs mainly at the edge routers, relieving the core routers of the processor from utilization burden. As a result, the main concern at the core routers is bandwidth provisioning to the different services. Core routing QoS does not form part of this dissertation. The focus of this work is between the customer and the service provider edge node connections.

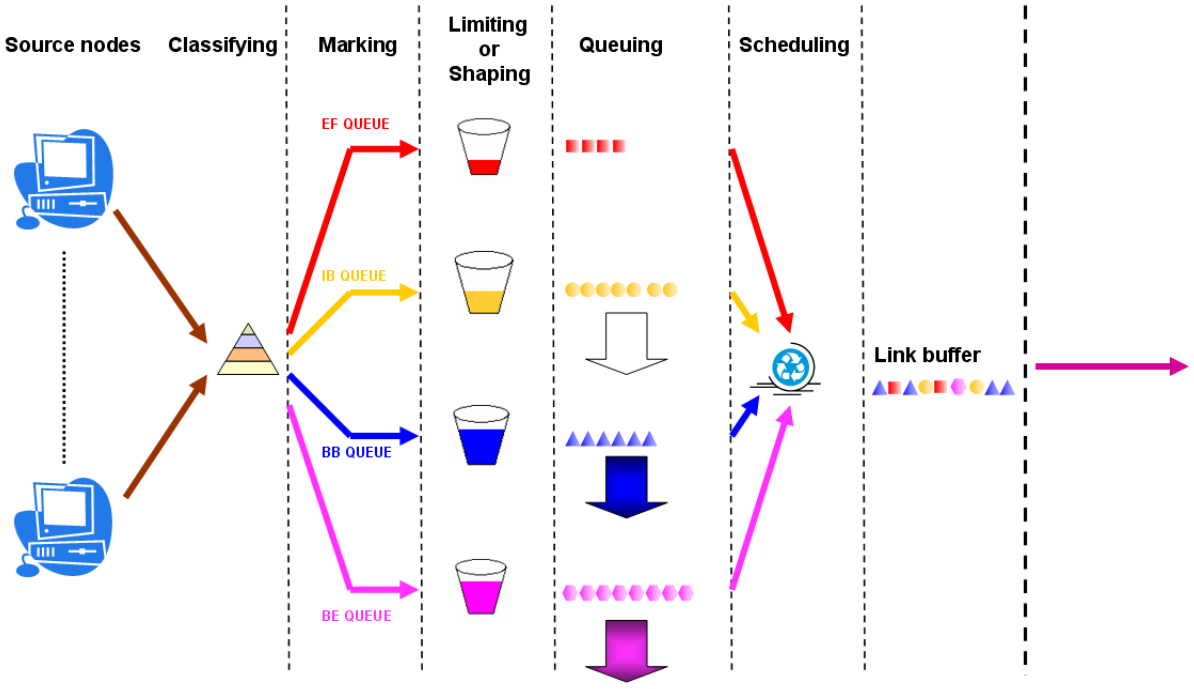


Figure 48 DiffServ implementation model

The *expedite forward* (EF) queue supports unreliable, delay sensitive traffic. As a result, *user datagram packets* (UDP) are used to transport data across this queue. An implementation such as *Voice over IP* (VoIP) carries voice and delay sensitive traffic where voice quality may suffer from some packet loss due to traffic policing. Mechanisms such as the *real-time transport protocol* (RTP) and the *compressed RTP* (cRTP) protocol buffers incoming data streams such that a constant data flow occurs at the end users' terminal. As this queue does not transport controllable traffic, RED is not applied to this queue. Packet loss as a result of the EF queue is negligible, since these packets have strict priority over the other queues. The important QoS measure for the EF queue is delay since this traffic is delay sensitive.

The *interactive business* (IB), *bulk business* (BB), and *best effort* (BE) queues transport controlled network traffic end to end across a network. As a consequence, the *transmission control protocol* (TCP) is used to transport data across a network. All data that reaches its destination node is acknowledged, thus enabling this protocol to support full-duplex functionality. However, UDP traffic is lost in the case of policing and congestion control within these three queues since UDP traffic is not acknowledged by the destination nodes.

Figure 49 illustrates the hardware configuration. All nodes, from *s(0)* to *s(i)*, connected to the edge routers, are connected by 100Mbps ethernet devices. The connection between the ingress edge and core routers provides the bottleneck of the network, which is configured to 512kbps. Congestion is introduced through the 512kbps link in this network setup. The dsRed object specifies that RED will be used across the connected links where the DropTail object does not use any congestion avoidance mechanism.

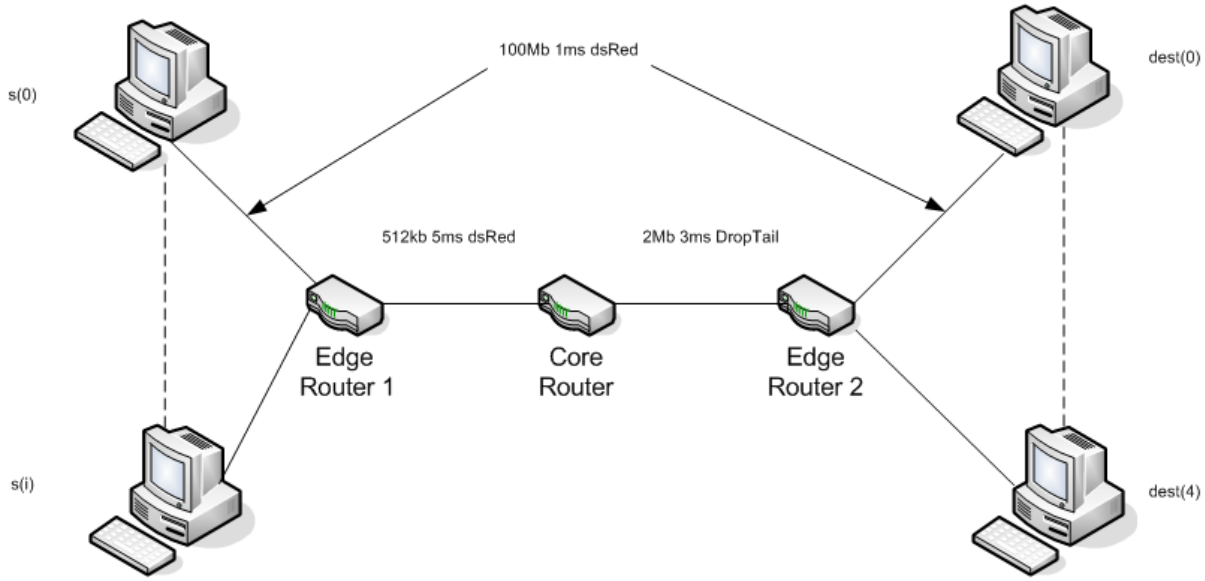


Figure 49 Simulation setup

5.4. QoS PARAMETERS

This section describes the parameters and definitions used by the simulation environment. Firstly, a number of broad definitions of the various business traffic queue are given in section 5.4.1. Secondly, the four separate queues are defined in section 5.4.2. Thirdly, the scheduler definitions used to provide guaranteed bandwidth to the separate queues is provided in section 5.4.3. Lastly, a definition of the RED queuing parameter is given as a fraction of the minimum queue length and maximum queue length to illustrate the influence of the queue parameters by varying queue lengths in section 5.4.4.

5.4.1. Network traffic classes

Table 5 lists the DSCP values that can typically be defined by a service provider for the management of network traffic across the IP backbone. The routing protocol and management packets receive higher precedence than the other packets since the protocol and management packets cannot suffer from loss or delay. Packet classifications occur on edge routers, where these packets are treated on a PHB at the core routers. Layer 3 routed packets marked with various DSCP values are mapped to layer 2 priority values and then marked into a *type of service* (ToS) value. The ToS value is a priority value for layer 2 switched network segments. The highest layer 2 priority value for a packet is switched first.

Table 5 Traffic class separation

Traffic classes (Customer)	DSCP	ToS precedence	Binary DSCP value	Queue class (Provider)
Real time	40	5	101 000	EF
Interactive business	32	4	100 000	IB
Routing protocol	48	6	110 000	BB
Management	24	3	011 000	BB
Bulk business	8	1	001 000	BB
Best effort	0	0	000 000	BE

5.4.2. Traffic regulation parameters

Table 6 lists the provisioning factors used for bandwidth allocation to the various class queues. These factors are based on delay and loss bounds guaranteed by the various queues. However, the provisioning factors do not influence the effect of RED.

Table 6 Traffic class provisioning

Queue Class	DSCP	Provisioning factor (PF)
Expedite forward (EF)	40	4
Interactive business (IB)	32	4
Bulk business (BB)	8	2
Best effort (BE)	0	1

5.4.3. Scheduler weights

Scheduler weights determine the priorities of the various queues. Since the real time queue is a strict priority queue, there is no need to assign a scheduler weight to this class. The remaining classes are assigned weights for scheduling priority.

Table 7 Scheduler weights

Traffic class	Scheduler weight
Real time	0
System administration	5
Management	5
Interactive business (IB)	80
Bulk business (BB)	8
Best effort (BE)	1
Sum	99

5.4.4. Random early detection settings

Congestion avoidance is generally used to maintain buffer sizes, but can also be used to assure that data conform to a certain average delay target specified within customer SLA agreements. RED is configured by setting the *maximum dropping probability* (P_{max}) to 1 for a very high packet drop rate for filled queues. As a consequence, this leaves the minimum and maximum threshold values of the queue size to be set.

Large values for Th_{max} , from the providers side, avoid unnecessarily dropping of packets at the expense of slightly harsher delays. The total buffer size for a class queue needs to be set to 2* Th_{max} [33]. The maximum threshold value does not set an upper bound for the physical queue since the RED algorithm calculates an average queue length and drop packets randomly as discussed in section 4.9.6. Another reason for the large threshold value is that the virtual buffer, used to shape network traffic, also needs allocated space. For instance, the *interactive business* (IB) classes' primary queue acts as a drop tail queue to guarantee packet delivery, since $Th_{max} \approx Th_{min}$. As a consequence, the virtual queues are configured to discard packets.

Packet size distribution

Packet size distributions (PSD) influence the way that the congestion avoidance algorithm treats network traffic and is therefore extremely important when determining the RED queue thresholds. The network traffic that the simulator will generate has to correlate to the cumulative PSD measured in the SPRINT IP backbone [67], as shown in Figure 50.

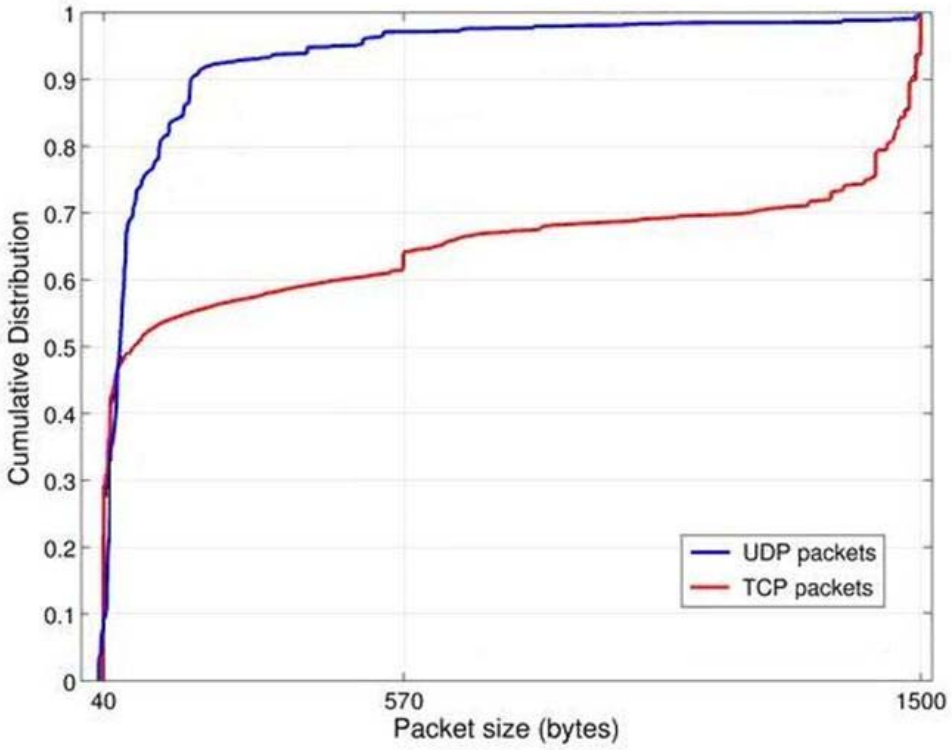


Figure 50 UDP and TCP packet size cumulative distribution

Table 8 illustrates recommendations to the service provider on how the RED algorithm values need to be set for the customer network and provider network. For these calculations, the *provisioned capacity* (BW), and per link *delay budget* (DB), as well as the *average packet size* (APS), are needed to calculate the *pipe bandwidth* (Q), where all the values are used to determine the threshold values shown in Table 8 [75].

From [67], this PSD is used to calculate the *average packet size* (APS). For 100 packets, the PSD = 58*40 bytes + 33*552 bytes + 9* 1500 bytes = 34036 bytes. Thus, the APS = 34036 / 100 = 340 bytes per packet.

Table 8 RED configuration settings

RED parameter	Customer portion	Provider portion
P_{max}	1	1
APS	340 bytes per packet	340 bytes per packet
Q	$Q = \frac{DB * BW}{APS * 8}$	$Q = \frac{DB * BW}{APS * 8}$
Th_{min}	Q	Q
Th_{max}	3*Q	6*Q
Buffer size	6*Q	12*Q

5.5. CONCLUSION

This chapter provides a testbed to conduct research in the congestion avoidance algorithm named RED. NS2 is described as the simulation environment which is followed by the differential service setup specifying queue marking class values and the congestion avoidance algorithm settings.

The following chapter provide results of the RED algorithm performance with varying queue length parameters on core provider links. This will show that RED enables optimal performance for all class queues under network traffic congestion.

CHAPTER 6 EMPIRICAL ANALYSIS RESULTS

This chapter outlines the empirical analysis conducted to investigate the effective configuration of the RED parameters. Section 6.1 gives an overview to the reason why and how the RED mechanism is used within an ISP network to increase network performance. Before the RED mechanism is studied, the simulator needs to generate network traffic that correlates to a real environment. The proof that appropriately generated traffic is produced by the simulator is given in section 6.2. The RED parameters are evaluated in section 6.3. The chapter is concluded with a discussion of the results in section 6.4.

6.1. INTRODUCTION

Congestion is a network condition in any wide area network that an ISP has to control. Network utilization perform worse as congestion increases, and visa versa. This chapter begins by showing how traffic is modelled to determine if the simulator generated traffic that corresponds with previously conducted research in [22]. Traffic modelling is an important aspect for simulation purposes. UDP and TCP traffic clearly behaves differently: UDP is unreliable, thus no attempt to guarantee packet transmissions exists, and TCP is reliable, thus some performance guarantees are essential. UDP transmission sources transmit traffic based on popular stochastic models, such as, the Poisson distribution, exponential distribution or Pareto distribution [29]. TCP transmission sources aims to utilize bandwidth to a maximum. Thus, UDP traffic is divided into CBR and VBR sources, and TCP traffic is an ABR source. The *autocorrelation function* (ACF), as shown in *appendix A*, illustrates the difference in transmission behaviour between TCP and UDP type sources. The different types of traffic are placed into queues which are then subject to a traffic control mechanism.

RED imposes packet loss into a network by discarding packets. The TCP mechanisms are used to control source node transmission speeds and improve network node performance. Results from [30] show that TCP source transmission rates are controlled in order to provide fair resource sharing. Bursty sources fill buffer spaces quickly, and as a consequence, more packets are acknowledged by the destination sources for these sources. This has a negative impact for a number of TCP non–bursty type sources because buffer space is rarely available for them. Therefore, discarding packets randomly for the other bursty sources avoids sudden

bursts of traffic that consume network resources. The parameters of RED are evaluated to determine which parameters maximize network utilization performance. As a conclusion, specific values are used to illustrate various source traffic and network behaviours.

Statistical models were used to represent the sampled data using statistical distributions. The Poisson distribution is simple due to the fact that the *average* (μ) and *variance* (σ^2) parameters are almost equal to each other. It is possible to illustrate both values for the Poisson distribution with one graph. The normal distribution is used whenever the average and variance values differ significantly for larger statistical deviations. As a consequence, the average and variance values need to be plotted. To verify that a normal distribution can be used for the sampled data, the *Kolmogorov-Smirnov* (K-S) and *Anderson-Darling* (A-D) [76] tests were used, as discussed in *appendix A*.

In the case where both the Poisson and normal distributions fail to model the captured data, the Weibull distribution was used. A difficulty with the Weibull distribution is that three parameters namely η, β and γ characterise this distribution, thus requiring that multiple graphs for each Weibull parameter needs to be plotted.

The data captured from the simulations was modelled with the most appropriate distribution for data analysis. *Appendix C* provides data samples to support the observations made for the sections that follow.

6.2. TRAFFIC MODELLING

Network traffic is dominated by mainly two protocol types, UDP and TCP traffic. Therefore, protocols other than these two do not influence analysis on network traffic behaviour. The distinction between these two types of traffic can be shown by the use of an *autocorrelation function* (ACF) that is discussed in *appendix A*. The captured traffic correlation at the source's edge router is shown in the normalized traffic autocorrelation [22], shown in Figure 51. The lag parameter is the sample number that was taken from the simulation sampled data.

To illustrate traffic autocorrelation, multiple edge routers are connected to a single core router with the node utilization and queue size measurements conducted at the core router. It can be seen in Figure 51 that the traffic generated by the simulator exhibits short- and long-range dependant traffic behaviour. As a result, the simulator generates adequate network traffic that was used for analysis.

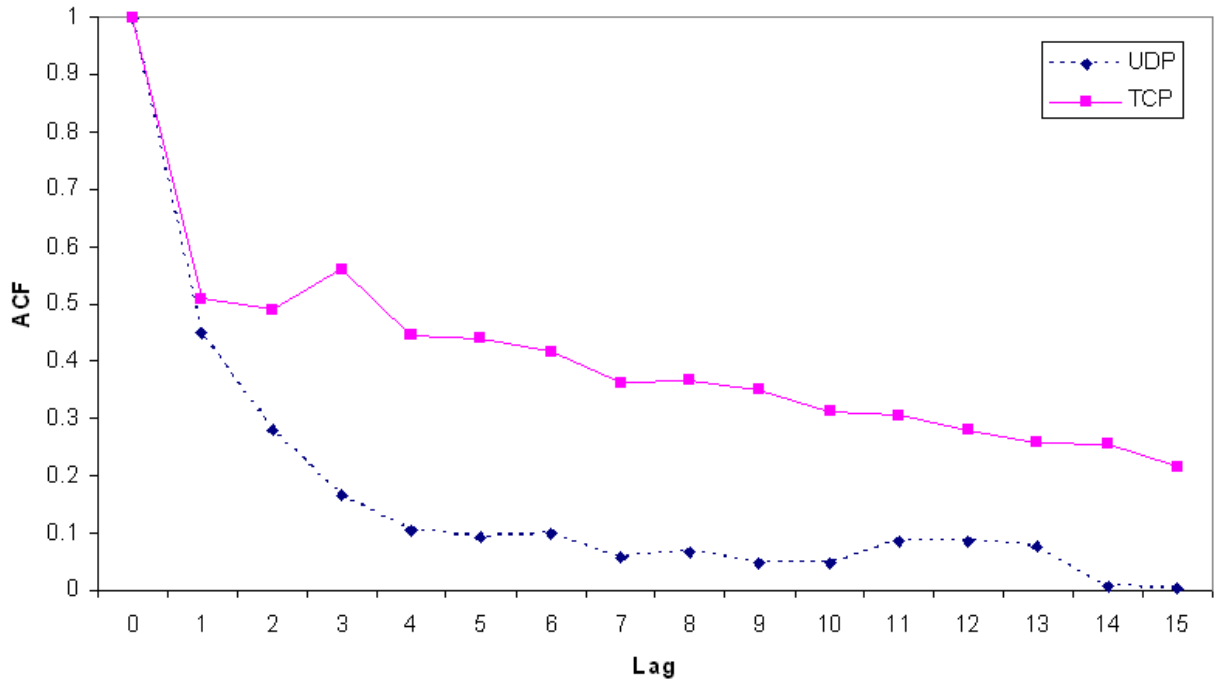


Figure 51 Autocorrelation of UDP and TCP traffic

6.3. RED PARAMETER PERFORMANCE EVALUATION

Simulations [33] show that delay and packet loss bounds can be determined by varying the QoS parameters. However, UDP and TCP traffics' quality of service are managed differently: UDP type traffic is limited by the number of bits that can be sent and TCP type traffic is delayed. This section does not consider the effects of the QoS traffic limiting bound mechanisms on the UDP traffic since RED uses capabilities within TCP. The parameters of interest are the RED queue length parameters, since these parameters control network traffic as a result of congestion as discussed in section 4.9.6. Each parameter was obtained by simulating the network five times with randomly generated network flows that closely resemble the PSD of [67], also discussed in section 4.6.

TCP traffic is dynamic in nature, changing transmission rates based on packet delay and loss. Therefore, much effort is required to control these connections to deliver maximum throughput and minimum delay and loss. The combination of shapers and active queuing mechanisms achieve these goals.

The queuing management algorithms ensure that queues do not overflow and cause customer connections to receive more service than other customer flows. Once the queue fills past a minimal threshold (Th_{min}) packets are dropped. If the queue being used exceeds a maximum threshold (Th_{max}) all the arriving packets are dropped. For the purpose of this dissertation the defined threshold function,

$$\alpha = \frac{Th_{\min}}{Th_{\max}} \tag{6.1}$$

is used as a means to present the measured results from the simulations relating to the RED parameters. It is important to observe the effect that the RED threshold values has on packet loss and delay, since packet loss controls customer bandwidth usage and decreases packet delay due to shorter queues. The succeeding subsections attempt to find an appropriate *threshold fraction* (α) such that best performance measures such as node utilization can be achieved. Therefore, it becomes relevant not only to observe delay and packet loss measures, but also the effects on the number of packet transmissions, transmission rates, and queue utilization.

6.3.1. Packet loss probability

TCP traffic's throughput is a function of delay and packet loss, as was discussed in chapter 1. This section focuses on packet loss introduced into a network environment. The two queuing methods of interest are drop tail and RED managed queues. These two methods operate very differently, as was discussed in chapter 3.

The implementation of the RED algorithm within the simulation environment allows the analysis of packets being discarded, caused by RED, and dropped, caused by buffer overflow. Test conditions, as shown in *appendix D*, determine whether a packet may enter a queue calculated by the RED queuing function within the simulation. The simulator is able to determine if a packet loss occurs due to buffer overflow or RED.

Figure 52 tests the likelihood as percentage model of fit, with the A-D goodness-of-fit test as discussed in *appendix A*, if the sampled data is normally distributed. The graph's *P* value does not reject the null hypothesis that the data is normally distributed. The only important parameters for the A-D test are the average and standard deviation values.

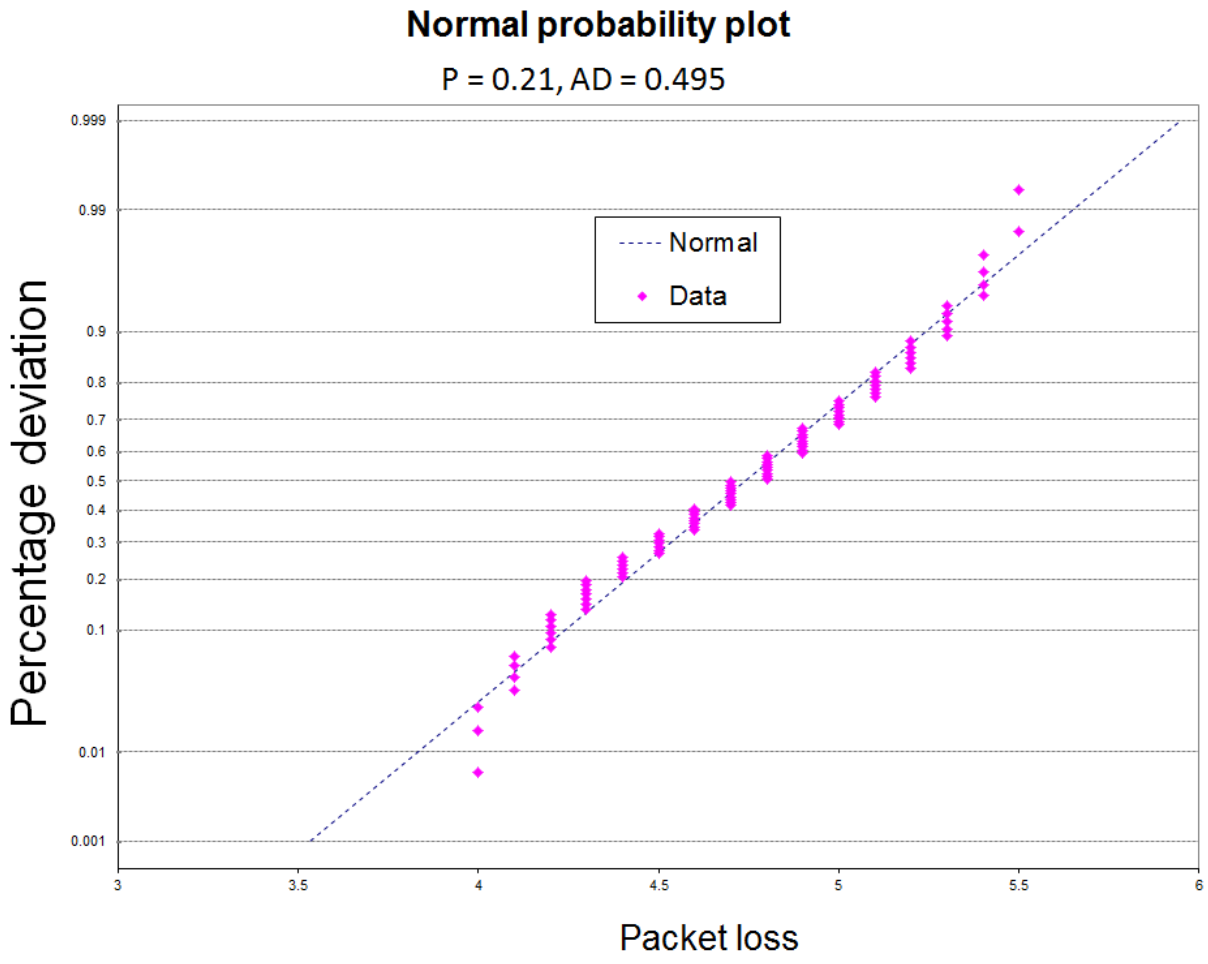


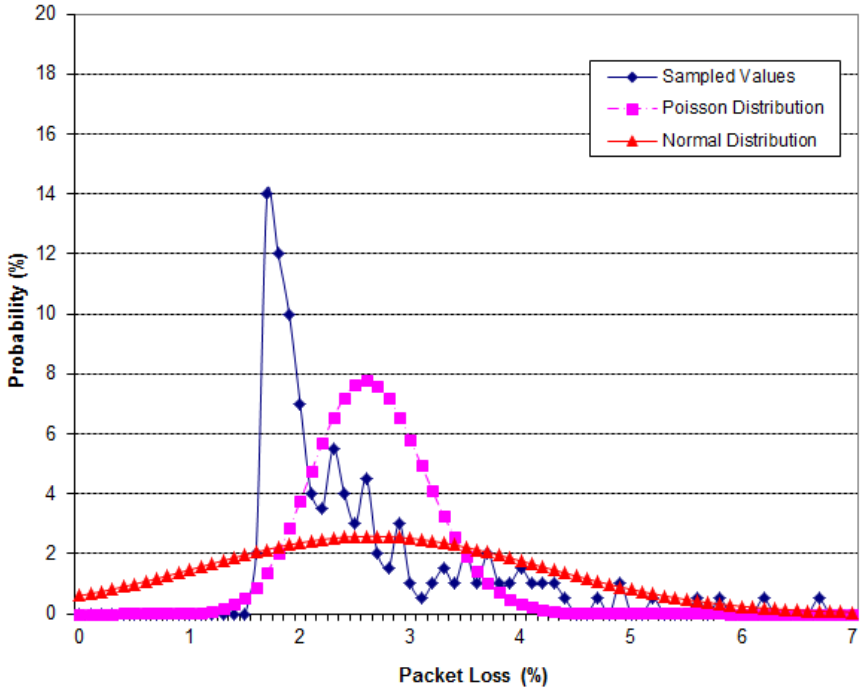
Figure 52 Normality test for packet loss probability employing RED

Figure 53 shows that drop tail queues (Figure 53(a)) and RED managed (Figure 53(b)) queues are distributed differently across 8000 sampled values. The drop tail queue behaviour shows a heavy-tailed characteristic, which cannot be sufficiently represented by a probability distribution whereas the RED queue appears to be a normal distribution. This is an important result due to the observation that the tail drop queues' random behaviour prohibits any type of guaranteed agreements concerning packet loss and delay.

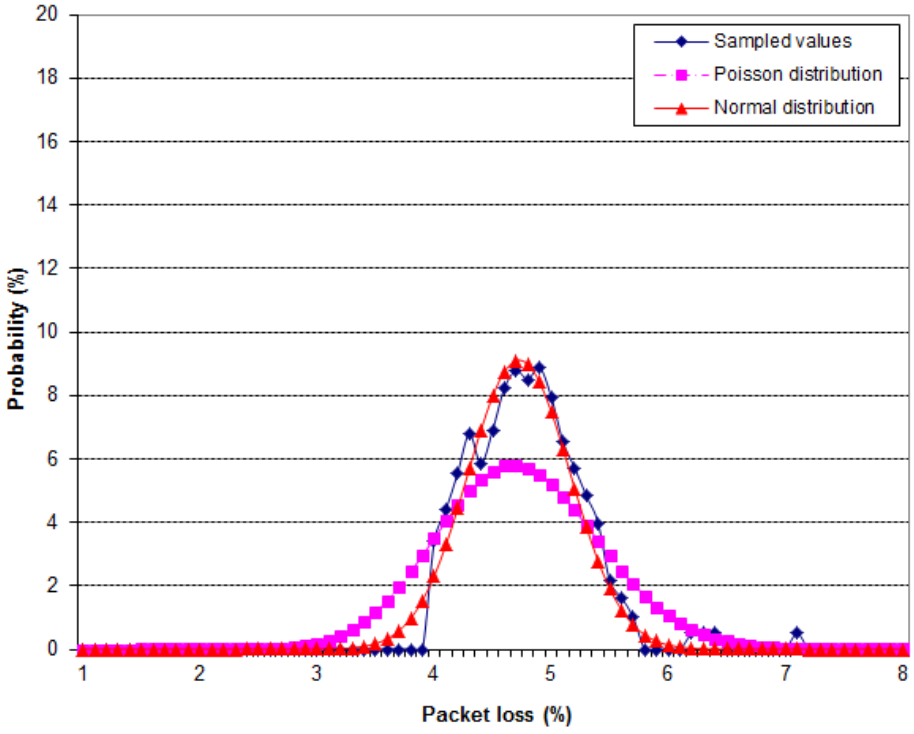
Also, the RED enabled queue shows that a distribution can be used to present captured data. Because packet loss affects networking dynamics, it is important to observe the TCP protocol traffic and resource efficiency as it is affected by RED. The following sections' analysis is only conducted towards finding optimal values for the RED algorithm.

Figure 54 shows packet loss as a result of the RED algorithm and buffer overflow. Buffer overflow leads to TCP sources being unable to determine if congestion is present within a network, as a result of bursty traffic. However, RED randomly discards packets resulting in higher throughput for sources that are not bursty.

Figure 55 illustrates the packet loss probability for a range of α values. It is observed that a high percentage of packet loss is present at $\alpha = 0$ causing excessive packet drops. Thus, a low throughput is expected for all traffic sources with a low value of α. It is observed at any other α value that packet loss is more controlled and a higher throughput is expected.



(a) Drop tail queue packet loss probability



(b) RED managed queue packet loss probability

Figure 53 Packet loss probability for two differently managed queues

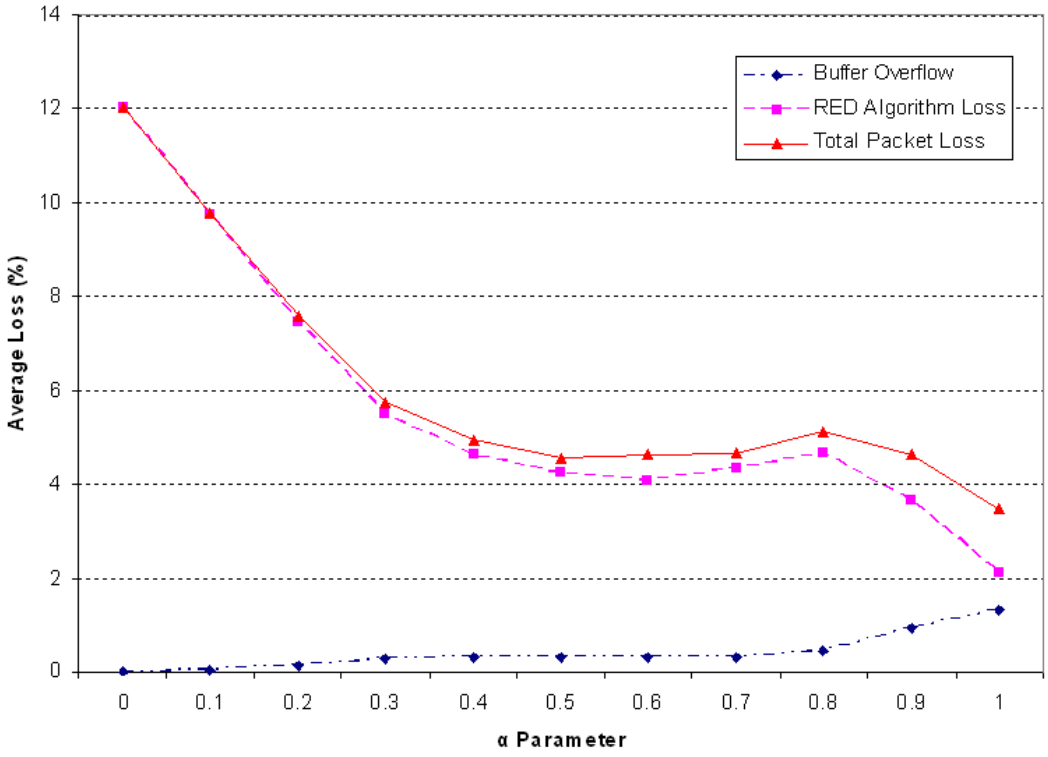


Figure 54 Buffer overflow packet loss and RED packet loss vs. RED parameter manipulation

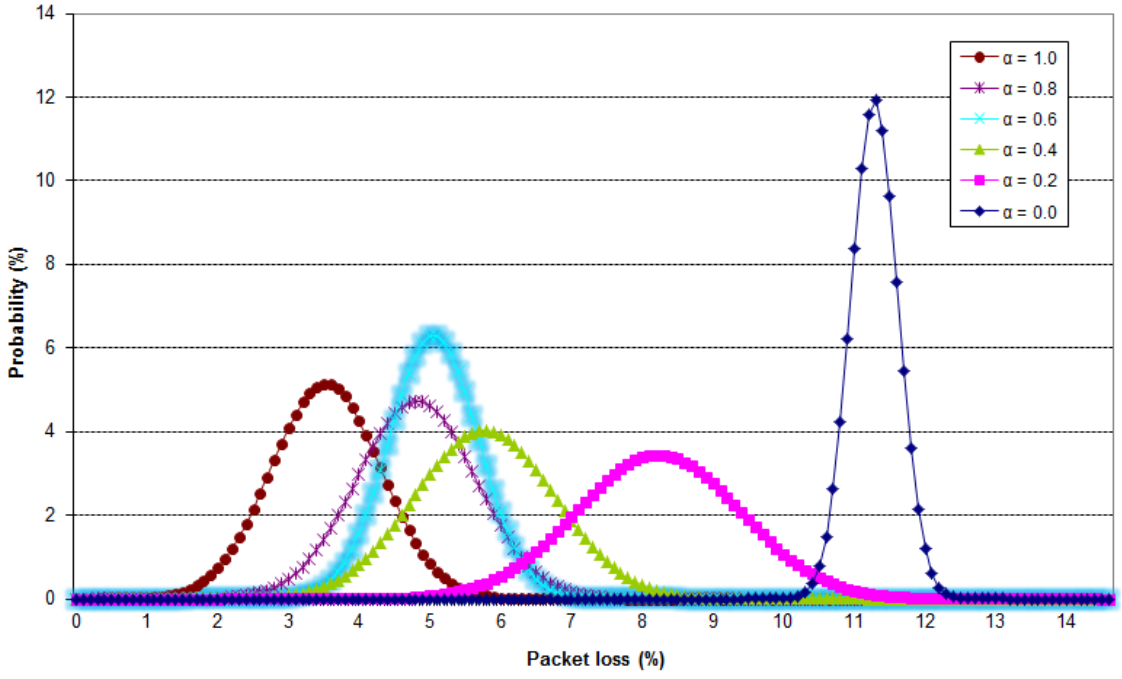
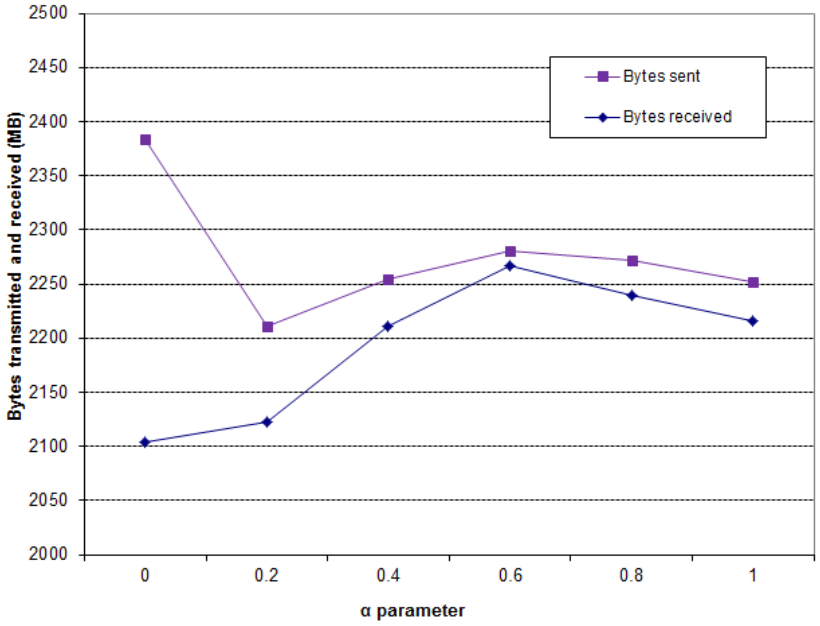
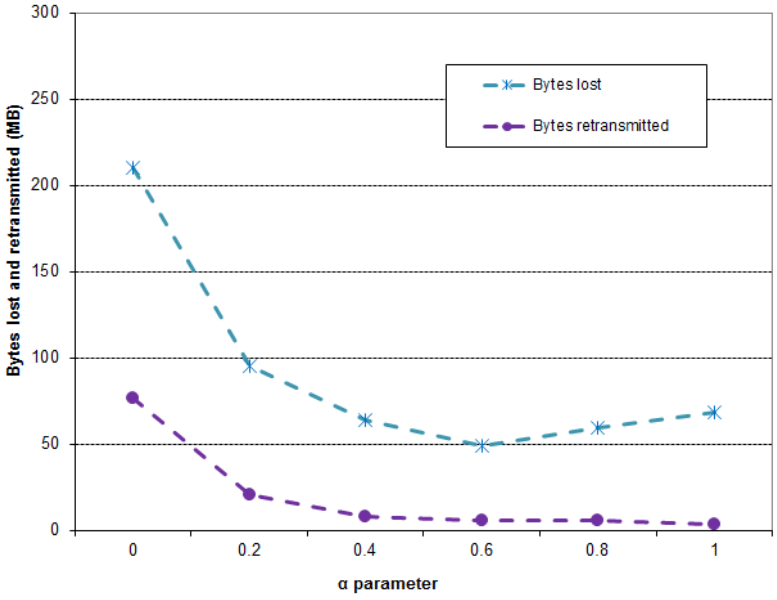


Figure 55 Total packet loss probability plots varying the α value

Even though α = 1 shows the least amount of packet loss, the following sections show that other values of α perform better. The number of bytes received shown in Figure 56 (a) verifies the relationship towards packet loss verified by Figure 56 (b) as mentioned in chapter 2, section 2.2.



(a) The number of bytes transmitted and received



(b) The number of bytes lost and retransmitted

Figure 56 Dynamic TCP protocol behaviour concerning throughput

Within the network topology, the link propagation delay is set to a small value. The reason for the little delay was to concentrate on packet loss, without the TCP time-out timers affecting the TCP traffic dynamics. Therefore, the threshold function shows that the best performance is achieved at $\alpha = \frac{Th_{mon}}{Th_{\max}} = 0.6$.

Observations on the packet loss probability results are summarized below:

- Drop tail queues behave statistically different to packet loss caused by RED managed queues. Due to the heavy-tailed distribution of packet loss caused by tail drop the network throughput cannot be accurately estimated. The RED managed queue provide a means to estimate throughput with regards to an evenly distributed packet loss.
- Statistically, the drop tail queues' packet loss probability does not correlate to any statistical distribution as shown in Figure 53 (a). This result shows that the packet loss distribution is heavy-tailed. As a result, not all TCP sources are able to adjust their congestion windows accordingly to maintain a stable transmission rate.
- Statistically, the normal distribution can be used for a RED managed queue behaviour as shown in Figure 52 and Figure 53 (b). This result shows that the packet loss probability is normally distributed; hence, throughput for all sources can be determined more accurately than drop trail queues. As a result, the RED mechanism enables the TCP sources to adjust their congestion window parameters to maintain a stable transmission rate over time, as shown in *appendix C.2*.

6.3.2. Congestion window probability

TCP aims to transmit data with regards to the amount of congestion which a network experiences by adjusting the congestion window parameter that controls the transmission rate. Therefore, TCP determines network congestion based on packet loss; high packet loss indicates high congestion and visa-versa. Figure 57 shows the TCP congestion window probability distribution obtained from simulations by varying the threshold fraction function, as defined at section 6.3.

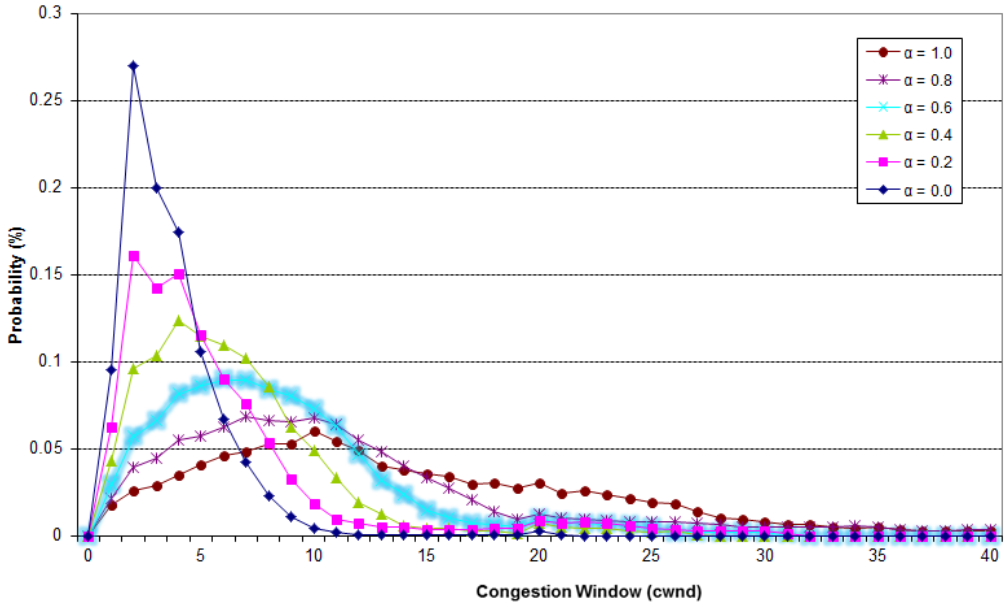


Figure 57 Congestion windows probability obtained from sampling

By observing the effects of the threshold factor as illustrated in Figure 54 and Figure 57, it can be seen that packet loss influences transmission rates of TCP sources. Even though packet loss decreases at high values of α, TCP transmission rates decrease which leads to poor resource utilization. The sampled values are modelled with the Weibull distribution as described in *appendix B*, given in Figure 58.

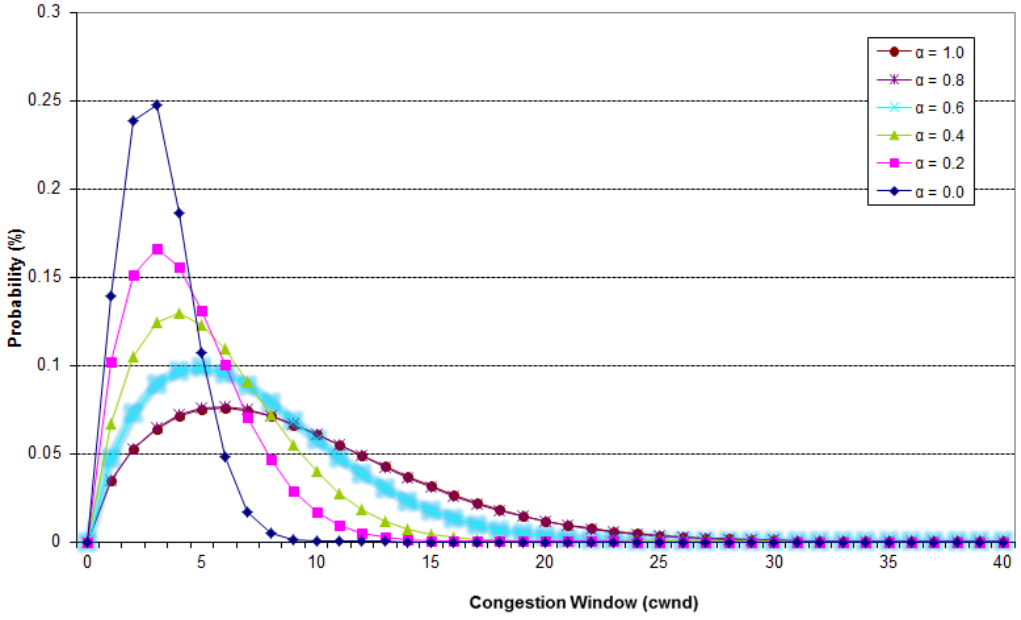
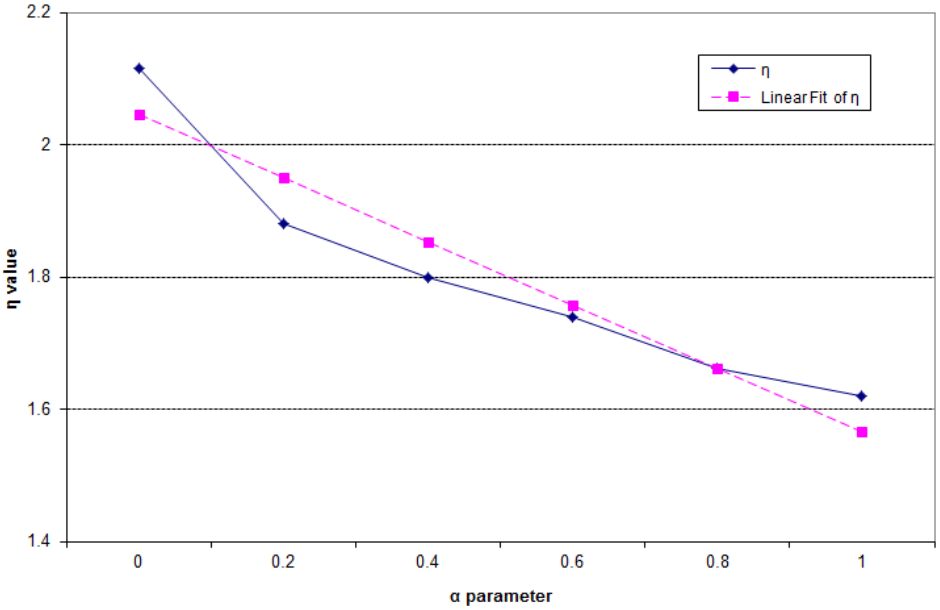
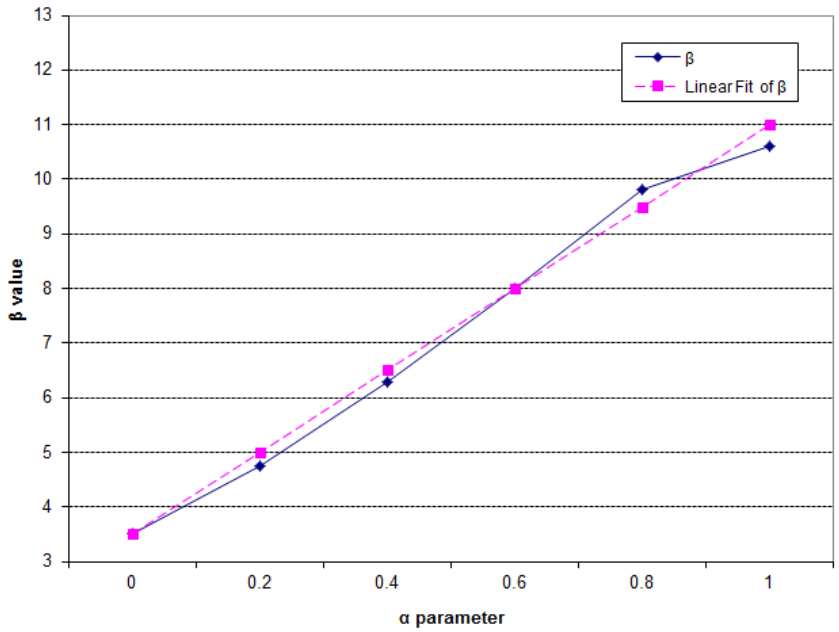


Figure 58 Congestion window probability modelled with the Weibull distribution

Appendix B provides a method to calculate the η (scale parameter) and β (shape parameter) distribution parameters, assuming that γ=0 (offset parameter). The η and β values are linear in regards to the RED threshold values, as illustrated in Figure 59. Hence, the TCP transmission rates are regulated across a network based on the threshold values.



(a) Linear fit of the η value to the threshold fraction



(b) Linear fit of the β value to the threshold fraction

Figure 59 Weibull distribution parameters for congestion window fittings

It can be observed that the graphs for Figure 59 are linear. Equations 6.2 and 6.3 were determined through a linear approximation by using two points from the graphs to calculate the slope and y axis intercept point to illustrate this relationship. This result shows that the Weibull distribution parameters are linear with regards to the RED parameters, where

$$\eta = -0.46 * \alpha + 2.03 \tag{6.2}$$

and

$$\beta = 7.43 * \alpha + 3.5 \tag{6.3}$$

The above equations show that the η parameters' slope decreases slowly while the β parameters' slope increases fast. Therefore, there exists a relationship between TCP transmission rate and the RED parameters. As a result, RED controls the TCP source transmission rates to avoid congestion, and with the use of the Weibull distribution the transmission rates can be predicted.

Observations on the congestion window probability results are as follows:

- The Weibull distribution can be used to model the congestion window probabilities.
- The η and β values have a linear relationship to the α parameter.

6.3.3. Utilization probability

Utilization provides a measure of how effectively received packets are transmitted by a network device. As shown with equation,

$$\rho = \frac{\lambda}{\mu} = \frac{\text{Transmitted bits}}{\text{Received bits}} \tag{6.4}$$

utilization is the relationship between the bits received and sent at the network device. Hence, utilization shows that the number of bits received by a network device needs to be as close as possible to the number of bits transmitted by the network device.

Figure 60 illustrates the probability that the highest node utilization is achieved for α = 0.6. The utilization probability for the various α values for a given utilization x is given by $P[X_\alpha=x]$. To determine the best α value the highest cumulative probability $P[X_\alpha>x]$ needs to be determined. By noting x as utilization the probability $P[X_\alpha>\text{Utilization}]$ is defined.

Thus, the P[$X_{0.6}$>94%] shows to be more than P[X_0>94%], P[$X_{0.2}$>94%], P[$X_{0.4}$>94%], P[$X_{0.8}$>94%], and P[$X_{1.0}$>94%]. The utilization probability of P[$X_{0.6}$>94%] corresponds to the observations in Figure 56 (a) regarding the number of received bytes.

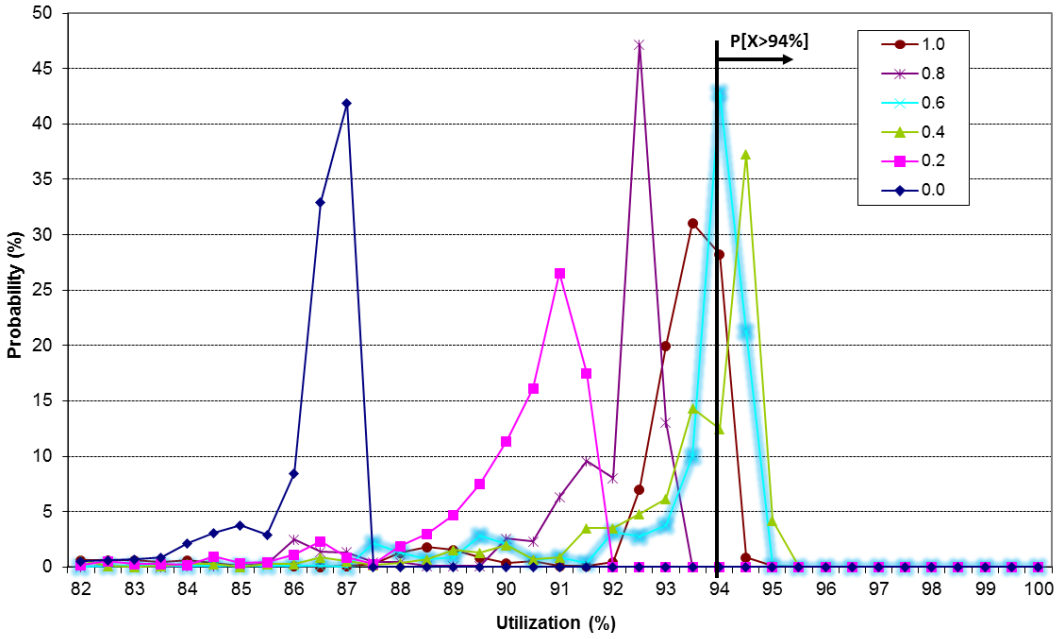


Figure 60 Node utilization probability from sampled data

Figure 61 illustrates the calculated Weibull distributions from the sampled data points. The same result is concluded for P[$X_{0.6}$>94%], as with Figure 60.

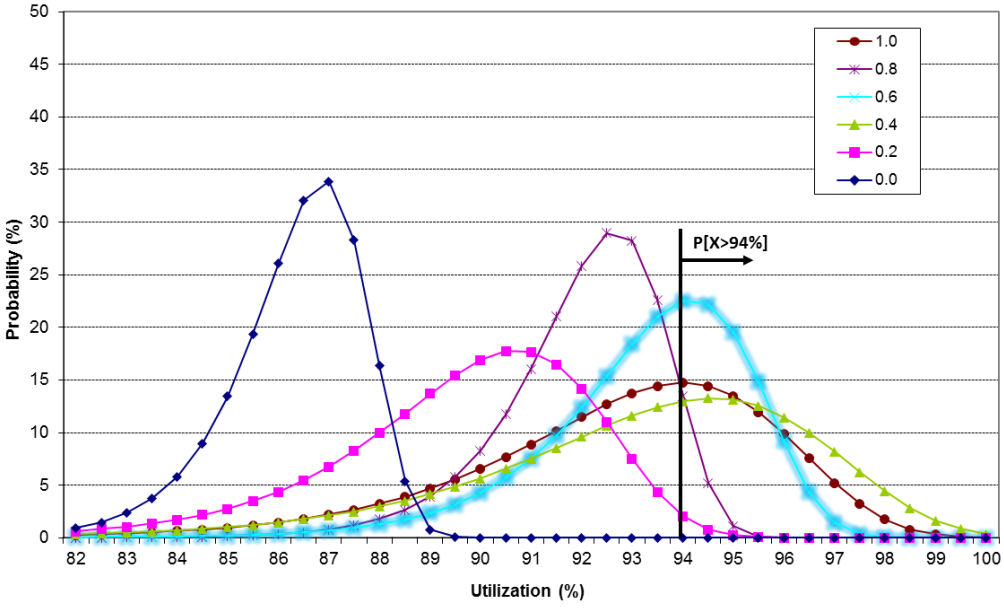
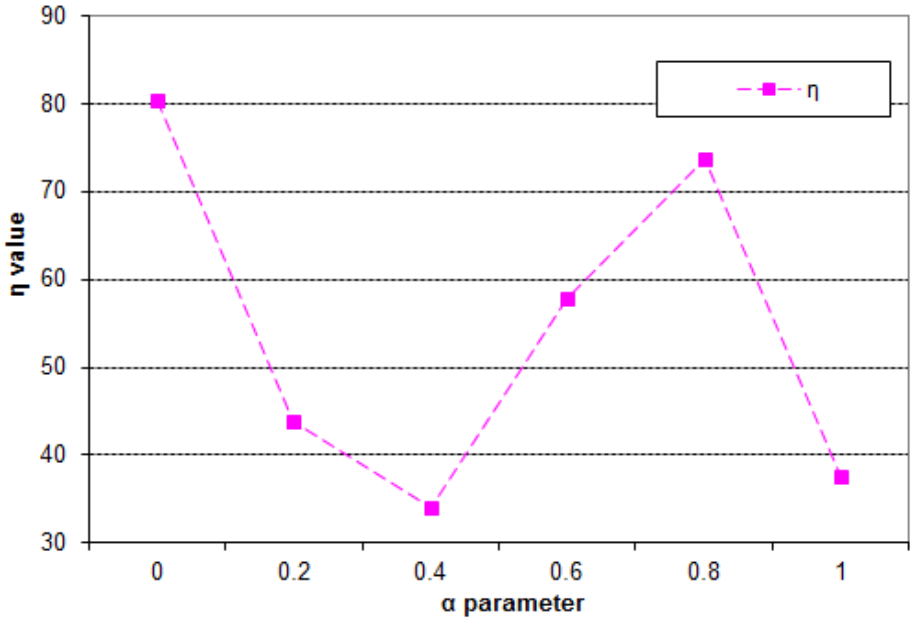
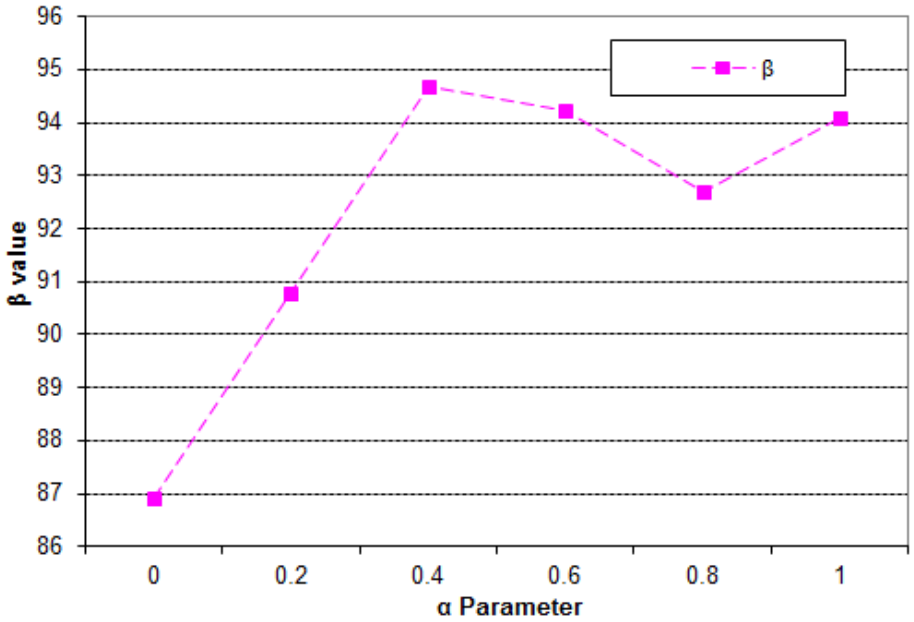


Figure 61 Node utilization illustrated by the Weibull distribution

Figure 62 (a) and (b) illustrate that the node utilization can not be predicted by a statistical distribution by the variations of the RED parameters. Due to the number of TCP sources adapting their transmission rates, the network node utilization can not be accurately modelled by a Weibull distribution by the variation of the α parameter.



(a) The η value with regards to the threshold fraction



(b) The β value with regards to the threshold fraction

Figure 62 Weibull distribution parameters for node utilization

Observations on the node utilization probability results are as follows:

- The normal distribution cannot be used to analyse the utilization probability. Comparing Figure 60 and Figure 61, it is even difficult to use the Weibull distribution to model node utilization. Figure 62 (a) and Figure 62 (b) also show that the scale and shape parameters for the various α values are not linear and do not give a predictable behaviour.
- As the *total packet loss decreases* in terms of buffer overflow and packet discarding, as shown in Figure 54, node *utilization increases* as shown in Figure 60 and Figure 61. This result shows that the transmission nodes accurately determine their available bandwidth by determining whether congestion is present across the network.
- The utilization variation shows that the TCP sources are controlled at $\alpha = 0.6$ since transmission rates are regulated by RED. The TCP sources transmit slower once congestion is detected, which decreases utilization as discussed in section 6.3.2. When the network node is no longer congested, the transmission nodes increase their transmission rates as shown in *appendix C*.

6.3.4. Queuing probability

This section aims to determine how the queues, managed by RED, are influenced by the changing α parameter. Figure 63 tests the likelihood, with the K-S test, as discussed in *appendix A*, that the sampled data is normally distributed. The graph's *P* value does not reject the null hypothesis that the data is normally distributed. Figure 64 shows that the normal distribution can be used to analyse queuing probability at $\alpha = 0.6$ which correlates to any other value of α.

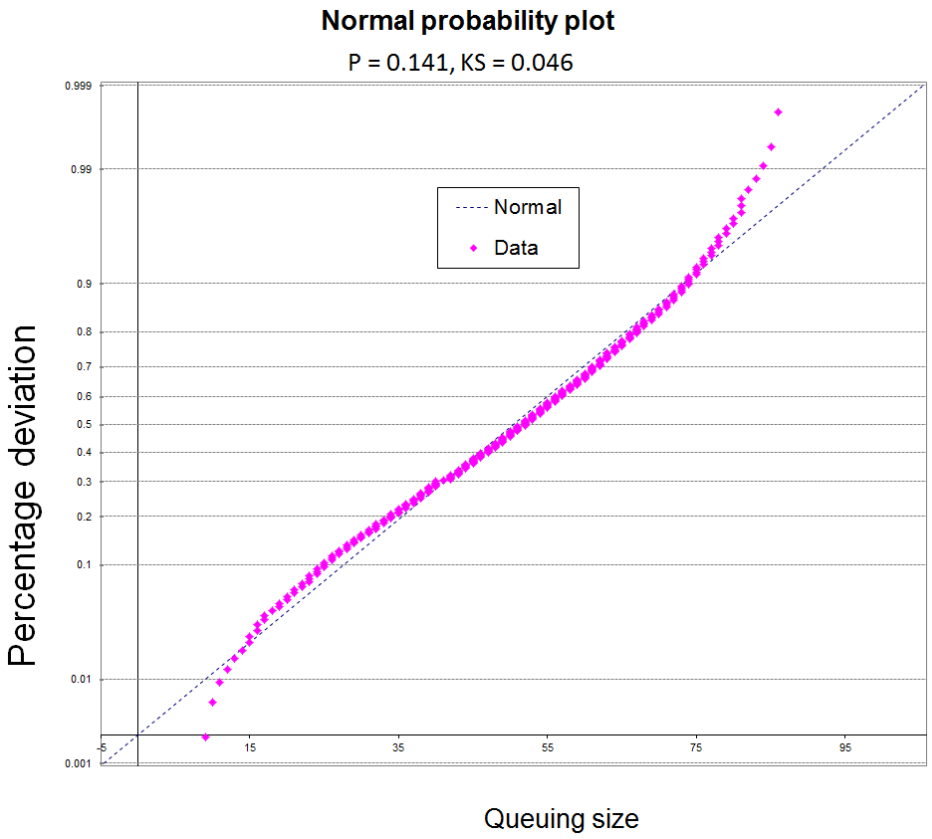


Figure 63 Normality test for the queue size probability employing RED

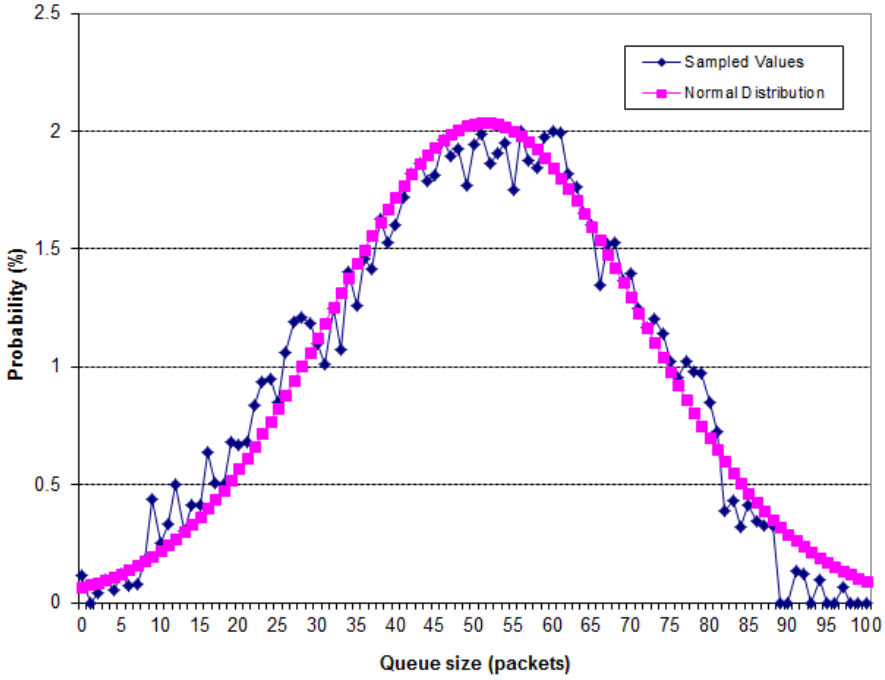
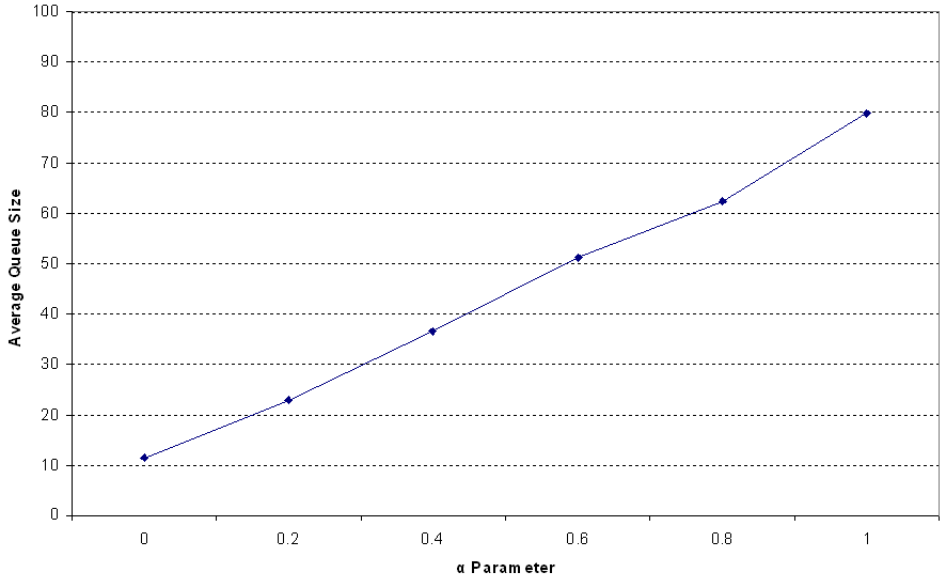
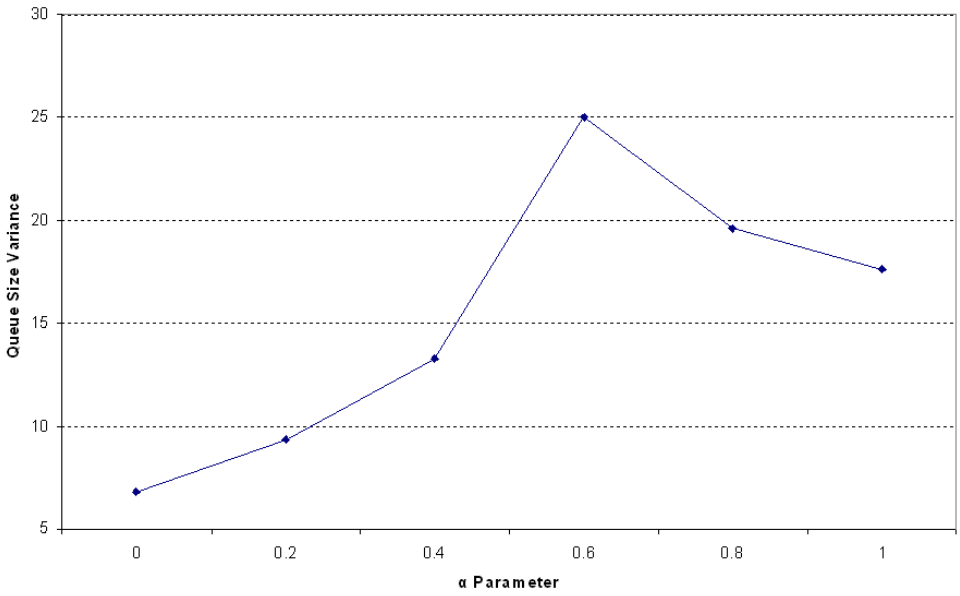


Figure 64 Queue size probability modelled with the normal distribution

As was discussed in chapter 4, the packet discard probability is defined as a linear function [7]. As a consequence, it is expected that the average queue length should also be linear. This expectation is illustrated in Figure 65 (a). The scheduler effectively services queued packets by having time to clear the various queues at high utilization when RED performs packet discarding.



(a) Average queue size



(b) Queue size standard deviation

Figure 65 Normal distribution parameters for queue size

Observations on the queue length probability results are:

- The normal distribution can be used to analyse the queue length probability as is tested with the K-S test.
- The average queue length is linear, as expected since the RED drop probability has a linear relationship as shown in [7].
- The queue variation shows that the queued packets receive effective service at the network device at $\alpha = 0.6$. When the queue is filled, the RED algorithm notifies the sources to transmit at a slower rate so that the currently queued packets can receive service.

6.3.5. Queue delay probabilities

This section shows how delay is influenced based on the threshold factor α. The normal distribution is used to fit the queuing delay and TCP round trip times. Little's theorem, as discussed in chapter 2, states that the average experienced delay is proportional to the number of customers within the queue and the service capacity. Therefore, the queuing delay was measured as a function of queue length (packets) and the server rate (packets per second). Thus,

$$Queue\_delay = \frac{\text{Packets in queue}}{\text{Packets serviced per second}} \tag{6.5}$$

The *round trip time* (RTT) values were obtained from the simulation environment, where the simulator record the round trip times for each packet acknowledged at the source node. The queuing delay and RTT sampled values where tested with the A-D test. Figure 66 and Figure 68 show that both the A-D test *P* values do not reject the null hypothesis that the data is normally distributed. Figure 67 and Figure 69 show that the corresponding delays experienced by the network indeed have a normal distribution.

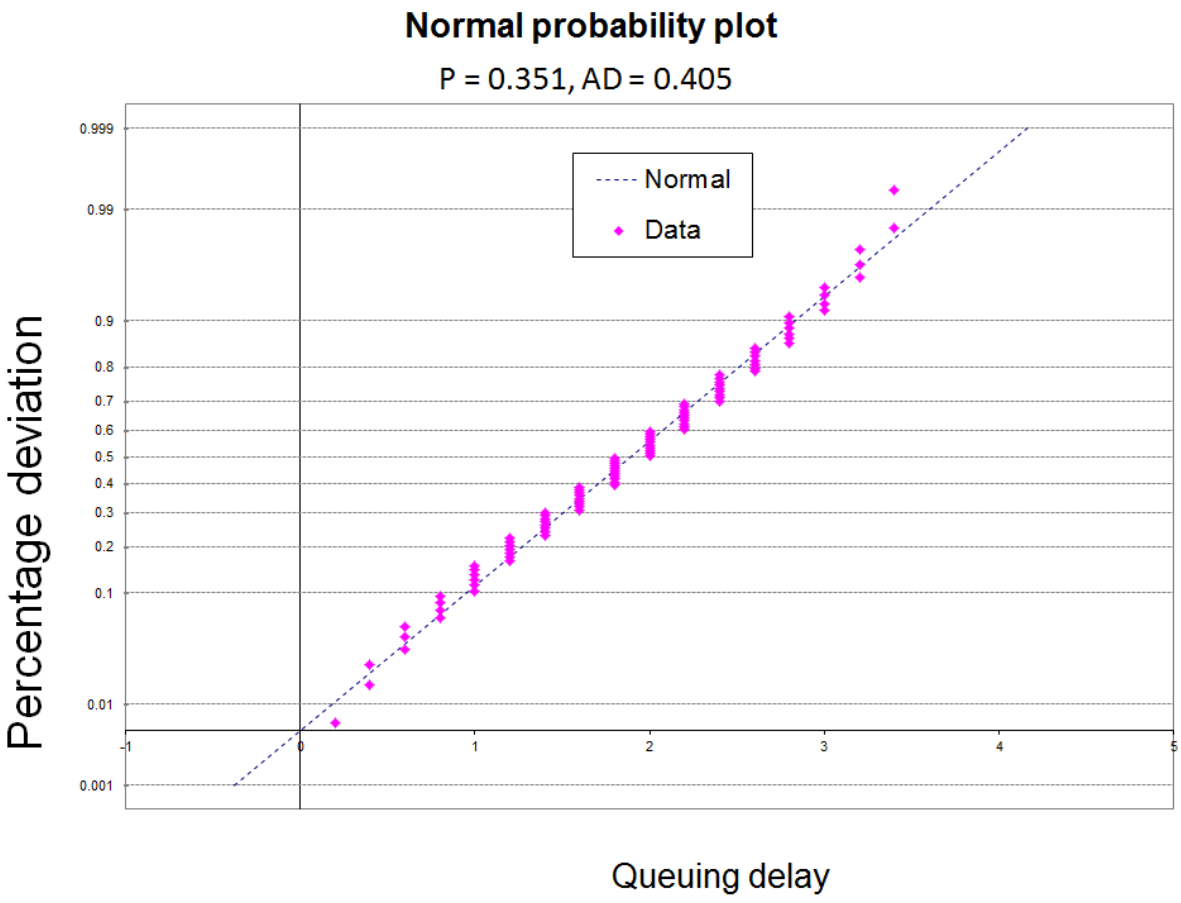


Figure 66 Normality test for queuing delay probability employing RED

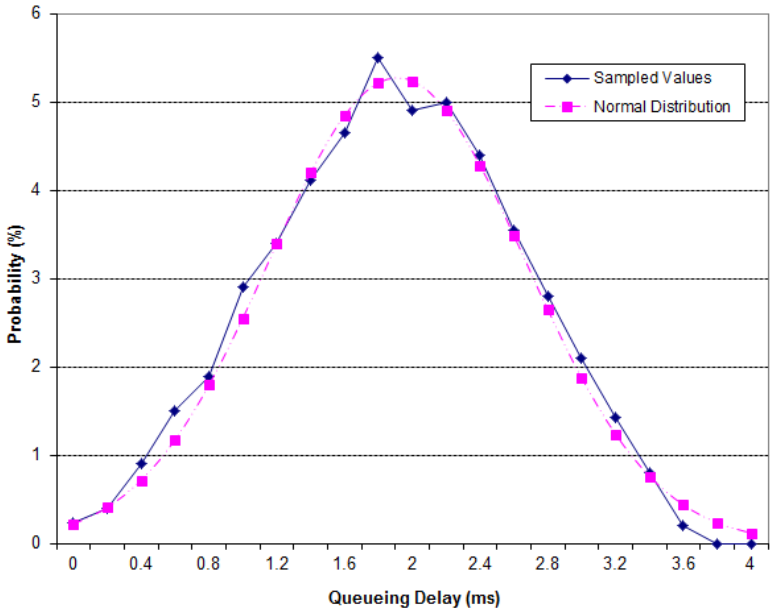


Figure 67 Queuing delay modelled to the normal distribution

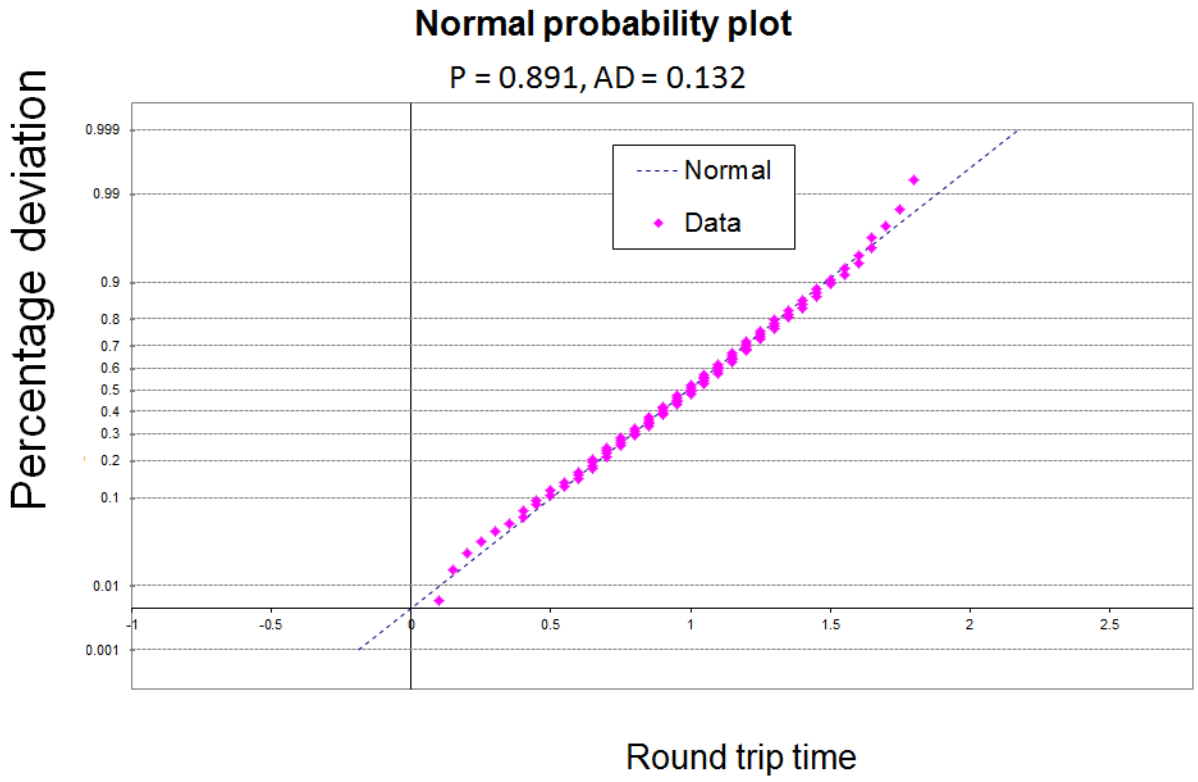


Figure 68 Normality test for packet round trip time probability employing RED

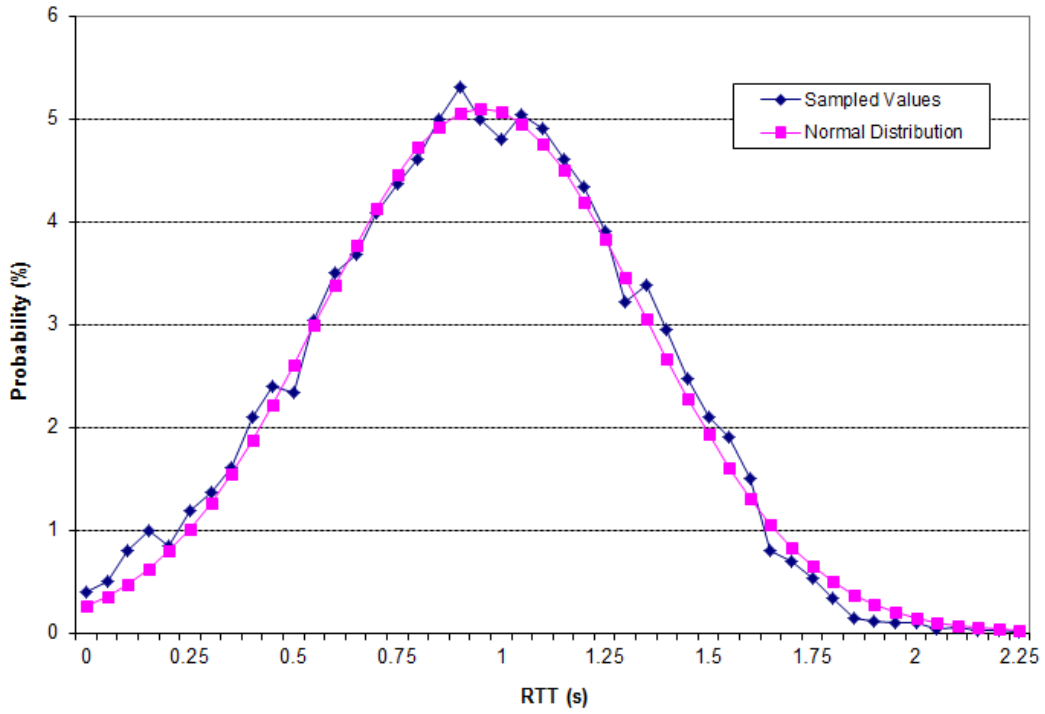
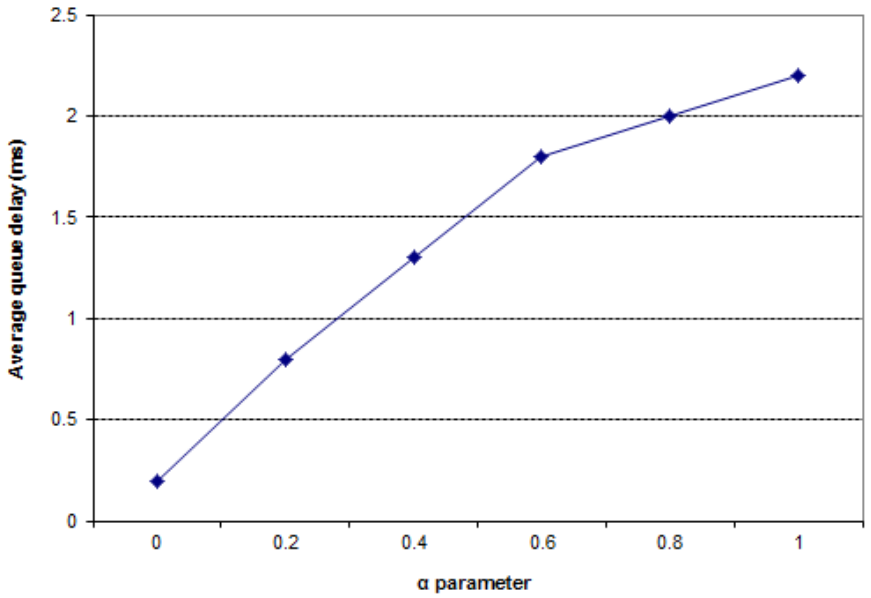
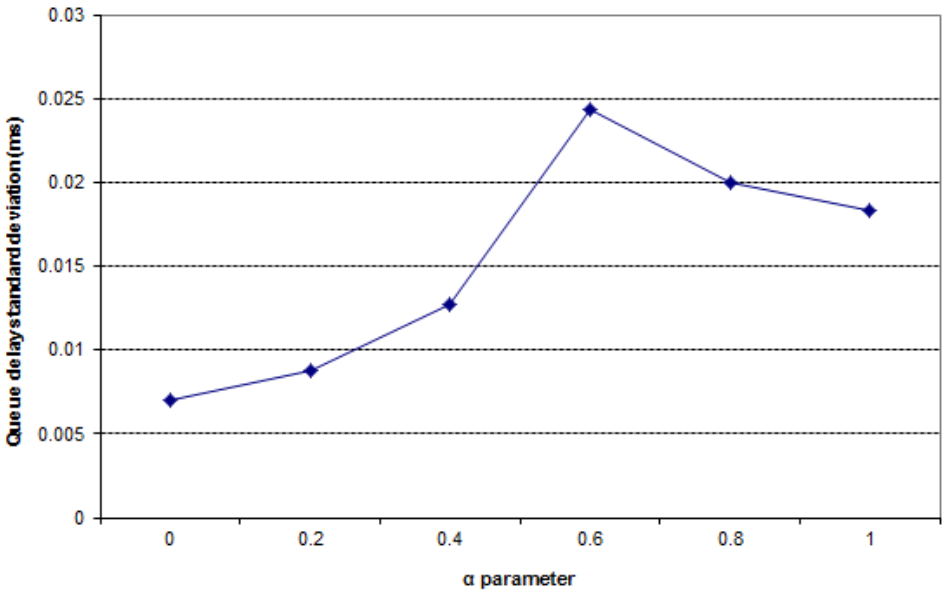


Figure 69 Round trip times (RTT) normal distribution

Figure 70 (a) illustrates that long packet queues, when compared with Figure 65 (a), introduce delay to packets when the packets are queued. From this observation it can be seen that queuing delay is added at a networking device. Figure 70 (b) illustrates that the queuing delay standard deviation behaves like the queue size deviation observed in Figure 65 (b).



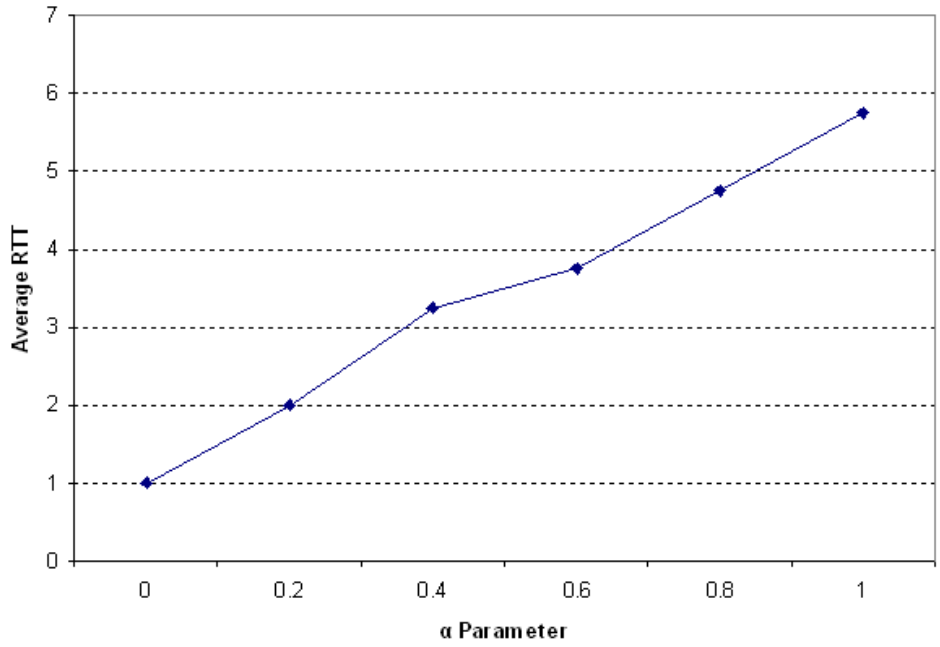
(a) Average queue delay



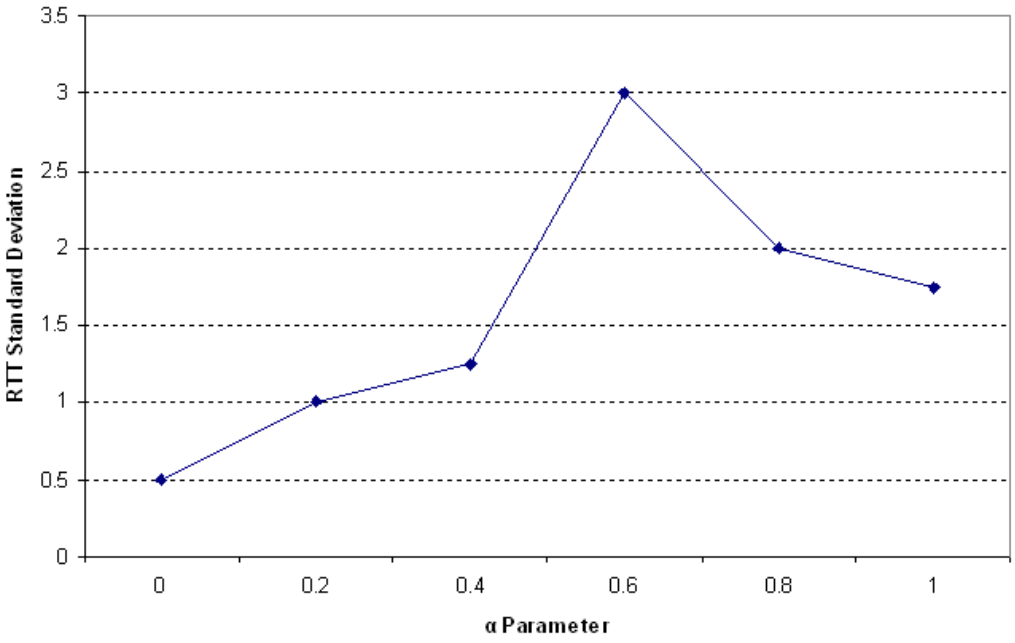
(b) Queue delay standard deviation

Figure 70 Normal distribution parameters for queue delay

Figure 71 (a) illustrates the same behaviour on TCP round trip time as shown in Figure 70 (a) due to queuing delay. Figure 71 (b) illustrates that TCP round trip time is also closely related to the queuing delay deviation as shown in Figure 70 (b).



(a) Average RTT



(b) RTT standard deviation

Figure 71 Normal distribution parameters for round trip time

Observations on the delay parameters probability results are:

- The delay parameters' probability function is the same as the queue length probability function, which is the normal distribution. However, this result does not consider the effects of propagation delay due to the negligible small delay that is added on the links.
- Due to packet queuing delay to network traffic that passes across each networking devices the round trip time delay will increase for each added hop. From Figure 70 (a) and Figure 71 (a) it can be concluded that each hop that employ congestion control keeps the queuing delay constant across each hop.

Section 6.3 illustrated by defining α as a relationship of RED parameters that packet loss is statistically normally distributed as illustrated in Figure 52 and shown in Figure 53. Packet loss is shown in Figure 54 and Figure 55 illustrating that the least number of packets are discarded at $\alpha = 1$. However, Figure 56 illustrate that the most successfully transmitted packets are received at the end host at $\alpha = 0.6$. Additional observations show that TCP sessions transmit the most packets at $\alpha = 0.6$ as illustrated by Figure 57. The network node that employ RED is utilized the best at $\alpha = 0.6$ as illustrated by Figure 61. Figure 63 illustrate that the queuing buffer sizes (queued packets) are statistically normally distributed and shown in Figure 64. The delay affects that packet queuing cause to packets are illustrated in Figure 67 in one direction and both directions in Figure 69.

6.4. CONCLUSION

This chapter depicts that congestion avoidance is essential within a service provider network for improved networking performance. The initial goal for the generated traffic was achieved by the appearance of short- and long-range dependence when the sampled traffic throughput values are subjected to the autocorrelation function. This correlation shows the nature of UDP and TCP as discussed in Chapter 2, section 2.4.2.

The definition of the threshold fraction (α) was used to show how the RED values influence performance measures. It was shown that the packet loss probability, congestion window size probability, node utilization probability and queuing probability for $\alpha = 0.6$ gave the best performance measures. It can therefore be concluded that RED does improve network performance for specific values of α.

CHAPTER 7 CONCLUSION

7.1. SUMMARY

Service providers aim to utilize network resources to achieve optimal performance to their clients. Chapter 2 provided information regarding the agreements that are involved from a providers' side. The different aspects of which a telecommunication network consists are described to give a high level theoretical background. These factors were widely studied by academics in order to achieve maximal performance from a network.

However, network resources use various mechanisms to achieve a certain level of *QoS*. Chapter 3 outlined the most important aspects of *QoS* which service providers use as a benchmark for networking performance. These factors were described to show how they relate to network performance. Also included within this chapter was the development of the network architecture to cater for the QoS requirements.

The *various technologies* that an IP backbone consists of were provided within Chapter 4. This information aids to understand how wide area networks treat packets that traverse across a backbone infrastructure. It was shown that approximately 85% of internet traffic consists of *TCP protocol* traffic, while approximately 15% consists of *UDP protocol* traffic. Since these two protocols dominate network traffic, it is important to understand how they operate. The *differentiated service* (DiffServ) provides service separation to network traffic, therefore, detailed information was provided to describe its operation. Since RED, which is a congestion control mechanism, was the main focus of this dissertation, it was also described in detail and how it relates to the differentiated service architecture.

Chapter 5 outlined the *simulation setup* for the purpose of simulating a customer connection to a service provider network. Also included within this chapter was information for the *QoS mechanisms* that is used with the DiffServ architecture. These settings are recommended by Cisco for service providers for network configuration purposes. Chapter 6 continued to outline the *results* obtained from the simulations representing traffic between a customer and service provider backbone. Based on recommended values, the simulation environment variables were limited to study the effects that RED has on network performance.

7.2. CONCLUSION

Congestion control required knowledge from various fields of study, such as traffic engineering, queuing management and network architecture. This dissertation contributed to controlling congestion, where large traffic volumes are present within high-speed networks. The RED parameters were based on best practice recommendations from Cisco, such as the maximum packet drop probability and the high threshold value. However, the minimum threshold values were obtained from simulation. It was seen that the minimum threshold value should be 60% of the maximum threshold value to achieve maximal resource utilization.

Even though packet loss probability was the main objective, it was also important to observe the network dynamics concerning traffic flows. The TCP protocol traffic adjusts to the condition of a network since TCP aims to utilize the available bandwidth across a network based on congestion, by simply discarding packets which lowers source transmission rates.

The congestion windows' distribution was modelled with the Weibull distribution, which corresponds to the sampled values' frequency distribution. This finding further illustrates that there exist a linear relationship between the η (scaling parameter) and β (shape parameter) values of the Weibull probability distribution. Therefore, TCP transmission rates are controlled by the RED algorithm by dropping packets to indicate that the flows' congestion window value must decrease.

Network utilization was shown to be improved by the use of the RED algorithm. As the transmission nodes are controlled to have fair sharing of network resources, the traffic arrival rate closely resembled the traffic service rate. The utilization variation illustrated that utilization decreased with an increase of congestion and an increase in utilization once congestion decreased. Therefore, the variation showed that the TCP congestion window for the various sources was maintained equally by RED.

Further analysis showed that the queue probability for queuing delay as well as queue length had a *normal distribution* between the maximum and minimum threshold values. As a consequence, the queuing delay distribution within the network correlated to the queue length probability. This was shown by the delay probability graphs.

7.3. FUTURE CONTRIBUTIONS

This work studied the RED algorithm for network performance optimization for a single networking node under congestion. RED is the algorithm used in practice for congestion control. However, various active queue management implementations exist, such as *distributed weighted random early detection* (DWRED), *stabilized random early detection* (SRED), *adaptive random early detection* (ARED), and BLUE. Research could empirically compare the various schemes, which may lead to even better network performance.

REFERENCES

[1] P. Simpson, “Out with the old”, Telecommunications Magazine, volume 14 issue 3, June 2004 Pages: 103 – 106, Available at: http://www.telecommagazine.com/default.asp?journalid=3&func=articles&page=0406t13&year=2005&month=11. Last visited on 1 December 2012.

[2] W. Stallings, “ISDN and Broadband ISDN with Frame Relay and ATM”, fourth edition, Prentice Hall, 2001.

[3] W. Odom and M.J. Cavanaugh, “Cisco QoS exam certification guide”, second edition, Cisco Press, 2005.

[4] IETF Integrated Services Working Group, http://www.ietf.org/html.charters/IntServ-charter.html. Last visited on 1 December 2012.

[5] IETF Differentiated Services Working Group, http://www.ietf.org/html.charters/DiffServ-charter.html. Last visited on 1 December 2012.

[6] S. Blake, D. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for Differentiated Services”, Internet RFC 2475, December 1998.

[7] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, volume 1, issue 4, August 1993, Pages: 397 – 413.

[8] D.C. Verma, “Service level agreements on IP networks”, Proceedings of the IEEE, volume 92 issue 9, September 2004, Pages:1382 – 1388.

[9] J. Padhye, V. Firoiu, D. Townsley, and J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation”, in Proceedings of ACM SIGCOM, September 1998, Pages: 972 – 978.

[10] A. Adas, “Traffic Models in Broadband Networks”, IEEE Communications Magazine, volume 35 issue 7, July 1997, Pages: 82 – 87.

[11] V. Paxson and S. Floyd, "Wide-area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, volume 3 issue 3 June 1995 Pages: 226 – 244.

[12] N. Gershenfeld, “The Nature of Mathematical Modelling”, Cambridge University Press, 1998.

[13] I. Stoica, H. Zhang, T.S.E. Ng, “A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services”, IEEE/ACM Transactions on Networking, volume 8, issue 2, April 2000, Pages:185 – 199.

[14] Y. Kim, S.Q. Li, “Time-scale of interest in traffic management for link bandwidth allocation design”, Proceeding of the IEEE Infocom conference, 1996, Pages: 738–748.

[15] M.S. Crouse, R.H. Riedi, V.J. Ribeiro, and R.G. Baraniuk, ”A multifractal wavelet model for positive processes”, Time-Frequency and Time-Scale Analysis, Proceedings of the IEEE-SP International Symposium, 6 – 9 October 1998, Pages:341 – 344.

[16] W. Willinger, M. S. Tuqqu, W. E. Leland, and D. V. Wilson; “Self-similarity in high-speed packet traffic: Analysis and modelling of Ethernet traffic measurements”, Statistical Science, volume 10 issue 1, 1995, Pages: 67 – 85.

[17] H.F. Zhang, Y.T. Shu, and O. Yang, "Estimation of Hurst parameter by variance-time plots", IEEE Pacific Rim Conference, volume 2, 20-22 August 1997, Pages: 883 – 886.

[18] R.L. Cruz, "Quality of service guarantees in virtual circuit switched networks", IEEE Journal, volume 13, issue 6, August 1995, Pages: 1048 – 1056.

[19] E. W. Knightly and H. Zhang, "D-BIND: an accurate traffic model for providing QoS guarantees to VBR traffic," IEEE ACM Transactions on Networking, volume 5, issue 2, April 1997, Pages 219 – 231.

[20] J. Turner, "New Directions in Communications (*or Which Way to the Information Age)*", IEEE Communications. Magazine, volume 24, October 1986, Pages: 8 – 15.

[21] A. Elwalid, D. Mitra, R.H. Wentworth, "A New Approach for Allocating Buffers and Bandwidth to Heterogeneous, Regulated Traffic in an ATM Node"**,** IEEE Journal, volume 13, issue 6, August 1995, Pages: 1115 – 1127.

[22] G. Procissi, M. Gerla, J. Kim, S. S. Lee, and M. Y. Sanadidi, "On Long Range Dependence and Token Buckets", Proceedings of SPECTS, July 2001.

[23] Y. Li, G. Liu; H. Li, and X. Hou, "Wavelet-based analysis of Hurst parameter estimation for self-similar traffic**",** Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), volume 2, May 2002, Pages: 2061 – 2064.

[24] Z. Fan and P. Mars, "Self-similar traffic generation and parameter estimation using wavelet transform**",** Global Telecommunications Conference, GLOBECOM '97, IEEE Volume 3, 3 – 8 November 1997, Pages: 1419 - 1423.

[25] M. Lelarge, Z. Liu and C.H. Xia, "Asymptotic tail distribution of end-to-end delay in networks of queues with self-similar cross traffic", INFOCOM , Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, volume 4, 7 – 11 March 2004, Pages: 2352 – 2363.

[26] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. "On the self-similar nature of Ethernet Traffic (extended version)". IEEE/ACM Transactions on Networking, volume 2, issue 1, February 1994, Pages: 1 – 15.

[27] B. Mandelbrot and J. Van Ness, "Fractional Brownian motion, fractional noises and applications", SIAM Review, volume 10, October 1968, Pages: 422 – 437.

[28] L. Kleinrock, Queuing Systems, Vol. 1: Theory, John Wiley, 1975.

[29] L. Kleinrock, Queuing Systems, Vol. 2: Computer Applications, John Wiley, 1976.

[30] D K. Thompson, G. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics", IEEE Network, volume 11, issue 6, November/December 1997, Pages: 10 – 23.

[31] D. D. Clark, S. Shenker, and L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", Proceeding ACM SIGComm Symposuim, August 1992, Pages: 14 – 26.

[32] S. Floyd and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions Networking, volume 3, issue 4, August 1995, Pages: 365 – 386.

[33] J. van Greunen and R. Achterberg , "A Network Operator's perspective on IP VPNs", Proceedings of the Southern African Telecommunication Networks and Applications Conference (SATNAC) conference, George, September 2003.

[34] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", Internet RFC 2474, December 1998.

[35] Cisco Systems, "Class Based Weighted Fair Queueing", 2004. Available at http://cco.cisco.com/en/US/tech/tk543/tk544/tk96/tsd_technology_support_sub-protocol_home.html. Last visited on 1 December 2012.

[36] Cisco Systems, "Low Latency Queuing", 2004. Available at http://cco.cisco.com/en/US/tech/tk543/tk544/tk399/tsd_technology_support_sub-protocol_home.html. Last visited on 1 December 2012.

[37] X. Wang and H. Schulzrinne, "Pricing Network Resources for Adaptive Applications in a Differentiated Services Network", Proceedings of the IEEE INFOCOM, volume 2, 2001, Pages: 943 – 952.

[38] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP)", Internet RFC 2205, September 1997.

[39] P.P.White, "RSVP and Integrated Services in the Internet: A Tutorial", IEEE Communications Magazine, volume 35, issue 5, May 1997, Pages: 100 – 106.

[40] F. Baker, C. Iturralde, F. Le Faucheur and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", Internet RFC 3175, September 2001.

[41] J. Postel, "Internet Protocol - DARPA Internet Program Protocol Specification", Internet RFC 791, USC/Information Sciences Institute, September 1981.

[42] Q. Wang, J. M. Peha, and M. A. Sirbu, "The Design of an Optimal Pricing Scheme for ATM Integrated-Services Networks", Presented at MIT Workshop on Internet Economics, March 1995.

[43] Cisco Systems, "Layer 3 VPNs", 2003. Available at: http://www.cisco.com/en/US/products/ps6604/products_ios_protocol_group_home.html. Last visited on 1 December 2012.

[44] G. Jin and B. Tierney, "Netest: A Tool to Measure Maximum Burst Size, Available Bandwidth and Achievable Throughput", Proceedings of the 2003 International Conference on Information Technology, August 10 – 13, 2003, Newark, New Jersey, LBNL-48350.

[45] G. Jin and B. Tierney, "System Capability Effects on Algorithms for Network Bandwidth Measurement", Proceedings of the Internet Measurement Conference October 27 – 29, 2003, Miami, Florida, LBNL-48556.

[46] M. Yuksel, K. K.Ramakrishnan, S. Kalyanaraman, J.D. Houle, R. Sadhvani, "Value of Supporting Class-of-Service in IP Backbones", Proceedings of the IEEE International Workshop on Quality of Service, 21 – 22 June 2007, Pages: 109–112.

[47] H. Johnson, M. Graham, "High-speed signal propagation: Advanced black magic", Prentice Hall, 2003.

[48] G. Bolch, S. Greiner, H. de Meer and K.S. Trivedi, "Queuing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications", John Wiley, 1998.

[49] C. Krintz and B. Calder, "Reducing Delay With Dynamic Selection of Compression Formats", Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing, 2001, Page: 266.

[50] R. Zurawski, "RTP, RTCP and RTSP protocols, The industrial information technology handbook", CRC Press, 2004.

[51] V. Paxson , J. Kurose , C. Partridge , E. W. Zegura, "End-to-End Routing Behavior in the Internet", IEEE/ACM Transactions on Networking, volume 5, issue 5, February 1996, Pages: 601 – 615.

[52] J. Heinanen and R. Guerin, "A Single Rate Three Color Marker", Internet RFC 2697, September 1999.

[53] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker", Internet RFC 2698, September 1999.

[54] W. Fang and N. Seddigh, "Time Sliding Window Three Color Marker", Internet RFC 2859, March 2000.

[55] O. Bonaventure and S. De Cnodder, "Rate Adaptive Shaper for Differentiated Services", Internet RFC 2963, July 2000.

[56] W. Luo, C. Pignataro, A. Chan and D. Bokotey, "Layer 2 VPN Architectures", Cisco Press, ISBN: 978-1-58705-168-5, 2005.

[57] Javvin Technologies, "Network Protocols Handbook", Javvin Technologies Inc., ISBN: 978-0-97409-452-6, 2005.

[58] Microsoft Press, "PC 97 Hardware Design Guide", Microsoft Press, ISBN: 1-57231-381-1, 1997.

[59] T. Bradley, C. Brown, and A. Malis, "Multiprotocol Interconnect over Frame Relay", Internet RFC 1490, July 1993.

[60] D. Teare, "Designing Cisco Networks", Cisco Press, July 1999

[61] D. E. McDysan, D. L. Spohn, "ATM Theory and Applications", McGraw-Hill. ISBN 0-07-045346-2, 1999.

[62] A. Malis, W. Simpson, "PPP over SONET/SDH", Internet RFC 2615, June 1999.

[63] W. Simpson, "The Point-to-Point Protocol", Internet RFC 1661, July 1994.

[64] ATM Forum, "The ATM Forum", 2002-2004. Available at: http://ftp.atmforum.com/. Last visited: 14 October 2005

[65] J. Postel, "User datagram protocol", Internet RFC 768, August 1980.

[66] J. Postel, "Transmission control protocol", Internet RFC 793, September 1981.

[67] Sprint IPMON DMS, "Packet Trace Analysis", 2001-2005. Available at: https://research.sprintlabs.com/packstat/packetoverview.php. Last visited: 1 December 2012.

[68] J. Postel, J. Reynolds, “File transfer protocol”, Internet RFC 959, October 1985.

[69] C. Hedrick, “Routing information protocol”, Internet RFC 1058, June 1988.

[70] J. Moy, “The OSPF specification”, Internet RFC 1131, October 1989.

[71] D. Oran, “OSI IS-IS inter-domain routing protocol”, February 1990.

[72] S V. Jacobson, K. Nichols, and K. Poduri, “An Expedited Forwarding PHB”, Network Working Group, RFC 2598, June 1999.

[73] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, “Assured Forwarding PHB Group”, Network Working Group, RFC2597, June 1999.

[74] “The Network Simulator: Building Ns”, Available at: http://www.isi.edu/nsnam/ns/ns-build.html. Last visited on 1 December 2012.

[75] J. van Greunen, V. Naidoo, H. Morar, “Dimensioning and provisioning links for IP quality of service”, unpublished.

[76] M.A. Stephens, “Test of fit for the logistic distribution based on the empirical distribution function”, Biometrika, volume 66, issue 3, 1979, Pages: 591-595.

APPENDIX A. STATISTICAL BACKGROUND

This section describes basic statistical definitions used throughout this dissertation.

A.1 STATISTICAL PROCESS

These processes provide a means to analyse network traffic, due to the fact that measured values change unpredictably over time. Because of sampling dependencies across a network, it is possible for future predictions to be made using historic values.

Given a series of N number of random samples, $X_i = \{X_o, X_1, ..., X_N\}$, over a certain sampling time, a process could be modelled deterministically or stochastically. Stochastic models partially account for future predictions, which aid to describe most network traffic models. However, deterministic models exactly describe random processes over a given time series, for example regulated network traffic.

A.2 AVERAGE FUNCTION

The first moment of a stochastic process is known as the expected value, or mean. The mean is calculated using,

$$E[X] = \mu = \frac{\sum_{t=1}^{N} X_t}{N} \tag{A.1}$$

Variable X_t represents the sampled values and N the number of measured values.

A.3 VARIANCE FUNCTION AND STANDARD DEVIATION

The second moment of a stochastic process is known as the variance. The variance is calculated using,

$$VAR[X] = \sigma^2 = \frac{\sum_{t=1}^{N} (X_t - \mu)^2}{N} \tag{A.2}$$

Variable X_t represents the sampled values, μ the mean, and N the number of measured values.

Another variable which is commonly used is the standard deviation variable, defined as

$$s = \sqrt{\sigma^2} \tag{A.3}$$

A.4 AUTOCORRELATION FUNCTION

The autocorrelation function (ACF) is used to calculate the linear predictability from a series of random variables calculated by,

$$\rho(k) = \frac{\sum_{t=1}^{N}(X_t - \mu)(X_{t+k} - \mu)}{N} \tag{A.4}$$

Since equation (A.4) delivers large values, it becomes more appropriate to normalize the correlation values, as follows,

$$\rho_n(k) = \frac{E[\sum_{t=1}^{N}(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2} \tag{A.5}$$

A.5 PROBABILITY DENSITY FUNCTION

A density function describes a distribution, which is used to estimate the likelihood of an occurrence. This function is mainly estimated by mean and variance values. A distribution such as Poisson only requires the mean value,

$$P[X = x] = Poisson(X = x) = \frac{\mu^x e^{-\mu}}{x!} \tag{A.6}$$

whereas the normal distribution requires the mean and variance values

$$P[X = x] = Normal(X = x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{A.7}$$

A.5 COMMULATIVE DISTRIBUTION FUNCTION

The cumulative distribution function aids in the interpretation of the probability density function. This function sums all the values of the probability density function,

$$F(X = x) = P[X \leq x] \tag{A.8}$$

Since some densities become heavy-tailed, this method provides a way to interpret the data more clearly.

A.6 GOODNESS-OF-FIT TEST

The *Kolmogorov-Smirnov* (K-S) and *Anderson-Darling* (A-D) tests are used to determine if captured data samples conform to a statistical distribution. These tests are used to determine if a normal distribution can be used to analyse the data samples. The K-S test focuses on the centre of a specified distribution whereas the A-D test focuses on the tails.

Kolmogorov-Smirnov

The *Kolmogorov-Smirnov* (K-S) test is based on the *empirical distribution function* (ECDF). Given N ordered data points, $X_i = \{X_o, X_1, ..., X_N\}$, the ECDF is defined as:

$$E_N = \frac{n(i)}{N} \tag{A.9}$$

where n(i) is the number of points less than X_i and where the X_i values are ordered from smallest to largest value. This is a step function that increases by $\frac{1}{N}$ at each sampled value of the ordered data point from large to small. The hypothesis definition is,

Hypothesis: H_0: The data samples $X_i = \{X_o, X_1, ..., X_N\}$ follow a specified distribution.

H_a: The data samples $X_i = \{X_o, X_1, ..., X_N\}$ do not follow a specified distribution.

The Kolmogorov-Smirnov test statistic is defined as

$$KS = \max_{1 \le i \le N} \left\{ F(X_i) - \frac{i-1}{N}, \frac{i}{N} - F(X_i) \right\} \tag{A.10}$$

where F is the theoretical cumulative distribution of the distribution being tested and a continuous distribution (no discrete distributions), and it must be fully specified where parameters such as location, scale, and shape cannot be estimated from the samples.

Anderson-Darling

The *Anderson-Darling* (A-D) test verifies if data samples can be modelled with a specific distribution. The A-D test is a modified K-S test that focuses more on the tails of a distribution than the K-S test. The A-D test also makes use of the specific distribution in calculating critical values. This has the advantage of allowing a more sensitive test and the disadvantage that critical values must be calculated for each distribution. The A-D test is an alternative to the chi-square and K-S goodness-of-fit tests. The hypothesis definition is

Hypothesis:

H_0: The data samples $X_i = \{X_o, X_1, ..., X_N\}$ follow a specified distribution.

H_a: The data samples $X_i = \{X_o, X_1, ..., X_N\}$ do not follow a specified distribution.

The Anderson-Darling test statistic is defined as

$$AD = -N - S \tag{A.11}$$

where

$$S = \sum_{i=1}^{N} \frac{(2i-1)}{N} [\ln F(X_i) + \ln(1 - F(X_{N+1-i}))] \tag{A.12}$$

and F is the cumulative distribution function of the specified distribution. Note that the X_i is ordered from small to large data.

APPENDIX B. THE WEIBULL DISTRIBUTION

This chapter illustrates the method used to obtain the Weibull distribution from sampled values. Therefore, this section should aid as a guide to implement the Weibull distribution. The reason for this guide is that the Weibull distribution forms part of *extreme value theory* (EVT). As a result, more calculation steps are needed than merely calculating the μ and σ^2 of the distribution.

B.1 MEDIAN RANKS

Median ranking gives a specific confidence level at 50% to all data points. This method is used to calculate estimation values, which is,

$$MR\% = F(i) = \frac{i - 0.3}{N + 0.4} \tag{B.1}$$

where *F(i)* is the ranking of a sample number *i* and *N* is the number of samples taken.

B.2 WEIBULL DISTRIBUTION

This distribution in its most general form consists of 3 variables, namely η, β and γ as shown as,

$$f(x) = \frac{\beta}{\eta}\left(\frac{x-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{x-\gamma}{\eta}\right)^{\beta}} \tag{B.2}$$

where η defines the scale parameter, β defines the shape parameter, and γ defines the probability offset for the Weibull distribution, as illustrated in Figure B.1.

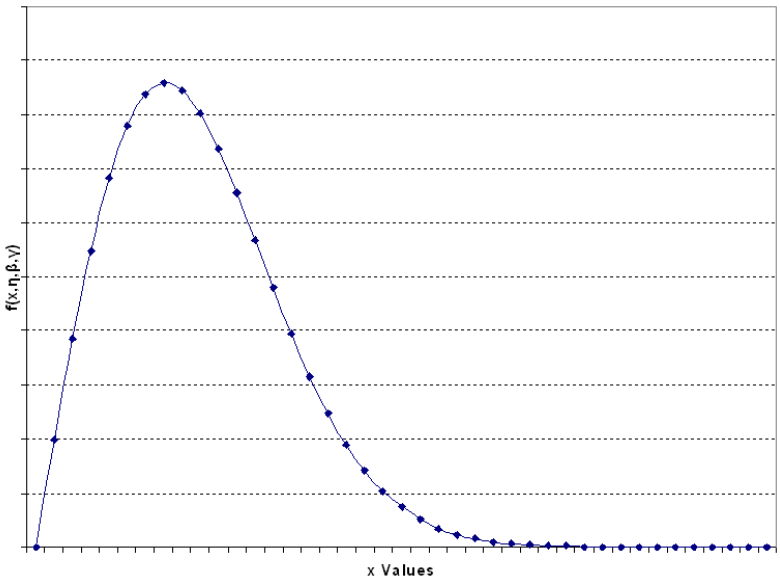


Figure B.1 The Weibull Distribution

This equation can be simplified by letting $\gamma = 0$, simplifying the distribution to

$$f(x) = \frac{\beta}{\eta}\left(\frac{x}{\eta}\right)^{\beta-1} e^{-\left(\frac{x}{\eta}\right)^{\beta}} \tag{B.3}$$

Using equation (B.3), the η and β values are calculated as follows:
By differentiating equation (B.3), the cumulative density function (CDF) is given as,

$$F(x) = 1 - e^{-\left(\frac{x}{\eta}\right)^{\beta}} \tag{B.4}$$

Equation (B.4) shows that the natural logarithm is obtained from both sides equation (B.4), to obtain

$$\ln\{1 - F(x)\} = -\left(\frac{x}{\eta}\right)^{\beta} \tag{B.5}$$

$$\therefore \ln\{-\ln\{1-F(x)\}\} = \beta \ln\left(\frac{x}{\eta}\right) \tag{B.6}$$

$$\therefore \ln\{-\ln\{1-F(x)\}\} = -\beta \ln \eta + \beta \ln x \tag{B.7}$$

Equation (B.7) provides a linear fit for data analysis. Set $y = \ln\{-\ln\{1-F(x)\}\}$, $a = -\beta \ln \eta$ and $b = \beta$ to obtain the linear equation,

$$y = a + bx \tag{B.8}$$

Regression analysis, also known as the least squared estimation method, is used to obtain the estimates $\hat{a}$ and $\hat{b}$, as shown in equations (B.8) and (B.9), where x_i is the "sample taken at" value and y_i is the sampled value.

$$\hat{a} = \frac{\sum_{i=1}^{N} y_i}{N} - \hat{b}\frac{\sum_{i=1}^{N} x_i}{N} = \bar{y} - \hat{b}\bar{x} \tag{B.9}$$

$$\hat{b} = \frac{\sum_{i=1}^{N} x_i y_i - \frac{\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N}}{\sum_{i=1}^{N} x_i^2 - \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N}} \tag{B.10}$$

For this case the linear values are obtained from equations (B.11) and (B.12) as follows,

$$y_i = \ln\{-\ln\{1 - F(x_i)\}\} \tag{B.11}$$

$$\therefore\ y_i = \ln x_i \tag{B.12}$$

APPENDIX C. SAMPLE SIMULATION RESULTS

The observations in this appendix support the findings in the main dissertation's statistics.

This section illustrates aggregate effects that the RED algorithm has on network traffic. It compares the effects that the various RED parameter values have on aggregate samples. A high threshold fraction results in buffer overflow, which shows weak utilization of network resources, as shown in section C.1. The same effect can be observed if the minimum and maximum queue threshold fraction of RED is set to a low, excessively dropping network packets, which results in resource under-utilization as shown in section C.2. However, this fraction can be optimized to improve resource utilization as shown in section C.3.

C.1 High threshold value ($\alpha = 1.0$)

The purpose of this section is to show what the affects on network traffic is when the minimum and maximum queue threshold for RED is the same.

Packet loss

Packet loss occurs mainly because of buffer overflow, which leads to weak utilization since it could indicate constant congestion. As a consequence, source transmissions rates are random because the TCP protocol will be unable to determine if the network is congested. As can be seen from the Figure C.1, packet loss is random over time, which provides no information to the sources about the state of the network resources.

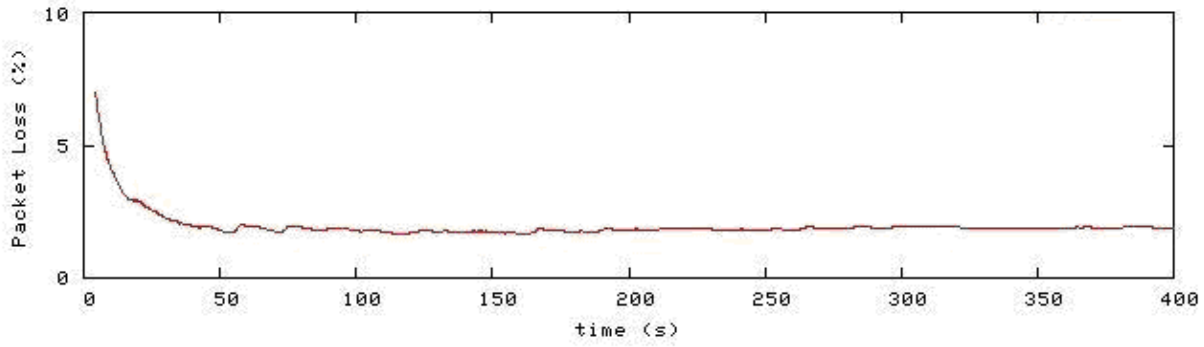


Figure C.1 Packet loss percentage vs. time

Congestion window

Source congestion window values vary drastically since packet loss is random, which results in random source transmission rates as can be seen from the time based congestion windows values in Figure C.2. It is observed that certain flows over-utilize network resources, depriving other sources from utilizing the resources by the high transmission rate.

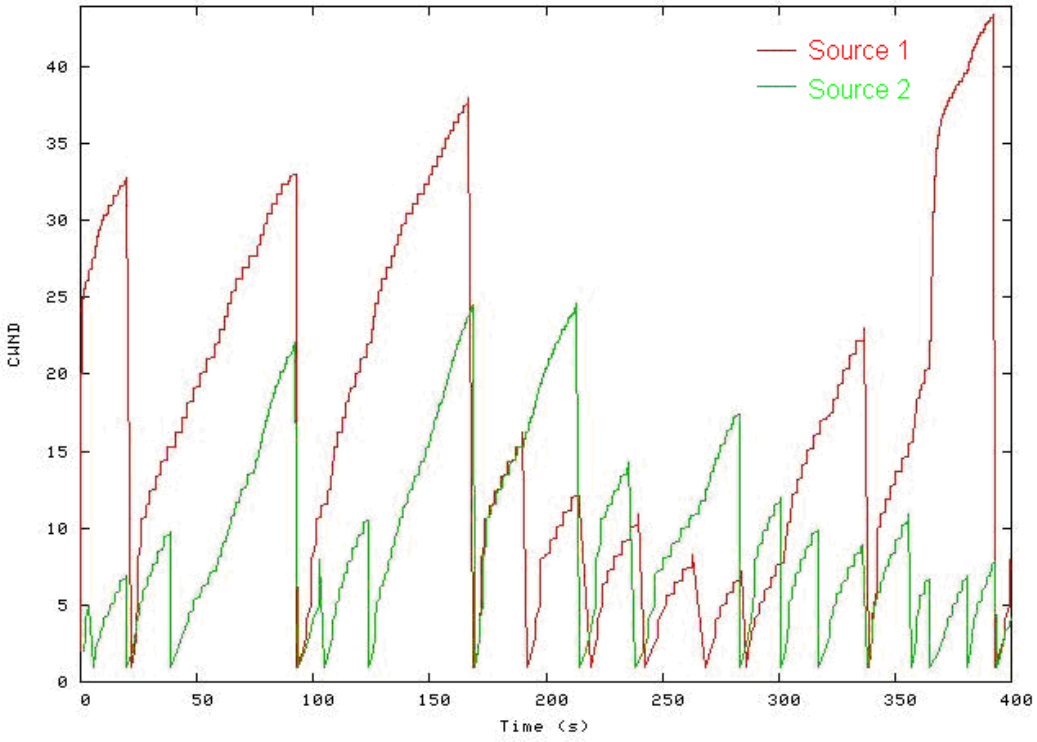


Figure C.2 TCP congestion windows vs. time

Queuing behaviour

Queues are constantly filled, which also introduces high delay measures to traffic flows. This observation also shows that buffer overflow is the cause by which a resource can be over-utilized with the values higher than the maximum threshold.

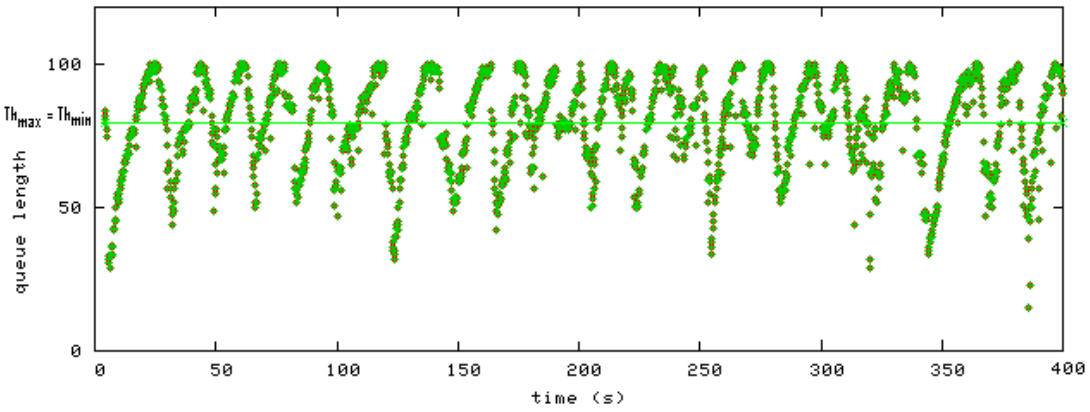


Figure C.3 Queue length vs. time

Round trip times

Packets experience high delay when queues are filled, as stated by Little's theorem, shown in Figure C.4. TCP retransmits packets if acknowledgement packets are not received from the recipient node, because of time-out timers within its protocol. Packets are assumed to be lost if the recipient node does not verify that packets were received. As a result, unnecessary data will be transmitted, resulting in resource under utilization.

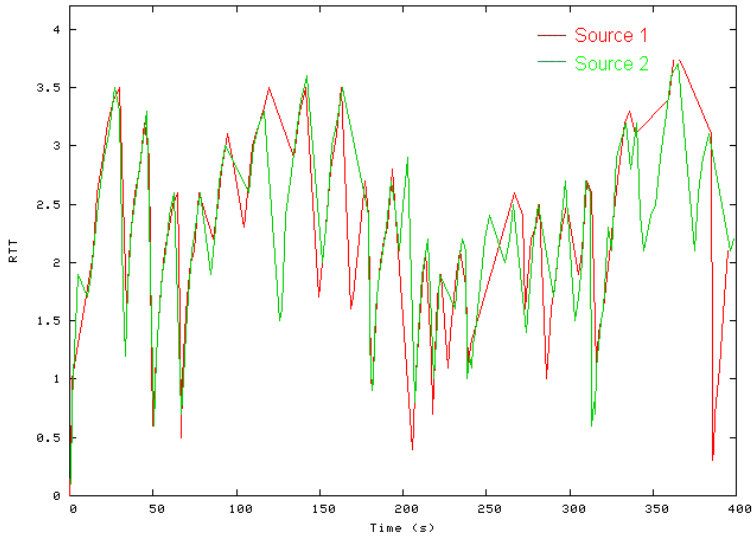


Figure C.4 Round trip time vs. time

C.2 Medium threshold value (α = 0.6)

The purpose of this section is to show what the affects on network traffic are when the minimum and maximum queue threshold for RED is 0.6.

Packet loss

This threshold value indicates that packet loss stays constant, which indicates that the network resources are effectively utilized. The TCP sources are able to determine network congestion more accurately.

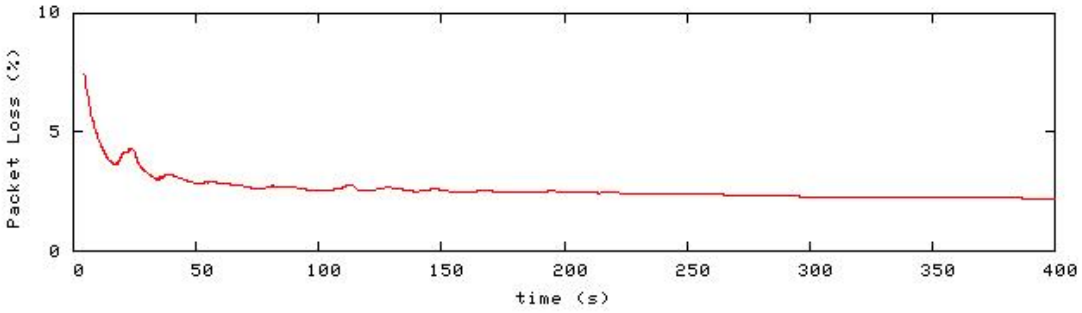


Figure C.5 Packet loss percentage vs. time

Congestion window

Sources are able to determine an optimal data transmission rate with the stabilization of packet loss. Source congestion windows, of all the transmitting sources, are likely to be similar as seen in Figure C.6. Thus, TCP sources utilize the network resources more effectively.

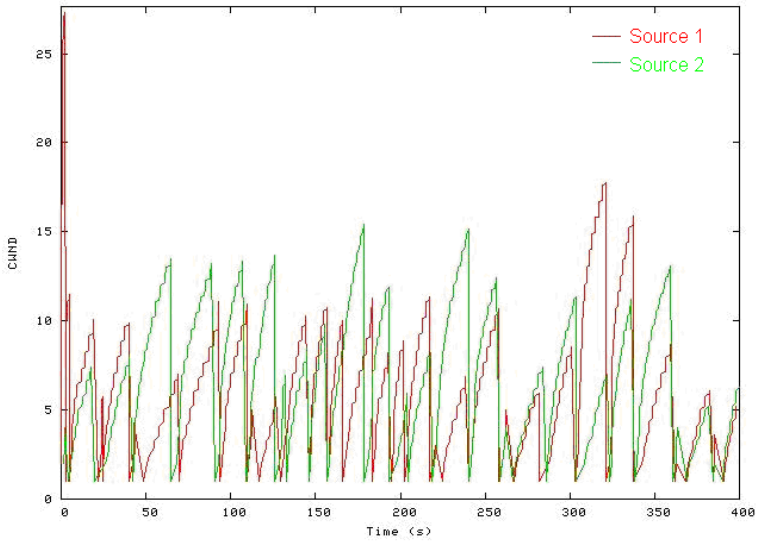


Figure C.6 TCP congestion windows vs. time

Queuing behaviour

Figure C.7 illustrates that queues are serviced more effectively, as it is observed that the queues empties over time. Therefore, packet delay will decrease causing less packet retransmissions. As a result, the absence of unnecessary data will result in better resource utilization.

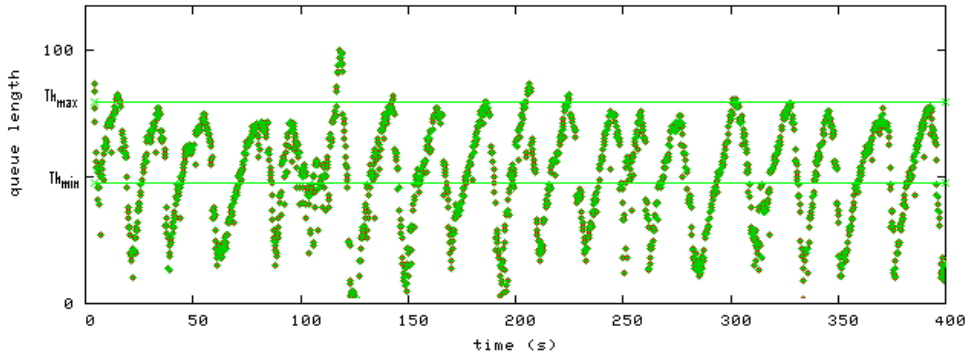


Figure C.7 Queue length vs. time

Round trip times

Packets are less likely to experience delay, which results in minimal packet retransmissions that cause weak resource utilization. This can be seen by comparing Figure C.8 with Figure C.4.

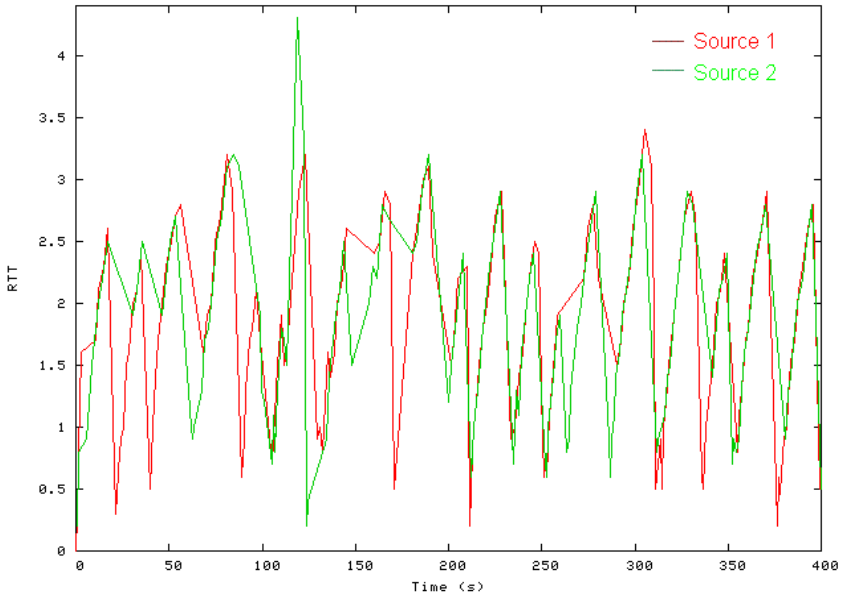


Figure C.8 Round trip time vs. time

C.3 Low Threshold Value (α = 0.0)

The purpose of this section is to show what the affects on network traffic are when the minimum and maximum queue threshold for RED is 0.

Packet loss

Over excessive packet loss is experienced if the threshold factor is set too low. With this example all packets are discarded by the RED mechanism, which under-utilizes the available resources.

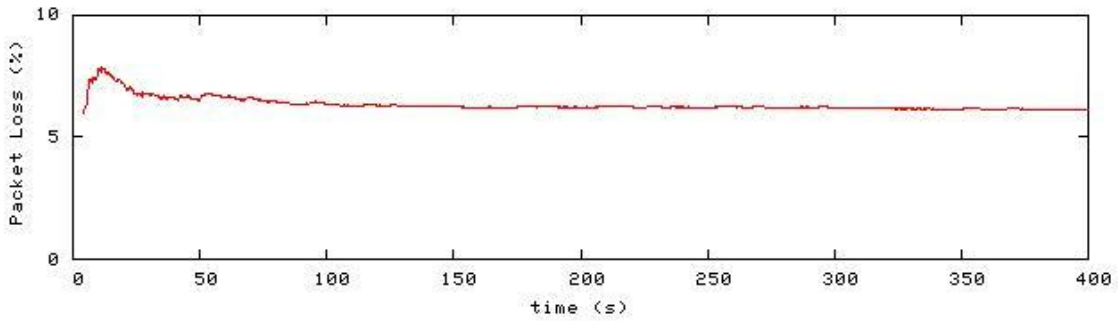


Figure C.9 Packet loss percentage vs. time

Congestion window

TCP source transmission rates are forces to be minimal because of the over excessive packet loss as shown in Figure 10.C. Even though the queue is not filled, the algorithm will start to discard packets immediately.

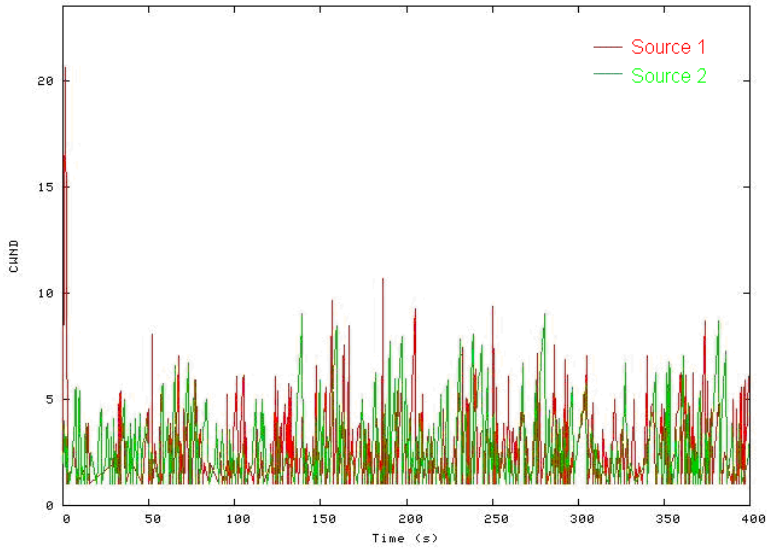


Figure C.10 TCP Congestion window vs. time

Queuing behaviour

Queue lengths are low since packets are very likely to be discarded as shown in Figure C.11. Packets experience little delay and are serviced faster.

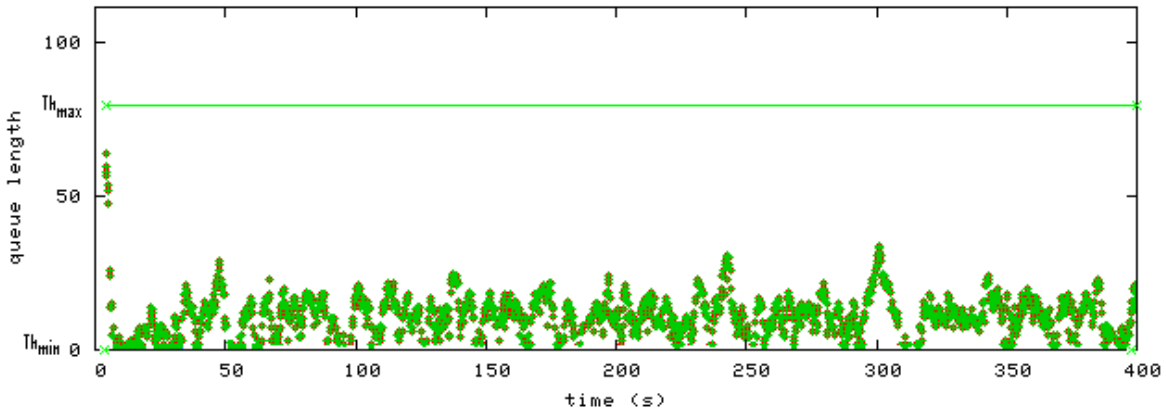


Figure C.11 Queue length vs. time

Round trip times

Packets that enter a queue are more likely to be discarded. Therefore, high packet retransmissions will result because of packet loss than time-outs. Figure C.12 shows that packets that are transmitted to the end-host will be fast as a result.

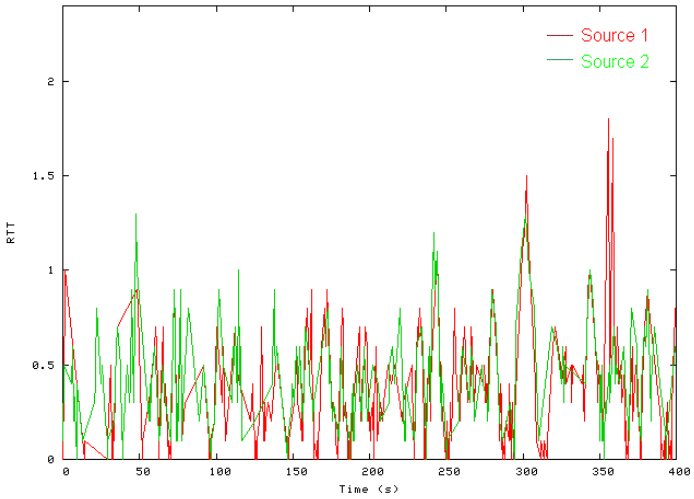


Figure C.12 Round trip time vs. time

APPENDIX D. RED IMPLEMENTATION WITHIN NS2

This section provides sample code implemented within NS2 for the variants of the RED algorithm.

D.1 Information stored for each physical queue

```
struct qParams {
   edp edp_;            // early drop parameters (defined in red.h)
   edv edv_;            // early drop variables (defined in red.h)
   int qlen;            // actual (not weighted) queue length in packets
   double idletime_;  // needed to calculate average queue
   bool idle_;          // needed to calculate average queue
};
```

D.2 Physical Queue Definition

```
class redQueue {
public:
   // Constructor
   redQueue();

   // The current number of precedence levels (or virtual queues)
   int numPrec;
   int qlim;
   int qlen;

   // Maximum Burstiness: maximum number of packets stored in the queue
   int qMaxBur;

   // Variable needed to select queue management type
   mredModeType mredMode;

   // Configures one virtual RED queue
   void config(int prec, const char*const* argv);
```

```
    // Initializes RED state variables
    void initREDStateVar(void);
    // updates RED variables after dequing a packet
    void updateREDStateVar(int prec);

    // Enques packets into a physical queue
    int enque(Packet *pkt, int prec, int ecn);

    // Deques packets
    Packet* deque(void);
    // Queries for the weighted queue length of a physical queue
    double getWeightedLength();
    // Queries for the queue length of a physical queue
    int getRealLength(void);
    // Queries for the length of a virtual queue
    int getVirtQueueLen(int prec) {return qParam_[prec].qlen;};
    // sets packet time constant values
    //(needed for calc. avgQSize) for each virtual queue
    void setPTC(double outLinkBW);
    void getPTC();
    // sets mean packet size (needed to calculate avg. queue size)
    void setMPS(int mps);
private:
    PacketQueue *q_;

    // Function used to maintain parameters for each of the virtual queues
    qParam qParam_[MAX_PREC];

    // Function used to calculate the average queue size of a virtual queue
    void calcAvg(int prec, int m);
};
```

D.3 Function that updates the RED state variables for a virtual queue

```
void redQueue::updateREDStateVar(int prec) {
  int idle = 1;
  int i;
  double now = Scheduler::instance().clock();
  qParam_[prec].qlen--;       // decrement virtual queue length

  if (qParam_[prec].qlen == 0)
  {
    if (mredMode == rio_c) {
      for(i=0; i<prec; i++) if (qParam_[i].qlen != 0) idle = 0;
      if (idle) {
        for (i=prec;i<numPrec;i++) {
          if (qParam_[i].qlen == 0) {
            qParam_[i].idle_ = 1;
            qParam_[i].idletime_ = now;
          } else break;
        }
      }
    } else if (mredMode == rio_d) {
      qParam_[prec].idle_ = 1;
      qParam_[prec].idletime_ = now;
    } else if (mredMode == wred) { //wred
      qParam_[0].idle_ = 1;
      qParam_[0].idletime_ = now;
    }
  }
}
```

D.4 Function when a packet is received

```
int redQueue::enque(Packet *pkt, int prec, int ecn) {
  int m = 0;
  double now, u;
  double pa, pb;

  // Value returned when a packet is dropped due to a mechanism
  if ((mredMode == dropTail)&&(qParam_[prec].edp_.th_min==-1))
      return PKT_EDROPPED;

  // Value returned when a packet is dropped due to buffer overflow
  if (q_->length() > (qlim-1)) return PKT_DROPPED;
  now = Scheduler::instance().clock();

  //now determining the avg for that queue
   if (mredMode == dropTail) {
      if (q_->length() >= qParam_[0].edp_.th_min) {
         return PKT_EDROPPED;
      } else {
         q_->enque(pkt);
         qlen++;
         qParam_[prec].qlen++;
         qMaxBur=(qMaxBur>qlen?qMaxBur:qlen);
         return PKT_ENQUEUED;
      }
   } else if (mredMode == rio_c) {
     for (int i = prec; i < numPrec; i++) {
       m = 0;
       if (qParam_[i].idle_) {
         qParam_[i].idle_ = 0;
         m = int(qParam_[i].edp_.ptc * (now - qParam_[i].idletime_));
       }
       calcAvg(i, m+1);
```

```
    }
  } else if (mredMode == rio_d) {
    if (qParam_[prec].idle_) {
      qParam_[prec].idle_ = 0;
      m = int(qParam_[prec].edp_.ptc * (now - qParam_[prec].idletime_));
    }
    calcAvg(prec, m+1);
  } else { //wred
    if (qParam_[0].idle_) {
      qParam_[0].idle_ = 0;
      m = int(qParam_[0].edp_.ptc * (now - qParam_[0].idletime_));
    }
    calcAvg(0, m+1);
  }

  // Enqueue packet if we are using ecn
  if (ecn) {
    q_->enque(pkt);
    qlen++;
    qMaxBur=(qMaxBur>qlen?qMaxBur:qlen);
    //virtually, this new packet is queued in one of the multiple queues,
    //thus increasing the length of that virtual queue
    qParam_[prec].qlen++;
  }

  //If the average variable is greater than the min threshold,
  //there can be only two cases.....
  if (qParam_[prec].edv_.v_ave > qParam_[prec].edp_.th_min) {
    //either the avg is less than the max threshold
    if (qParam_[prec].edv_.v_ave <= qParam_[prec].edp_.th_max) {
      //in which case determine the probabilty for dropping the packet,
      qParam_[prec].edv_.count++;
      qParam_[prec].edv_.v_prob = (1/qParam_[prec].edp_.max_p_inv) *
```

```
                    (qParam_[prec].edv_.v_ave-qParam_[prec].edp_.th_min) /
                    (qParam_[prec].edp_.th_max-qParam_[prec].edp_.th_min);
      pb = qParam_[prec].edv_.v_prob;
      pa = pb/(1.0 - qParam_[prec].edv_.count*pb);
      //now determining whether to drop the packet or not
      u = Random::uniform(0.0, 1.0);

      //drop it
      if (u <= pa) {
        if (ecn) return PKT_MARKED;
        return PKT_EDROPPED;
      }
    } else { //if avg queue is greater than max. threshold
      qParam_[prec].edv_.count = 0;
      if (ecn) return PKT_MARKED;
      return PKT_EDROPPED;
    }
  }
  qParam_[prec].edv_.count = -1;

  // If ecn is on, then the packet has already been queued
  if(ecn) return PKT_ENQUEUED;

  //If the packet survives the above conditions it
  //is finally queued in the underlying physical queue
  q_->enque(pkt);

  //Virtually, this new packet is queued in one of the multiple queues,
  //thus increasing the length of that virtual queue
  qParam_[prec].qlen++;
  qlen++;
  qMaxBur=(qMaxBur>qlen?qMaxBur:qlen);
  return PKT_ENQUEUED;
}
```

D.5 Function that updates the RED average queue length

```
void redQueue::calcAvg(int prec, int m) {
    float f;
    int i;

    f = qParam_[prec].edv_.v_ave;

    while (--m >= 1) {
        f *= 1.0 - qParam_[prec].edp_.q_w;
    }
    f *= 1.0 - qParam_[prec].edp_.q_w;

    if (mredMode == wred)
      for (i = 0; i < numPrec; i ++)
          qParam_[i].edv_.v_ave = f;
}
```